Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

AD/A-000086

# REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

| | |
|---|---|
| 1. REPORT NUMBER: STAN-CS-74-419 | 2. GOVT ACCESSION NO. |
| 3. RECIPIENT'S CATALOG NUMBER | |
| 4. TITLE *(and Subtitle)*: CONCEPTUAL MEMORY: A THEORY AND COMPUTER PROGRAM FOR PROCESSING THE MEANING CONTENT OF NATURAL LANGUAGE UTTERANCES. | 5. TYPE OF REPORT & PERIOD COVERED: technical, July 1974 |
| | 6. PERFORMING ORG. REPORT NUMBER: STAN-CS-74-419 also(AIM233) |
| 7. AUTHOR(s): Charles J. Rieger, III | 8. CONTRACT OR GRANT NUMBER(s): DAHC 15-73-C-0435 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS: Stanford University, Computer Science Dept., Stanford, California 94305 | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: ARPA Order No. 2494 |
| 11. CONTROLLING OFFICE NAME AND ADDRESS: ARPA/IPT, Attn.: Stephen D. Crocker, 1400 Wilson Blvd., Arlington, Va. 22209 | 12. REPORT DATE: July 1974<br>13. NUMBER OF PAGES |
| 14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*: ONR Representative: Philip Surra, Durand Aeronautics Bldg., Rm. 165, Stanford University, Stanford, California 94305 | 15. SECURITY CLASS. *(of this report)*: Unclassified<br>15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT *(of this Report)*

Releasable without limitations on dissemination.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

PRICES SUBJECT TO CHANGE

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

Humans perform vast quantities of spontaneous, subconscious computation in order to understand even the simplest language utterances. The computation is principally meaning-based. With syntax and traditional semantics playing insignificant roles. This thesis supports this conjecture by synthesis of a theory and computer program which account for many aspects of language behavior in humans. It is a theory of language and memory.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

Reproduced by
NATIONAL TECHNICAL INFORMATION SERVICE
U S Department of Commerce
Springfield VA 22151

N O T I C E

THIS DOCUMENT HAS BEEN REPRODUCED FROM THE BEST COPY FURNISHED US BY THE SPONSORING AGENCY. ALTHOUGH IT IS RECOGNIZED THAT CERTAIN PORTIONS ARE ILLEGIBLE, IT IS BEING RELEASED IN THE INTEREST OF MAKING AVAILABLE AS MUCH INFORMATION AS POSSIBLE.

STANFORD ARTIFICIAL INTELLIGENCE LABORATORY
MEMO AIM-233

COMPUTER SCIENCE DEPARTMENT
REPORT NO. STAN-CS-74-419

JULY 1974

# CONCEPTUAL MEMORY: A THEORY AND COMPUTER PROGRAM FOR PROCESSING THE MEANING CONTENT OF NATURAL LANGUAGE UTTERANCES

by

Charles J. Rieger, III

Abstract:

Humans perform vast quantities of spontaneous, subconscious computation in order to understand even the simplest language utterances. The computation is principally meaning-based, with syntax and traditional semantics playing insignificant roles. This thesis supports this conjecture by synthesis of a theory and computer program which account for many aspects of language behavior in humans. It is a theory of language and memory.

This research was supported in part by the Advanced Research Projects Agency of the Office of Defense under Contract No. DAHC 15-73-C-0435 and by NIMH under contract MH06645-12.

The views and conclusions in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U.S. Government.

Reproduced in the USA. Available from the National Technical Information Service, Springfield, Virginia 22151.

This document was generated by the Stanford Artificial Intelligence Laboratory's document compiler, "PUB", and reproduced on a Xerox Graphics Printer (XGP). I extend my thanks for PUB and the XGP software to the following people: Les Earnest, Larry Tesler, Rich Johnson, Brian Harvey, Martin Frost, Ralph Gorin, Tovar Mock, and the rest of the systems staff at SAIL.

I would also like to thank John McCarthy and Les Earnest, who are the benevolent deities at SAIL, and who have made the Laboratory the wonderful research experience it was for me.

# CONCEPTUAL MEMORY

## ABSTRACT

Humans perform vast quantities of spontaneous, subconscious computation in order to understand even the simplest natural language utterances. The computation is principally *meaning-based*, with syntax and traditional semantics playing insignificant roles. This thesis supports this conjecture by synthesis of a theory and computer program which account for many aspects of language behavior in humans. It is a theory of language and memory.

Since the theory and program deal with language in the domain of conceptual meaning, they are independent of language form and of any specific language. Input to the memory has the form of analyzed **conceptual dependency graphs** which represent the underlying meaning of language utterances. Output from the memory is also in the form of meaning graphs which have been produced by the active (inferential) memory processes which dissect, transform, extend and recombine the input graphs in ways which are dependent upon the meaning context in which they were perceived.

A memory formalism for the computer model is first developed as a basis for examining the inferential processes by which comprehension occurs. Then, the notion of *inference space* is presented, and **sixteen classes of conceptual inference** and their implementation in the computer model are examined, emphasizing the contribution of each class to the total problem of understanding. Among the sixteen inference classes are: **causative/resultative** inferences (those which explain and predict cause and effect relationships relative to the memory's model of the world), **motivational** inferences (those which infer the probable intentions of actors), **enabling** inferences (those which predictively fill out the circumstances which were likely to have obtained at the time of an action), **action prediction** inferences (those which make guesses about what a person might be expected to do in some situation), **knowledge propagation** inferences (those which predict what knowledge is available to a person, based on what the memory already knows or can infer he knows), **normative** inferences (those which assess the "normality" of a given piece of information), and **state duration** inferences (those which predict the probable duration of specific states in the world). All inferences are probabilistic, and "backup" is deemphasized as a programming tool.

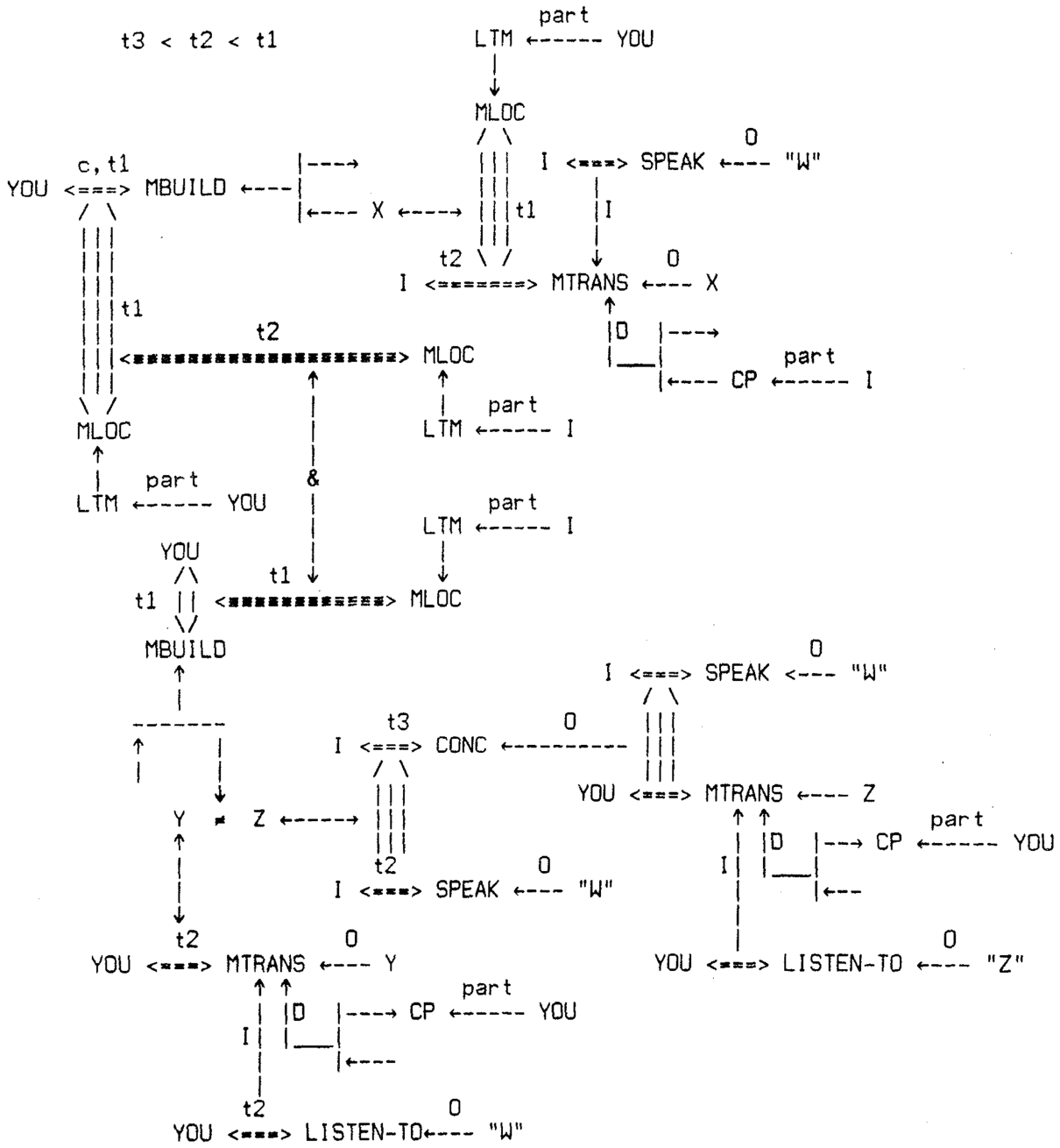
The idea of *points of contact* of information structures in inference space is explored. A point of contact occurs when an inferred unit of meaning from one starting point within one utterance's meaning graph either confirms (matches) or contradicts an inferred unit of meaning from another point within the graph, or from within the graph of another utterance. The quantity and quality of points of contact serve as the primary definition of understanding, since such points provide an effective measure of the memory's ability to relate and fill in information.

Interactions between the inference processes and (1) **word sense promotion** (how meaning context influences the language analyzer's choice of lexical senses of words during the parse), and (2) **the processes of reference** (how memory pointers to tokens of real world entities are established) are examined. In particular, an important inference-reference "relaxation cycle" is identified and solved.

The theory forms a basis for a computationally effective and comprehensive theory of language understanding by conceptual inference. Numerous computer examples are included to illustrate key points. Most issues are approached from both psychological and computational points of view, and the thesis is intended to be comprehensible to people with a limited background in computers and symbolic computation.

(Thesis committee: Profs. Roger Schank (advisor), Ken Colby, and Jerry Feldman, Computer Science Dept., Stanford University)

"I KNOW YOU BELIEVE YOU UNDERSTAND WHAT YOU THINK I SAID, BUT I AM NOT SURE YOU REALIZE THAT WHAT YOU HEARD IS NOT WHAT I MEANT."




If you are baffled by this modest conceptual graph, read on, read on. But do not despair... for now there is a computer program which is also baffled by it!

# ACKNOWLEDGMENTS

I extend my deep thanks and appreciation

To my advisor, Prof. Roger Schank, who set the stage for this research, and whose encouragement sustained me through difficult times. Our many arguments provided a constant source of intellectual stimulation, even if we were both wrong!

To Prof. Ken Colby and Prof. Jerry Feldman for their encouragement and comments, and for their forbearance concerning the eleventh hour push to get this manuscript in shape

To Prof. Gordon Bower, who spent a great deal of time and effort discussing and helping focus my ideas

To Ken Colby and to NSF who supported me during my graduate work

To Chris Riesbeck and Neil Goldman, whose programs grew up and played with my program

To the Stanford Ai Project PDP10, who had to listen to every word of this thesis. One of its keyboards is surely on the verge of a breakdown (the one in 251E, fellows). Since I am still alive and well, I must conclude the machine possesses a certain primitive benevolence which prevented it from zapping my typing finger with 110 volts. It is a mellow machine indeed.

To Dick and Janet Sweet, our dearest friends, who took us in after the movers came...

To David Levy and Paul Martin, who helped in the final editing operations

To my parents and parents-in-law, whose vicarious enthusiasm for my work, and whose love, have contributed to my peace of mind in so many subtle ways

To Andy, my son -- the greatest little guy in the whole world -- who, on numerous occasions, refrained from dismembering preliminary manuscripts of this thesis, and who graciously consented, if I read him right, to appearing as a character in chapter 8.

To my wife Linda, for being everything to me always

# TABLE OF CONTENTS


** Chapter 1. Introduction 1

1.1 Essence 1

1.2 Challenge 2

1.3 Perspective 3

1.4 General Goals 6

1.5 Tenets 7

1.6 Specific Goals 9

1.6.1 Themes 10

1.7 The Program and Some Assumptions It Makes 11

1.7.1 The Conceptual Analyzer, Briefly 11
1.7.2 The Conceptual Generator, Briefly 12
1.7.3 Where the Memory Fits, All Things Considered 12
1.7.4 Overview of the Program's Operation 14
1.7.5 Computational Issues 16
1.7.6 Examples 16
1.7.7 About the Computer Examples 19

1.8 What This Is: Comments 20

1.9 Related Work 21

1.10 Reading This Monster 22

** Chapter 2. Representation: The Approach to Meaning 24

2.1 Concerning Representational Formalisms 24

2.2 Conceptual Dependency: An Overview 25

2.2.1 Action Primitives 26
2.2.1.1 Conceptual Cases
2.2.1.2 Scope of the Action Primitives
2.2.2 The Primitive Acts 29
2.2.2.1 The Dummy DO
2.2.3 Picture Producers 35
2.2.4 Mental Locations 36
2.2.5 States and Attribution 37
2.2.6 State Scales and Statechanges 39
2.2.7 Adjectives and Relative Clauses 42
2.2.7.1 The REL Link
2.2.8 Causality 43
2.2.8.1 The Conceptual Causal Link
2.2.8.2 Conditional Causality

2.2.9 Instrumentality 46
2.2.10 Time 48
2.2.11 Interrogatives 49
2.2.12 Capability and Negation 50

2.3 Conclusion 51

Chapter 3. Representation: The Conceptual Memory 52

** 3.1 What Needs Representing in Memory 52

3.2 Design Criteria 53

3.2.1 Representing Knowledge: Problems of Total Referenceability 54

3.3 Concepts and Tokens 59

** 3.3.1 Comments on Notation 62
3.3.2 The Logical Organization: Two Important Relations 63
3.3.2.1 "NAME"
3.3.2.2 "ISA"
3.3.3 "Meta" Properties of Concepts and Tokens 66
3.3.3.1 Two "Meta" Properties Related to the Conceptual Analyzer and Generator
3.3.3.2 Three Other Inference-related Properties of Conceptual Predicates

3.4 Storing Conceptual Information 69

3.4.1 Bonds 69
* 3.4.2 "Meta" Information Associated with Bonds: Contextual Truth 71
3.4.2.1 The Properties "TRUTH" and "STRENGTH"
** 3.4.3 Preserving Connectivity in Inference Space: "REASONS" and "OFFSPRING" 73
3.4.3.1 Reasons and Offspring vs. Conceptual Causality
3.4.4 Deviations from the "Pure" Representation 74

** 3.5 Summary of Memory Data Structures 76

* 3.6 Representing and Storing Time 76

3.6.1 Time Tokens and Relations: the Requirements 78
3.6.1.1 Relations Between Event Structures and Time Tokens
3.6.1.2 Relations Between Time Tokens
3.6.2 Deictic Time References 82
3.6.3 Fuzzy Duration Concepts 82
3.6.4 Time Example 83

3.7 Comments about the Memory 83

3.7.1 Partitioning the Memory 84

** 3.8 How It All Hangs Together: An Example of an Internalized Conceptualization 85

## Chapter 4. Getting Conceptual Graphs Into the Memory: Reference, Word Sense Promotion, Internalization 96

** 4.1 Referencing Concepts and Tokens from Language: Descriptive Sets 96

4.1.1 Multiple Occurrences of a Concept in a Graph 98
4.1.2 When Referent Identification Is Performed 99

4.2 The Reference Mechanism: Searching for Referents of Descriptive Sets 100

* 4.2.1 Ordering the Descriptive Set 101
* 4.2.2 The Intersection Search 102
4.2.3 Additional Heuristics -- And Problems 103
** 4.2.4 Handling Unidentified References 104
4.2.4.1 When Several Candidates Are Located
4.2.5 Reference Signals and the Special Predicate "REF" 107
4.2.5.1 REF *A*
4.2.5.2 REF *THE*
4.2.5.3 Null REF
4.2.5.4 A Summary of Reference Signals
4.2.6 A Special Reference Heuristic Involving REF *THE* Signals 111

** 4.3 Modeling Immediate Memory: Implicit Word and Concept Activation 113

4.3.1 Activating Implicitly-referenced Concepts 113
4.3.1.1 Free Association Among Words
4.3.1.2 Association Among Concepts Through Conceptual Structures
4.3.1.3 Association Among Concepts Through Inference Structures
4.3.1.4 Examples
4.3.2 Two Memory Tasks Related to Implicit Concept Activation and the Analyzer 118
4.3.2.1 Implicit Concept Activation and Word Sense Promotion
4.3.2.2 Relation Pathfinding: Another Source of Implicit Concept Activation

4.4 Subpropositions 128

4.4.1 Illustrations 129
4.4.1.1 Explicit-peripheral Information
4.4.1.2 Implicit Information
* 4.4.2 Sources 130
4.4.2.1 Explicit Focused Subpropositions
4.4.2.2 Implicit Subpropositions
4.4.2.3 Explicit Peripheral Subpropositions
4.4.3 Conceptual Adverbs 134

* 4.5 Storing the New Conceptual Graphs in Memory Structures 135

4.5.1 Concerning the Referential Identity of Actions and States 136
4.5.2 The Example 136
4.5.3 Linking In the Time Relations 141

* 4.6 A Summary and a Preview 142

## Chapter 5. Conceptual Inferencing: A Subconscious Stratum of Cognition 144

** 5.1 The Spontaneous Substratum 144

5.1.1 Conceptual Inference: The Expansion Force 146
5.1.2 A Brief Illustration 146
5.1.3 About the Psychology of It All 147
5.1.4 The Flywheel Effect 148
5.1.5 Two Other Approaches to Understanding, Briefly 149

** 5.2 What Is a Conceptual Inference? 151

5.2.1 "Conceptual Inferences" vs. "Logical Deductions" 152
5.2.2 The Conceptual Inference Evaluation Process: A Preview 154

** 5.3 The Mainstream Conceptual Inferences 155

5.3.1 An Important Caveat 156

* 5.4 Specification Inferences: Predicting and Filling in Missing Conceptual Information 157

5.4.1 Why Do It? 158
5.4.2 Detection and Marking in the Conceptual Analyzer 158
5.4.3 The Specification Process 160
5.4.3.1 Detecting Missing Information in the Inference Monitor
5.4.4 Specifier Molecules 162
5.4.4.1 Applying Specifier Molecules to Memory Structures
5.4.4.2 Inside the Specifier Molecule
5.4.5 Merging the New and Old Entities 165
5.4.5.1 The Predicate "IDENTIFIES"
5.4.6 Specifier Molecule Example 167
5.4.7 Specification-reference Interaction 168
5.4.8 Other Examples: Typical Sources of Missing Specification 169
5.4.9 A Summary of Specification Inferences' Utility and Operation 170

** 5.5 Causality 176

5.5.1 Causality Communicated By Language and Causal Chain Expansion 177
5.5.1.1 "Should", "Ought to", Etc. and Causal Chain Expansion
5.5.2 Resultative and Causative Inferences 181
5.5.2.1 Resultative Inferences
5.5.2.2 Causative/Resultative Inferences, "CAUSE" and "REASONS"
5.5.2.3 Causative Inferences
5.5.2.4 Make Them All!
5.5.3 Language-communicated Cancause Relations 186
5.5.4 Implementing Causal Chain Expansion 188

5.5.5 Another Task Related to Language Use of Causality 190
5.5.6 Preserving Causal Connectivity 191

* 5.6 Motivational Inferences: Actions and Intention 201

5.6.1 Resultative Inferences and Motivation 202
5.6.2 Modeling the Actor's Knowledge 203
5.6.3 Implementation of Motivational Inferences 206
5.6.4 Problems With Intentionality:
"Peculiar" Motivational Inferences 207
5.6.4.1 Detecting the Non-intentionality of the Original Action Itself
5.6.5 Motivational Inferences and Future Actions 211

** 5.7 Enabling Inferences 215

5.7.1 Intrinsic and Extrinsic Enabling States 216
5.7.2 Arguments for Generating
Many Enabling Inferences Spontaneously 219

Chapter 6. More Conceptual Inferences 223

* 6.1 Function Inferences 223

6.1.1 Triggering Information 224
6.1.2 Normal Function Inferences 225
6.1.2.1 The Normal Functions of Objects
6.1.2.2 Generating a Function Inference
6.1.2.3 Contribution to Understanding
6.1.3 Overriding Normal Function Inferences 232
6.1.3.1 Overriding Function Inferences by Specific Knowledge
6.1.3.2 The Influence of Action Predictions on Function Inferences

* 6.2 Enablement Prediction Inferences 241

6.2.1 Why and How 244
6.2.2 Examples of Utility 245

6.3 Missing Enablement Inferences 248

6.4 Intervention Inferences 249

6.4.1 The Triggering Pattern 250
6.4.2 The Substance of an Intervention Inference 252
6.4.3 Examples 254

** 6.5 Action Prediction Inferences:
Applying Algorithmic Knowledge to Understanding 255

6.5.1 Overview of the Action Prediction Process 257
6.5.2 An Action Prediction Example 258
6.5.3 Seeking an Action on the Part of the Wanter 262
6.5.4 Predicting the Desire for Enabling States 263
6.5.5 Action Prediction Inferences' Utility 264
6.5.6 An Inadequacy 265

6.5.7 Enlistment Prediction 266

** 6.6 Knowledge-propagation Inferences 267

6.6.1 Generating Knowledge Propagation Inferences 269
6.6.1.1 Modeling the Knower's Knowledge
6.6.2 Avoiding the "He Knows That He Knows That ..." Problem 273

** 6.7 Normative Inferences 273

6.7.1 Normality Molecules 276
6.7.1.1 Assessing a Structure's Compatibility
6.7.2 Inside an N-molecule 278
6.7.2.1 Supplying the Reasons for the Assessment
6.7.3 Where N-molecules Are Useful 282
6.7.3.1 How N-molecules Massage Fuzzy Matches
6.7.4 Suggestions for Research 283

* 6.8 State-duration Inferences and the Frame Problem 284

6.8.1 Possible Approaches 285
6.8.2 Normal Durations 285
6.8.3 Mapping Fuzzy Durations Onto Compatibilities 288
6.8.4 The Complete Process 289

6.9 Feature and Situation Inferences 290

6.9.1 Feature Inferences 291
6.9.2 Situation Inferences 294

6.10 Utterance Intention Inferences 295

6.10.1 Other Examples of U-intent Inferences 297

** 6.11 Some Thoughts About Comprehensiveness 302

** 6.12 Some Thoughts About Practicalities 303

Chapter 7. The Inference Control Structure, the Structure Merger, and Other Aspects of the Program 305

7.1 Implementing the Inference Capability 305

** 7.1.1 Data vs. Program vs. Data vs. Program vs. ... 305

7.2 The Inference Control Structure 308

7.2.1 The Basic Monitor 308
7.2.1.1 The Queues
7.2.1.2 Applying Inference Molecules to Each Structure on the Queue
7.2.1.3 The Structure Generator
7.2.1.4 Evaluation and Reordering
7.2.1.5 Summary of the Basic Monitor
7.2.2 The Inference Postscanner 314
** 7.2.3 Relaxing the Inference Network 315
** 7.2.4 Inference-reference-inference Interaction 316

* 7.3 Inference Molecules 318

7.3.1 External Organization 319
7.3.2 Internal Structure 320
7.3.2.1 Inference Atoms
7.3.2.2 Common Low-level Pattern Matching Tests
7.3.3 An Inference Molecule Example 325
7.3.4 Multiple Inference Passes: Smart Inference Atoms 329
7.3.4.1 Concerning the Referenceability of Inference Atoms

7.4 Multiplexing Inferences by Theoretical Types 332

7.4.1 Why Multiplexing Is Necessary 333
7.4.2 The Multiplexor 334
7.4.3 Avoiding Inference Backflow in the Network 335

7.5 Recognizing Points of Contact: The Inference Evaluator 337

** 7.5.1 Possible Interactions: Confirmation, Contradiction, Augmentation 337
7.5.2 Reactions to Comfirmation, Contradiction and Augmentation: Intuitively 338
7.5.2.1 Direct Confirmations
7.5.2.2 Direct Contradictions
7.5.2.3 "Normality" of the New Structure
7.5.2.4 Augmentation
* 7.5.3 Detecting Confirmations and Contradictions 341
7.5.3.1 The First Problem: Compatibility of Occurrence Sets
7.5.3.2 Detecting Confirmations: The Problems of Matching Bonds
7.5.3.3 Detecting Contradictory Bonds
7.5.4 N-molecules and the Evaluation Process 347
7.5.4.1 Evaluator/N-molecule Communication
7.5.5 The Evaluation Sequence in the Program 349
7.5.6 Problems, Problems, Problems 352
** 7.5.7 Inference Cutoff: Is It a Real Issue? 352

* 7.6 Merging Information Structures: The Structure Merger 353

7.6.1 The Merge Sequence 354

** 7.7 A Swipe At the Notion of "Backup": An Editorial 358

7.8 The Memory as a Conversationalist 359

7.9 Reorganizing Things a Bit 360

** Chapter 8. Inferences Applied to Reference Establishment and Time Relations 363

8.1 Inference and Reference Establishment 363

8.1.1 An Illustration 364

8.1.2 Additional Merge Processing at Referent Identification: Identity Merge 370
8.1.3 An Extension to the Reference Mechanism 372

8.2 Time and Inference 381

** Chapter 9. Conclusions, Future Work 384

9.1 Nutshell 384

9.2 Specific Conclusions 384

9.3 General Conclusions 387

9.4 Disappointment 388

9.5 Implications for Psychology and AI 388

9.6 Future Questions 389

References 390

# TABLE OF FIGURES


## Chapter 1

1-1 The inevitable block diagram. 13

## Chapter 2

2-1 The conceptual cases. 28

## Chapter 3

3-1 56
3-2 Separating typeless links from substantive links. 57
3-3 58
3-4 The relationship of a superatom, SA, to its occurrence set, OS, and to its conceptual features. 61
3-5 Part of the occurrence set for some John Smith the memory might know. 61
3-6 64
3-7 64
3-8 XFORM templates. 68
3-9 70
3-10 "John had a book." 79
3-11 "John had a book", internally. 80
3-12 81
3-13 83
3-14 86

## Chapter 4

4-1 Descriptive set for "The big red dog who ate the bird". 98
4-2 Syntax of a descriptive set. 98
4-3 "EQ" LISP pointers to identical references within a graph. 99
4-4 A dumb reference mechanism. 101
4-5 The same dumb referencer, working with a reordered descriptive set. 102
4-6 105
4-7 Deferring reference decisions. 107
4-8 A REF *A* signaled descriptive set. 108
4-9 The book about whales which John gave Mary. 109
4-10 Mapping inferences back into proto-sentences, activating many word senses. 122
4-11 Underlying conceptual relations referenced by concept pairs in language. 125
4-12 Relation pathfinding by expanding spheres through conceptual information. 127
4-13 Implicit subpropositions. 132
4-14 133
4-15 The LISP form in which the memory receives conceptual graphs. 135
4-16 An analyzed graph example. 135
4-17 The time tokens and their relations for the graph of Fig. 4-16. 141
4-18 From utterance to conceptual inferencing to response. 143

## Chapter 5

5-1 Missing information in the utterance "John hit Bill". 159
5-2 Subpropositions in "John hit Bill". 160
5-3 The process of detecting missing information. 161

5-4 The specification request vector. 163
5-5 165
5-6 The IDENTIFIES structure which stores the REASONS for the identification. 166
5-7 A very simple specifier molecule. 168
5-8 Representable causal forms. 177
5-9 178
5-10 179
5-11 180
5-12 Causality in the memory. 181
5-13 Two very simple resultative inferences. 182
5-14 The relations among causative/resultative inferences, REASONS and CAUSE. 184
5-15 Having a catcher's mitt would make Mary happy. 187
5-16 190
5-17 191
5-18 Inferences which are based on causal relations. 193
5-19 How REASONS, CAUSE, resultative and causative inferences are related. 194
5-20 Mary kissed John becaues he hit Bill. 195
5-21 One explanation of why Mary's kissing was related to John's hitting. 196
5-22 201
5-23 A first approximation: candidates for motivational inferences. 204
5-24 Where predictions about the actor's knowledge fit. 205
5-25 The generation of motivational inferences. 207
5-26 Retracing tentative assumptions. 210
5-27 How motivational inferences lead to interesting things. 212
5-28 Actions cause changes in states. States enable actions. 217

## Chapter 6

6-1 Four common NFCT patterns. 226
6-2 Searching up ISA sets for an NFCT property. 227
6-3 The memory structure which stores the normal function of printed matter. 228
6-4 The instantiated NFCT pattern about reading. 230
6-5 The complete function inference and its surroundings. 231
6-6 Discovering unusual relationships between two tokens or concepts. 234
6-7 The process of enablement prediction. 243
6-8 Why Andy might be blowing on the meat. 244
6-9 John asked Mary where Fred was. 245
6-10 246
6-11 246
6-12 The circumstances surrounding an intervention inference. 250
6-13 The triggering pattern for intervention inferences. 251
6-14 Locating the culprit action. 253
6-15 Summary of the intervention process. 255
6-16 What the action prediction inference process tries to do. 258
6-17 260
6-18 261
6-19 269
6-20 The offspring set. 270
6-21 The knowledge propagation inference process. 272
6-22 Precise testing to arrive at a fuzzy compatibility. 277
6-23 How we might go about deciding whether person P owns an X. 278
6-24 286
6-25 A typical STRENGTH function for fuzzy duration #ORDERHOUR. 288
6-26 The format of a fuzzy duration concept's step function. 289
6-27 The process of making a state duration inference. 290
6-28 291
6-29 "Andy's diaper is wet." 292

6-30 292
6-31 An underlying causal communicated conceptually by a REL link. 298

## Chapter 7

7-1 "Passive" vs. "active" pattern matching. 306
7-2 A typical starting inference queue. 309
7-3 The queues which collect the breadth-first expanded inferences. 309
7-4 Relationships between the queue, the structure, the predicate, and the inference molecule. 310
7-5 The inference monitor. 313
7-6 "Bad timing." 315
7-7 Multiple reference-inference interaction passes. 318
7-8 The logical internal organization of a typical inference molecule. 321
7-9 The structure of an inference atom. 322
7-10 An inference molecule used by the current program. 326
7-11 Recording the successful application of an inference atom to a memory structure. 330
7-12 Making program-based conceptual rules of inference referenceable as data structures. 332
7-13 Multiplexing inferences by theoretical type. 335
7-14 Inference backflow. 336
7-15 When are two occurrence sets compatible with each other? 342
7-16 Typically, time relations in the memory are sparse. 343
7-17 The external logic of an N-molecule. 349
7-18 Actions of the evaluator, based on how the new inference relates to other knowledge. 351
7-19 Merging two memory structures. 355
7-20 Plans for a more two-level inference organization. 361

# CHAPTER 1

## INTRODUCTION

### 1.1 ESSENCE

This thesis describes a computer program which exhibits a primitive capacity to think.

The basic unit of input to the program is the *conceptual graph*. A conceptual graph is a cluster of computer symbols linked together in structured patterns to represent the thoughts underlying natural language sentences. The natures of the symbols and connecting links allow the graph to capture the underlying meaning of the sentence in a way which is not dependent upon the way the thought was phrased in language, or even upon *which language* was used to communicate it. The program is therefore designed to function in a pure meaning environment. It assumes the existence of two other programs: a conceptual analyzer [R2], which can transform sentences of a language into these language-free conceptual patterns, and a conceptual generator [G1], which can transform conceptual patterns into sentences of a language. Both these programs interact with the memory during their tasks. Although both companion programs must deal with the specifics of a particular language, all inter-program communication occurs through meaning patterns.

The program has one central reflex response, *conceptual inference*, which is activated by incoming conceptual patterns. From each pattern, this reflex generates many new, meaning-related patterns which represent predictions about facets of the larger situation of which the input pattern might have been a part. That is, the program assumes that what it perceives is *always* only a very small part of a much larger pattern, and it is motivated to discover as much of the larger pattern as possible, and to *relate* what it discovers to other patterns it already knows. To determine points at which one pattern joins with another pattern is its single most important goal.

As subgoals of this task, the program tries to determine why actions were performed, what an action might have caused, what must have been true in order for the action to have occurred in the first place. If a person is in state X, what might he desire as a result of being in that state?

If a person desires Y, what might he do to achieve Y? The program will make predictions about what is likely to happen next, and can realize when its predictions match subsequent incoming patterns. It makes assumptions about what other people know, based on what it already knows they know. It can detect when one pattern of meaning conflicts with another one, and it can combine similar patterns which have come from two different sources. By combining them, it opens new pathways between other information patterns. If there are gaps in the incoming meaning patterns, it tries to fill them in. Based on the larger patterns in which they occur, it can make decisions about who and what the smaller patterns are referencing, even though these things might be undecidable from examining the smaller patterns separately.

The program can get along with less than perfect data. When it cannot locate information it needs, it can make assumptions about that information, based on patterns of what is normal and expectable. It can guess how long certain states and actions in the world last, and use those guesses in its predictions. It is sensitive to time factors in all patterns.

Taken all together, the processes in the program define a theory of understanding which is related to language, yet independent from it. This theory will be called Conceptual Memory.

## 1.2 CHALLENGE

In recent years, the stored-program computer has posed some of the greatest challenges ever to man's ingenuity to synthesize and analyze. One such challenge is to discover a starting combination of ones and zeroes in a computer's memory, and a set of stored programs which manipulate them, which will allow the computer to use and comprehend natural language in the same way humans do.

I have therefore posed the following general question as the starting point for this research:

> **What does the brain of a human language user do with the information communicated to it via natural language? How can a computer be made to do the same things?**

## 1.3 PERSPECTIVE

Most of us take for granted our ability to use and understand the language of our culture. Because of this, it is quite natural to assume that language comes equally readily to computers. This assumption was eagerly made by natural language researchers in the early 50's. But their enthusiasm was quickly dampened: their efforts were stymied on two fronts.

First, computer hardware was in its infancy. It was so new, slow and unpredictable in fact, that even if someone had discovered some computationally effective general principle of language and intelligence, he might never have had the opportunity to confirm it! Also, there was a certain diverting fascination with getting *anything* to work on the new equipment. This instilled a kind of euphoria known only to those who have written a computer program and watched it automatically carry out their own thoughts right in front of their eyes. Perhaps because of this it was thought that the main step had been taken in just getting the computer to do *something* -- that the rest would follow easily. That proved to be incorrect.

Second, even though the field of Linguistics had been around for quite some time, when researchers attempted to encode language in the ones and zeroes of the hulking watt-eaters, a new crop of problems -- a new level of unanticipated detail and intricacy -- arose. For the first time, *everything* had to be made *totally explicit.* Whereas corners could easily be cut in a "paper" theory of language in order to get at some of the deeper issues, in a computationally effective theory, the burden of proof ultimately rests upon the computer's performance. Any cut corners were reflected, to the chagrin of many a researcher, as direct idiocy in his mechanical prodigy. And it was more than engineering details; it was an absence of theory. In short, it was quickly discovered how little was actually known about language.

At the crest of this first wave of excitement was the vision of automatic machine translation of one language into another. It initially looked as though translation could be achieved by extremely simple and local transformations on words. Because of this, most of the main issues which came into focus were related to the maintenance and use of dictionaries in the computer: storing words and word senses, organizing large vocabularies, devising faster and faster lookup techniques, cross-referencing entries (synonymy, antonymy), and so on. Just off the main stream of automatic translation were endeavors in word frequency analysis, automatic keyword

compilation, and the like. It was the era of a new form of computation: symbolic transformation and manipulation of natural language vocabulary words.

Notably missing in these first efforts were any serious attempts to revamp and incorporate traditional ideas about grammar and syntax. Then in the 60's, perhaps revitalized by Noam Chomsky's new approach to syntax which appeared in 1957, the field experienced a rush on syntax. It was the next logical step to take, and interest in it was all the more heightened and sustained by the emerging need for more sophisticated *artificial* (programming) languages. The new issues became those of how to represent a grammar as precise syntactic structures within a computer, how to make them flexible and extendable, and above all, how to use them to analyze (parse) sentences of the language into syntactic structures. This latter issue gave rise to innumerable theories of syntax, and to theories of how best to parse. Better and better syntactic analyzers were written, and a precious wealth of discovery was made. But computers still could not understand language, even though they could now babble prolifically in meaningless -- but grammatically impeccable -- sentences, and could chastise with flashing lights and ringing bells all those who spoke to them ungrammatically. Basically, researchers had beaten the dead horse, and he still would not rise.

Relative to the goal of getting a computer to understand natural language, the shortcomings of syntactic parsing were threefold. First, it was far too precise: although it would work in the laboratory on carefully selected sentences, the smallest deviation (something a human would scarcely notice) from what the grammar prescribed would invariably cause the system to fail. Second -- and it is surprising that so many researchers deluded themselves so long on this -- correct syntactic analysis is inseparable from the individual meanings of *each word* in the language. To classify one word as a noun, another as a verb, indeed led to a parse capability, but the parse which resulted was at best just an analysis of form rather than meaning. At its worst, the parse was even an incorrect analysis of form, because of "peculiar meanings" of certain words. Third, and most important, even if a correct syntactic analysis could be *guaranteed*, what was it good for, relative to understanding? It is not at all taxing to find very ordinary sentences whose syntactic *form* is not of much use in predicting their meaning:

**John's refrigerator was running.**
**John's horse was running.**
**John's candidate was running.**
**John's nylon shirt was running.**
**John's specialty was running.**
**John's nose was running.**

(concentrate on the picture each elicits in each case). Because of this, and because the goal of syntactic parsing was to render an analysis of *form*, understandably little thought was given to this question of the utility of syntactic analysis to understanding. It finally came to be generally accepted that, no matter how clever one was, syntax was simply the wrong way to begin an understanding system.

Researchers had at that point come to grips with one of the key realizations about language: that syntax and meaning are thoroughly intertwined, and that syntax -- regardless of how elaborate a role it plays -- should serve only as a means to an end: to discover the underlying *meaning* of each sentence. This realization marked the beginning of the third generation of language researchers, the "good guys". The new issues became how to *represent meaning* (Schank [S4] was among the first good guys here), and how to be diplomatic and charitable in the merger of syntax and semantics in the programs designed to extract meaning from sentences (Winograd [W5] was among the first here). At last, the veneer of pure syntax was being sanded away, and the underlying issues of language and cognition were beginning to be recognized as one and the same.

In 1971 Terry Winograd's *Procedures as a Representation for Data in a Computer Program for Understanding Natural Language* served to coalesce and further this undercurrent which had been around for several years prior to his program. Winograd showed how syntax, semantics, and a model of the world all fit together in a way which permitted a computer to converse with a human in a limited domain, and to perform simple manipulations of the world model. The system exhibited a noteworthy use of information *from all three levels* -- syntax, meaning, world model -- in its task.

The broad significance of Winograd's program was in the way it *arrived at* the underlying meaning of each sentence. It was chiefly a theory of how knowledge from several independent sources can be applied to predict *meaning* from *form*. Less attention was paid to the problem of

*what to do with the meaning* once it had been obtained. (In particular, the goals were to answer questions about the model, to manipulate the model, and to explain and justify its actions in this regard). In this theory of Conceptual Memory, the emphasis is reversed: I have been less interested in *how* the underlying meaning of each sentence is extracted, and more interested in the problem of what to do with the meaning after it has been extracted. I am interested in the effects of each sentence's underlying meaning within a memory: how the information in each sentence logically flows through various cognitive processes, and how it interacts with the meanings of other sentences and with a model of the world. In short, how does the content of language utterances interface with our ability to think?

## 1.4 GENERAL GOALS

What *does* a person do with the information content of natural language, anyway? -- what does it mean to understand, beyond the stage of syntactic or even meaning analysis of sentences? These are tough questions -- things we cannot answer by direct analysis of our brains because they concern abstractions whose relation to the brain's physical properties are extremely complex. One of the goals has been just to identify some of the questions!

We can all explain the how and why of our ability to comprehend language on a case-by-case basis: "Oh yes, I understood that because I knew that ...", or "You must be talking about John, because...". Because of this, everything I will discuss is "what everyone already knows anyway" -- to study language is to study *everything*, because everything can be described and assigned meaning by language; it is the most powerful means of representing knowledge that exists. Language and knowledge simply cannot be separated. In this sense, any theory developed will address issues which are second nature to us all. Unlike a theory of high-energy physics, it will be a "theory of the familiar."

But I am not interested per se in the case-by-case analyses at which we are all so facile. Rather, the real challenge lies in discovering -- either by synthesis of an artificial system, or by analysis of a natural one -- the *underlying logical (as opposed to physical) organization* which accounts for in this case-by-case ability to comprehend. The moment one makes a conjecture about the nature of the underlying organization, the character of the theory abruptly changes from familiar to esoteric: although we are *certain* of why we concluded X in situation Y, we can

only guess at the general mechanism in our brain which underlied the ability to conclude X in situation Y, W in a similar situation Z, and so on. It is both difficult to discover these general mechanisms by introspection, and difficult to comprehend their scope once they have been discovered.

The general goal of this theory is to make many guesses about underlying higher-level logical functions of the brain, to synthesize them into a unified theory of understanding, then to implement them in computationally effective algorithms which can be carried out by a computer program.

> **The goal is to develop a computationally effective model of the logical flow of information in the brain of a natural language user. This model should predict and explain the ways in which information communicated to him by language is dissected, transformed, rearranged, extended, and recombined in novel patterns which are influenced by the situation in which he perceives that information.**

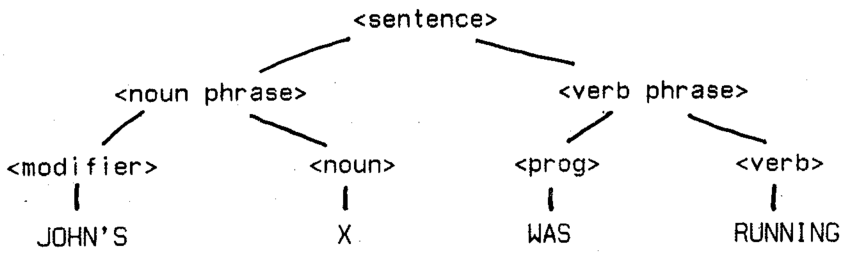
I will not be concerned so much with a model of the physics of the brain -- neurons, charges, electrical wavefronts, and the like, or with a model of the physical organization of the brain -- short term memory, long term memory, engrams, recall, forgetting, and the like -- as with the abstract flow of information and with the information structures which must exist to realize aspects of the processing.

## 1.5 TENETS

I want to ask questions about some of the deeper cognitive processes in humans. One tenet is that these processes are independent from language and culture. It is of course important to distinguish the *processes* from the *data* the processes manipulate. The data will obviously be highly language and culture-specific.

In order to deal at this language and culture-free level, (a) there must exist an effective method of representing information in a form which is language- and culture-free, and (b) care must be taken that the processes defined and synthesized are truly language- and culture-free: any process must be able to function entirely in this pure meaning environment which is buffered from language form, and independent of specific knowledge.

To illustrate what it means to buffer the memory processes from the *form* of the language from which they derive their data, return to the examples of "running". All of these have the same *syntactic* form, which might be analyzed as follows in a phrase-structure grammar:




However, each has a thoroughly different underlying meaning -- each produces a completely different picture when we hear it. To capture the meaning is, in a sense, to represent the *structure of the picture.* If we view language this way, these sentences about running come out as follows:

**John's refrigerator was running.** an appliance was functioning normally

**John's horse was running.** an animal was propelling itself rapidly by moving its legs

**John's candidate was running.** another person who lies in an unspecified relation to John was performing actions intended to result in his attaining some office

**John's nylon shirt was running.** an article of clothing was falling apart

**John's specialty was running.** John's ability to propel himself rapidly by moving his legs was more highly developed than most of his other athletic abilities

**John's nose was running.** fluid was being unintentionally expelled from a body organ

And not only can similar language forms convey completely different meanings, completely different forms can communicate similar meanings. Instead of saying John's specialty was running, we might say "John was a specialist at running." The comprehender gets the same message from both. And this phenomenon is limited only by our ability to obfuscate and distort the issue in tedious language forms:

1. Indeed, the quantity of faith held by myself in the structural integrity of my motorcycle is as abundant as the number of "I"s in that honorable southern state in which my great Aunt Jessica was conceived.

2. I trust my bike.

So I *assume* that what the memory receives as input is as independent from language as the state of the art of language analysis will allow. There will of course be much interaction between a good analyzer and the memory which I will not cover. These assumptions afford a starting point from which to examine some language-related cognitive processes independently of any particular language. We will also examine, to a lesser extent, how these processes relate (in language-free ways) back to the processes which perform meaning analyses of sentences in a language.

## 1.6 SPECIFIC GOALS

The use of language presupposes that both speaker and comprehender have access to roughly the same storehouse of knowledge. There must be some common frame of reference. Because of this, *no* language utterance is ever any more than a very lean allusion to the very rich situation it describes. My specific goal has been to identify how a person who hears a lean utterance expands it in his mind into the rich underlying circumstances surrounding it, and then how he discovers how aspects in this expanded situation relate to aspects of other expanded situations. To discover these interrelationships between the situation described by one utterance and the situation described by another utterance will serve as my general definition of comprehension.

The specific goal is to identify classes of *conceptual inferences* which contribute to this automatic expansion. Sub-goals are to define memory structures which are the medium for these expansions, and to determine how information gets into these structures so that expansion can occur. Also, I want to examine how memory structures knit together when regions of one larger pattern abut with those of another.

To do these things, the memory will be making guesses about things of which it isn't certain, modeling other people's knowledge, making predictions about people's motivations and possible future actions, guessing how long certain situations in the world last, imagining what must have been true for someone to perform an action he is said to have performed, making guesses about missing information, inferring what caused what and why, predicting who is being talked about if it could be more than one person, and so on.

In addition to conceptual inferences and their control structure in the program, problems of reference, and an important reference-inference relaxation cycle will be identified and solved.

### 1.6.1 THEMES

The following unordered list of themes is presented here as a montage of what is to come. It is intended only to communicate the general flavor of the research by keywords.

**Inference molecules and spontaneous expansion in inference space**

**The inference evaluator, confirmation, contradiction, and structure merging**

**Knowledge propagation inferences, motivational inferences, action prediction inferences**

**Occurrence sets, conceptual bonds, reasons and offspring**

**Implicit concept and token activation, word sense promotions**

**The Conceptual Dependency representation formalism**

**Internalization, identification and extraction of subpropositions for inference**

**Causative and resultative inferences, and causal chain expansion**

**Descriptive sets and identification of referents**

**Enablement inferences, function inferences, intervention inferences and enablement prediction inferences**

**Time atoms, fuzzy durations, state duration inferences, and time maintenance**

**Assumptions about normality in the world, normality molecules and normative inferences**

**Inference multiplexing by theoretical type**

**Feature inferences, situation inferences, utterance-intention inferences**

**Reference-inference relaxation processing**

**Specifier molecules, and the filling-in of missing conceptual information**

## 1.7 THE PROGRAM AND SOME ASSUMPTIONS IT MAKES

The program which implements this theory of Conceptual Memory is called MEMORY. It is written in the programming language MLISP, which was developed by David Smith [S13] at the Stanford Artificial Intelligence Laboratory for the PDP10 computer. The program occupies approximately 50,000 36-bit computer words when it is run with its starting data file of world knowledge (approximately 300 memory structures and 50-60 program modules which contain specific world knowledge in program form). It assumes the existence of two other programs which implement a theory of conceptual analysis and a theory of conceptual generation. The three programs, when run together, occupy approximately 90,000 36-bit computer words.

### 1.7.1 THE CONCEPTUAL ANALYZER, BRIEFLY

The conceptual analyzer (for English) was designed and is under current development by Chris Riesbeck [R2]. It relies as much as possible upon meaning. Syntax, where essential, is incorporated in the same *feature / request* control structure as all other information about words and their meanings. Because of this emphasis on meaning, the input language string need not be syntactically well-formed; the only requirement is that it be *conceptually* meaningful.

"Feature / request" means the following: each new word which is encountered in the analyzer's left-right scan of the utterance is treated as a unit of meaning which exerts an influence in two ways. First, it can contribute its conceptual *features* to a queue. Second, it can cause *requests* -- skeleton conceptual graphs which underlie the word -- to be set up. Requests represent the active, goal-directed processes which attempt to combine the features on the queue. At any given time, each unfilled slot in a request constitutes a goal to be satisfied. In this sense, the analysis can be called top-down. But since the requests are initiated in the first place by the words of the sentence, the process can also be called bottom-up.

The dictionary entries for word senses which reference simple concepts, like John, cake, bicycle are simple sets of conceptual features which characterize the concept. For words which are underlied by entire complex structures (most verbs, for instance), the dictionary entries are the skeleton conceptual templates which become the requests during the parse. The dictionary currently consists of 300-400 words, of which perhaps 100 are verbs.

In chapter 3, we will see the computer form of a conceptual graph which is the output from the analyzer, and the input to MEMORY.

### 1.7.2 THE CONCEPTUAL GENERATOR, BRIEFLY

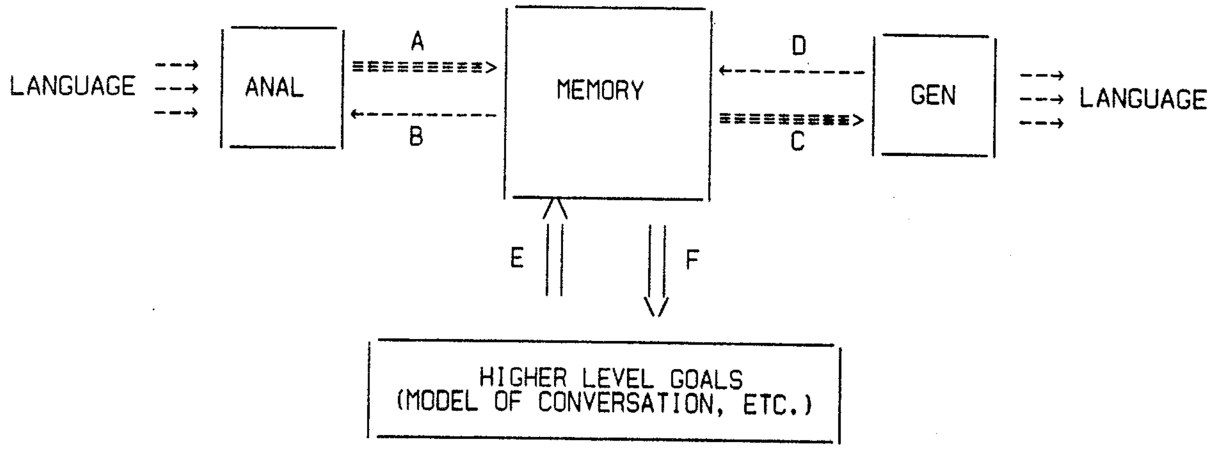
The conceptual generator (also for English) was designed and is under current development by Neil Goldman [G1]. It is logically a two-step process. The first step carries a deep conceptual graph into a semantic network of English words. The second step carries this network into a grammatical English string by means of an Augmented Finite State Transition Network approach described by Simmons [S11]. The program which implements the second step is in fact an adaptation and extension of Simmon's program.

In order to construct the net, the generator examines the conceptual graph's general structure, and on this basis selects one of 20-30 binary discrimination nets. The conceptual graph is then filtered through this net, which performs tests, lying in three general categories, on the graph's structure and contents. Tests in the first category inquire about the identity and conceptual features of objects in the graph. Tests in the second category ask whether an entire substructure could be expressed by some particular language construction. The third category involves general queries to the memory to ascertain time relations, and the existence of particular contexts which would allow the generator to select more compact or appropriate words than would be possible outside that context.

At the terminal nodes of the network are lexical verb senses with which are associated *case frameworks*. The filtering process therefore serves to select the central verb for the main graph, and nested subgraphs. A case framework specifies what cases are required for the verb, where they may be found in the conceptual graph, and what the correspondence between the verb's conceptual and syntactic cases is. This correspondence is then used to construct the semantic net.

### 1.7.3 WHERE THE MEMORY FITS, ALL THINGS CONSIDERED

The following block diagram is intended to help put the memory in perspective, as a component of a larger picture.




**A -- analyzed conceptual graphs which the analyzer "hears"**

**B -- answers to analyzer-initiated questions such as:**

"Who is this John I'm hearing about likely to be,
and does he have an unusual occupation?"
"Clarify the relationship underlying 'John's yard'"
"Is there an animate concept which the word 'dog' could reference?"
"What is the most likely meaning of 'bank' in the current context?"

**C -- memory structures to be expressed in language**

**D -- answers to generator-initiated questions such as:**

"Was time C0082 before time C1178?"
"Could John's doing X cause Mary harm?",

**E -- goal-specific directions and queries to memory such as:**

"We might have a guilt pattern emerging;
start emphasizing John Smith's reasons for acting."
"Could John's saying X to Bill have hurt Bill' feelings?"
"Believe what John says implicitly."

**F -- suggestions and tips about interesting events in the memory such as:**

"What Bill just said to Mary probably hurt her feelings. Want to intervene?"
"John has done several things which might indicate he no longer loves Rita.
Call up a special program to analyze further."
"Bill might be getting ready to go to the store.
Want to ask him to get anything?"
"I've inferred that Mary wants the chair moved; want to respond?"

**Figure 1-1. The inevitable block diagram.**

The nature of the information which flows over paths E and F is merely conjecture.

Furthermore, as chapters 5 and 6 will illustrate, there is *anything but* a well-defined boundary between the memory and "higher level" and goal-oriented processes, such as dialog and bargaining models, or the model which might drive an information-seeking robot. The existence of the "higher-level-goal" box in Fig. 1-1 serves only to emphasize that the processing I will propose is not all-emcompassing, and that any specific application of my theory must be driven by a set of goals which function "on top of" (or, more precisely, "within") the memory.

**We will be exploring the "lower-upper class" of cognition.**

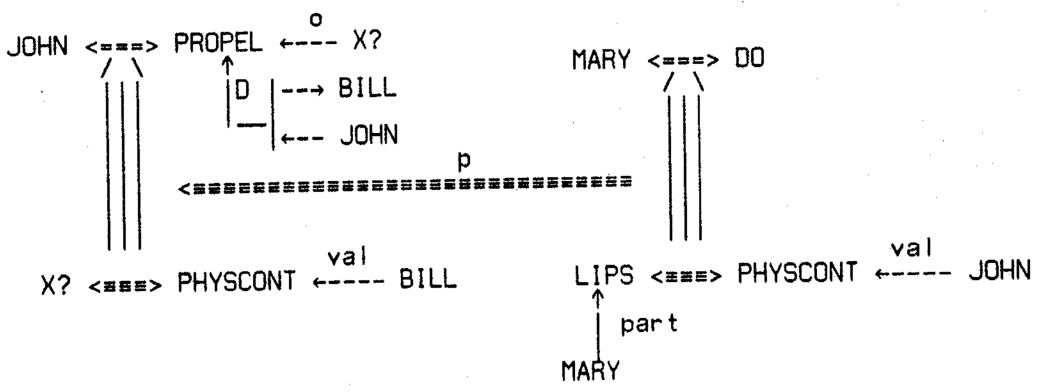
### 1.7.4 OVERVIEW OF THE PROGRAM'S OPERATION

To give you a large-scale idea of things, I will describe here the overall behavior of the program as it runs in response to one utterance.

The language analyzer (Riesbeck [R2]) is requested by MEMORY to "listen" for an utterance (it does not have to be either complete or syntactically correct) to be typed at its keyboard. An example of an utterance is:

(MARY KISSED JOHN BECAUSE HE HIT BILL)

When an utterance is sensed, it is analyzed by the conceptual analyzer into a conceptual graph which is the meaning representation for that utterance (don't analyze it yet, just enjoy):



During and after the construction of the meaning graph, all references in the graph to objects and concepts in the world are established as best as possible by the memory. This process will, for instance, replace the *symbol* "JOHN" with a pointer to an entity in the memory which represents some *particular* John, about whom much may be known. Also, as this referencing occurs, the graph is dissected into its components and these components are linked in to the memory's network of other knowledge. Interesting facets of the graph are identified, and become the starting points for expanding (by making conceptual inferences) the situation to which the utterance alludes.

Next, MEMORY assumes that the thought was communicated for a reason, and that it conveys interesting information which it does not directly contain, and which depends on the context in which the utterance has occurred. It begins generating conceptual inferences *as a routine response* in order to see how the information conveyed by the utterance relates to other knowledge in its memory. This is *spontaneous* -- a reaction to each new input, rather than upon external demand; in a sense, then, the memory generates its own goals.

As each new inference is generated, an evaluation function is applied to it. The evaluator attempts to relate the new inference to existing knowledge in the hope of discovering interesting relations with other information structures in the memory. One important result of this is the merging of two structures into one, thereby establishing a new pathway between previously unrelated information.

After interesting interactions of the new conceptual inferences from each utterance have been discovered, the memory makes numerous responses. This theory does not extend into the domain of deciding what is appropriate to say. MEMORY therefore proposes everything of interest which results from the utterance, forming a list of conceptual graphs to be expressed by the conceptual generator.

The generator (Goldman [G1]) is capable of transforming a meaning graph into natural language utterances in some target language. In this process, MEMORY is consulted often to determine if suitable conceptual information exists to allow the use of a particular word of the language to express some part of the meaning graph, and to assist the generator in constructing the appropriate tense framework from conceptual time structures in the memory.

The memory is therefore not yet a conversationalist: what you get by running the program is a rambling, stream-of-consciousness monologue. The analyzer and generator with which it works both are designed for English.

### 1.7.5 COMPUTATIONAL ISSUES

The current program simulates several dozen rather involved processes. When it is turned loose in full gear, it requires annoying amounts of computer time (5 minutes is typical) for responses to each utterance. This is partly because of blatant programming inefficiencies; but it is mostly because of the theory. Thus, the memory has rarely been run in "all-at-once" mode. Instead, features can be turned on and off for purposes of demonstrating their effectiveness, and how they interrelate. I am convinced that all processes are consistent, cooperative, and coordinatable, even though the all-at-once mode more often than not blows up because of one or more program bugs! To find one can take the better part of a day, and I have not recently had the luxury of such quantities of time.

These things neither disappoint nor discourage me. If they disappoint you, consider what we are trying to model: a highly parallel associative network of billions of active nodes, where *each node itself* might realize a function which would take many seconds to model on today's computers!

If we can identify what is useful, and sketch out how to do it, we can then worry about time, parallelism, and programming oversights. We must pass our theories through finer and finer sieves. While the boulders still intimidate us, we must preserve our sanity by this attitude long enough to get to the sand! (Whew!)

### 1.7.6 EXAMPLES

The following examples are actual computer output, and illustrate th current level of the program's ability. By "actual computer output" I mean that the memory can accept and generate the underlying meaning graphs associated with each example (and others which follow similar patterns). Since the conceptual analyzer and generator were developed independently, they had slightly different vocabularies and abilities. In cases where they were not adequate, I have

doctored the input and output. That is, sometimes I gave the memory a hand-analyzed meaning graph which would normally be constructed from an English utterance by the analyzer, and sometimes I either expressed by hand the meaning graphs which were produced by the memory, or touched up the generator's output.

Also, as will be seen in chapters 5 and 6, the memory will generally produce many more responses than would normally be seen externally if a more sophisticated model of conversation were coordinating things. This abundance of potential responses is a crucial aspect of the theory, but until there is a model of conversation, it means that I have to select the responses for illustration manually.

(SIMPLE EXPANSION BY INFERENCE)

INPUT: John told Mary that Bill wants a book.

RESPONSES: John believes that Bill wants a book.
Mary now knows that Bill wants a book.
Bill wants a book.
Bill wants to come to possess a book.
Bill probably wants to read a book.
Bill might want to know the concepts contained in the book.
A book about what?
Bill might get himself a book.
John might give Bill a book.
Mary might give Bill a book.
John may want Mary to give Bill a book.
John and Mary may have been together recently.

(DIALOGUE)

INPUT: John went to the drugstore.
RESPONSE: What did he want to buy?
INPUT: He wants some aspirin.
RESPONSE: Is he sick?
INPUT: No. His wife is sick.
RESPONSE: I thought John was single. Oh, wait a minute. You must mean John Smith, not John Jones.

(STORY)

INPUT: John hit Mary.
INPUT: Mary's mother took Mary to the hospital.
INPUT: Mary's mother called John's mother.
INPUT: John's mother spanked John.

INPUT: Why did John's mother spank John?
RESPONSE: Because she was angry at him.

She was angry at him because he hit Mary.

INPUT: Why did Mary go to the hospital?
RESPONSE: Because she was hurt.
John hit her.

INPUT: Why did John hit Mary?
RESPONSE: He was probably mad at her because she did something to him.

(REFERENCE-INFERENCE INTERACTION)

(a) INPUT: John was mad at Bill.
INPUT: He kicked him.
INPUT: Who kicked who?

RESPONSE: John kicked Bill.

(b) INPUT: Andy Rieger is a baby.
INPUT: Andy Jones is an adult.
INPUT: Andy's diaper is wet.

INPUT: Whose diaper is wet?
RESPONSE: Andy Rieger's diaper is wet.

(FEATURE, NORMATIVE, AND ENABLING INFERENCES)

(a) INPUT: Phighdough barked at a policeman.

INPUT: What is Phighdough?
RESPONSE: Phighdough is probably a dog.

(b) INPUT: Mary was sunbathing yesterday.

INPUT: Where was Mary yesterday.
RESPONSE: Mary was probably at the beach yesterday.

(c) INPUT: I saw John yesterday.
RESPONSE: I thought John was in France.

(CONTRADICTION DETECTION)

INPUT: John said he killed himself.
RESPONSE: Dead people can't talk.

(INFERENCE USING NORMALITY INFORMATION)

INPUT: Does Mary own a book?
RESPONSE: Probably, people normally own books.

(CONTEXT EFFECTS ON INTERPRETATION)

(a) INPUT: Mary was feeling extremely despondant.
INPUT: John saw she had a knife.
INPUT: He asked her for the knife.

INPUT: Why did John want the knife?
RESPONSE: Probably so Mary can't use it to hurt herself.

(b) INPUT: John wanted to eat a steak.
INPUT: John asked Mary for the knife.

INPUT: Why did John want the knife.
RESPONSE: He probably wants to cut the steak with it.

(c) INPUT: John was furious at Bill.
INPUT: John asked Mary for a knife.

INPUT: Why did John want the knife?
RESPONSE: Perhaps to hurt Bill.

(CAUSAL CHAIN EXPANSION)

INPUT: Bill took Mary's book away from her.
INPUT: Mary kissed John because he hit Bill.

RESPONSE: John hit Bill.
This caused Bill to become hurt.
This caused Mary to be pleased because she felt a negative emotion toward Bill at the time.
Mary's pleasure was caused by John's action.
This caused Mary to feel a positive emotion toward John.
This caused Mary to kiss John.

### 1.7.7 ABOUT THE COMPUTER EXAMPLES

I have tried to illustrate by computer example many of the theoretical points The examples were generated over a period of several months as the program was still developing (as I hope it will continue to develop). For this reason, you may notice differences in trace format, or in small details of the processing from example to example. Also, it more often than not happened that, to illustrate one smaller point, it was necessary to shut off other features of the system which were not relevant to the demonstration, or to edit their traces out of the example after it had been generated. This was the only space-wise practical thing to do, but it makes it hard to absorb the gestalt of the system's operation. I have made efforts not to isolate any one example too severely from the rest of the system.

All examples are unretouched computer trace output, modulo shuffling the output around so that it would fit readably in the left column of the page. I have tried to indicate by ellipses those points at which trace output was edited out, but make no claims as to the thoroughness of this convention. The documentation in the right column was added by hand after the tracing.

## 1.8 WHAT THIS IS: COMMENTS

Part of the task in synthesizing a theory of cognition and language is to *define* problems whose solution will be theoretically meaningful. This is not the case in many other language-related endeavors. For example, the problem of transforming (parsing) sentences of a particular language into some underlying structure is "well-defined" in the sense that at least the general goal can be concisely described. The inverse process of transforming underlying structures into acceptable language strings is a similarly well-defined task. Indeed, there are many ill-defined subgoals in the solution of such problems, and the problems are no less difficult because there is a general goal. Nevertheless, the goal provides a standard by which the relative success of the solution can be measured, and it is fairly straightforward to realize failures and deficiencies, pinpoint their cause, then patch them up or extend the deficient processes.

There is no one identifiable goal for a language-independent model of cognition. What does it mean "to understand", and how do we know when we have a program which does it? How do we know when to be happy and when to be disappointed with our understanding program's behavior. What do we do to make it better when we're disappointed? This thesis is the result of asking these sorts of questions.

I have come to believe that all research involving language and human memory must of necessity lie on the "lunatic fringe" of many established disciplines: linguistics, computer science, cognitive psychology, and philosophy foremost among them. And each issue in a comprehensive theory of language and intelligence will require justification, or at least reconciliation, with the existing, generally insightful dogma of each discipline.

If we look far enough back, there seems to be no aspect of language or intelligence which has not been explored or pondered at some time or another by more capable men than ourselves -- experts in their fields. But their discoveries and insights are only the pieces of the puzzle which must be fit together.

**Today's ideas about language and intelligence must be measured by how well they re-partition, connect and extend things which have been known a long time.**

And they must do so at a very explicit level so that a computer can learn them.

So in this thesis I have tried to identify and coordinate many ideas about language and intelligence *in a computationally effective way.* I do not pretend to be a master of any of the four areas above; I *do* claim to be a craftsman with some fresh ideas about what underlies language, and how to put it into a computer. To cover *any one* of the issues I have chosen to address in the thoroughness any *particular* discipline would demand would subvert my immediate goals. We can sweep up the shop later. Let's build something first!

## 1.9 RELATED WORK

In writing this thesis, my goal was to start out fresh, developing a theory of language understanding as I saw it, and Conceptual Memory is the result. The research here has by no means been conducted in a vacuum. One is influenced in many subtle and not so subtle ways by reading the literature of his field, and this certainly applies to me as much as to anyone. In particular, I am indebted to the following people: Roger Schank [S4,S5], Terry Winograd [W5], Ross Quillian [Q2], Gordon Bower [B4,A5], John Anderson [A5], Ken Colby [C3], David Rumelhart [R4], Peter Lindsay [R3], Don Norman [N5,R3], Robert Abelson [A1], Joe Becker [B1], L. A. Zadeh [Z1], Yorick Wilks [W3], and (to a lesser degree since much of my research occurred concurrently with his), Eugene Charniak [C1]. The works of each of these people stand out in my mind as important influences on my thinking.

However, the issues with which we are all dealing are so broad, and the goals so ill-defined at this stage, that there is little ground for direct comparison of what I have done with what they have done. In my view, this thesis represents a new approach to language processing and understanding. Thus, rather than review individual works, I will assume the reader has a "cultural" knowledge of previous work in the field of language understanding by computer. Those completely unfamiliar with the field are directed to the reference list at the end of the thesis. I believe, however, that most will find the thesis fairly well self-contained.

## 1.10 READING THIS MONSTER

This thesis is longer than I wanted it to be. In many places it is overwritten, and there is for the most part too much cross-referencing. For these things, I apologize. In retrospect, I suppose they happened because of my enthusiasm for discovering and demonstrating how one process interrelated to all the rest. As I discovered something, it tended to become immortalized far too quickly in writing, and this made for a non-compact expression of the ideas. And perhaps I tried to point out too many relationships which are either too obvious, or too obscure, to make them worthwhile. Therefore, as you read things for the first time, you are likely to be better off simply ignoring all cross-references. If you stick it out, the story unfolds in my conception of a logical order.

You can read at several levels of generality. If you have a couple of hours and just want to develop a general feeling for what I'm getting at, look in the table of contents and read those sections marked with two asterisks. If you want to absorb enough to argue with my ideas, read everything with one or two asterisks. If you are a masochist, read it all. My advisors and I did, and it didn't kill us.

Chapters 5 and 6 are important -- they are the heart of the thesis -- but they are long and tiring to read if you try to take them all at once. My suggestion is to read sections 5.1 - 5.3, look at the brief description of the classes of conceptual inference given in section 5.3, them jump right into the one which looks most interesting. Because they were basically written independently over a period of time (and not in the order presented), they should perhaps be read with with the same abandonment of organization. Don't forget to read chapter 8; it ties many ideas together.

The chapters are broken down as follows:

**Chapter 1, Introduction**

**Chapter 2, Representation: The Approach to Meaning** The representation of meaning is discussed. The chapter is mainly an overview of Schank et al.'s theory of Conceptual Dependency, which is the theoretical formalism which allows us to get at issues of meaning comprehension.

**Chapter 3, Representation: The Conceptual Memory** Another level of representation issues

arises when the relatively passive meaning graphs described in Chapter 2 must represented in the more active networks of a conceptual memory. The data structures which represent concepts, tokens, actions, states, times, strengths of belief, and so forth are developed.

**Chapter 4, Getting Conceptual Graphs into the Memory: Reference, Word Sense Promotion, Internalization** The processing which transforms the meaning graphs given the memory by the language analyzer into structures in the memory network is described. This includes how tokens of things in the world are identified from their language references, how the memory might interact with the language analyzer (affecting how it perceives incoming language), and how memory structures representing the information in an utterance come into existence. This chapter leads up to the point of conceptual inference.

**Chapters 5, 6: Conceptual Inferencing: A Subconscious Stratum of Cognition** These two chapters present the core of this theory of language comprehension. The notions of conceptual inferences and a multi-dimensional inference space are presented. 16 classes of inference are described. How they fit into the theory and how they have been implemented in the program are described.

**Chapter 7, The Inference Control Structure, The Structure Merger, and Other Aspects of the Program** The program processes which coordinate the functioning of the various kinds of conceptual inferences are described. How the program relates newly-inferred information to existing information, and what it does when relations are discovered, are described.

**Chapter 8, Inferences Applied to Reference Establishment and Time Relations** How the program realizes a very important theoretical interaction between the processes which identify tokens from language descriptions and the processes which generate new information from old by conceptual inferences is described. This chapter ties together many of the ideas of the previous chapters.

**Chapter 9, Conclusions, Future Work**

# CHAPTER 2

## REPRESENTATION: THE APPROACH TO MEANING

This theory of conceptual memory involves information representation issues at four distinct levels: a theory and formalism for representing the meaning content of a natural language utterance in context, and a theory and formalism for representing (and processing) information in a conceptual memory. These four levels are highly interrelated. This chapter and the next describe and relate them.

## 2.1 CONCERNING REPRESENTATIONAL FORMALISMS

Many of the ideas of conceptual processing to be presented in chapters 4-8, particularly those of chapters 5 and 6 concerning inference, can be viewed, at some level of abstraction, as existing independently from any particular scheme for representing knowledge. That is, much of this theory of conceptual memory describes and predicts the *flow of information* -- what needs to be done, and when -- and the reasons for this flow independently from details of substance and form of the information itself. Since they could exist independently from an effective formalism (one for which there is hope of implementation on a computer), we might call these ideas about memory and reasoning "meta" ideas. They will map out the crucial features of the theory. However, the realization of those features is left to a particular formalism which implements the theory, and the interaction between theory and formalism can be crucial: the formalism can determine the "tone" of the theory and influence its substance by uncovering new problems as it solves the ones already prescribed by the theory. Casting the ideas of the theory in an explicit formalism also helps delimit what is and is not possible, and what is and is not desirable in the theory.

Examples of these platitudes occurred frequently during the evolution of the memory. Looking back, it is difficult to sort out and reconstruct the subtle interplay between ideas and implementation. One good example concerns the development of function inferences (section 6.1). There, we will see a point at which the theory prescribes selecting between two alternate courses based on a very ephemeral test: "is there some unusual relation between person X and physical object Y?" Framing this question in the formalism of a particular theory forces us to

address the question: "what are effective test procedures which will discover such a relation?" The solution of this problem in the *memory formalism* pointed out a fairly general, effective notion of what it means for a relation to be "unusual", and this augments the theory of conceptual memory by uncovering a set of more specific and effective tests. These tests, now part of the theory, have become generalizable to other formalisms.

The moral is that, although a theory and a representational formalism can exist independently, their relationship can and should be a developmentally symbiotic one.

## 2.2 CONCEPTUAL DEPENDENCY: AN OVERVIEW

The representational theory and formalism adopted for this theory of conceptual memory is called *Conceptual Dependency*. Conceptual Dependency (CD), developed by Roger Schank et al. [S4,S6,S7,S8,G2], is a theory for representing the underlying meaning of natural language utterances, and is based upon two general precepts:

1. It is *independent from language form*; utterances in two languages which communicate the same thought are represented by the same structure in CD. Likewise, within a given language, utterances which communicate the same thought are represented by a unique meaning structure (a graph) in CD, regardless of differences in their form (what particular words and syntax were used).

2. It is a reductionist theory. It defines a small set of primitives, which, connected in various graph configurations by a small set of links, have the potential for representing any thought a human might have or communicate. The notions of *actions*, *states* and *causality* constitute the central core of CD's expressive power. The primitives and links are intended to bear psychological reality, and the intent of the theory is to extract the meaning content of utterances in the same ways and to the same units as we might expect humans do. There are many possibilities for experimental verification of the individual primitives and links, and of the theory in general.

The description of CD which follows is a description of that theory as it has been adapted and extended for the purposes of this thesis. There are no large differences between the goals and premises of "standard" Conceptual Dependency and those about to be described. Many of the variances concern small issues and the rest are extensions or elaborations of the basic theory. [S4,S6] give an overview of the "standard" theory.

The description will not cover the processes by which a language is effectively mapped onto this formalism. There is a computer program which can analyze English sentences at the

level of complexity of most of the examples I will use. The theory of analysis and a description of this program can be found in [R2].

Conceptual Dependency consists of six components: a set of action primitives, a set of primitive states, a number of psychological and physical scales, an open-ended set of primitive concepts, a set of conceptual links, and a set of rules which specify the well-formedness of combinations these entities. Rather than state the rules formally, I will instead develop an intuitive feeling for the well-formedness of combinations of the other five types of objects. A well-formed combination of these objects is called a *conceptual graph.*

### 2.2.1 ACTION PRIMITIVES

The memory uses 11 action primitives (ACTs). What is an action primitive, and where do they come from? The ACTs arose by collective introspection, with an eye kept to (1) their psychological reality, (2) their descriptive efficacy in the conceptual domain, and (3) their viability as the atomic units for effective computer procedures. Never in the development of the primitives, was one of these three considerations allowed to overshadow the others completely. In addition, since the goal was to be able to represent a broad spectrum of common daily discourse -- to talk about people, what they do and talk about daily -- rather than some esoteric or more technical discipline, the primitive ACTs are actions that *people* do. In fact, one of the rules of primitive ACTs is that only humans (or their personification by machines and natural forces) can serve as actors. Books don't "fly" across rooms; they are propelled by a person, machine or natural force.

#### 2.2.1.1 CONCEPTUAL CASES

Each ACT governs a *conceptual case framework,* which consists of from 2 to 4 *nuclear* cases and several *incidental* cases. A conceptual case may be thought of as a slot, a placeholder, into which some concept or other conceptual graph fits. All conceptual cases, whether nuclear or incidental, are obligatory; that is, a conceptual graph involving an ACT is not well-formed unless the contents of all its slots have been filled as well as possible. This frequently amounts to filling some cases with "dummies" because, at the time the conceptual graph is constructed, the identity of a case filler may be unknown and not predictable. Specifying such "missing cases" as best it

can in the contextual environment in which it occurs is one task of a conceptual memory. Although I will frequently write conceptual actions without specifying all the cases, this is for convenience only.

Nuclear and incidental cases are distinguished on the basis of their "intimacy" with the ACT which governs them. Cases without which the ACT could not exist even as an abstracted or idealized event in the world are nuclear. For instance, the ACT GRASP simply cannot stand for an event without its nuclear ACTOR and OBJECT cases. *Someone* must grasp *something*. The ACT, together with its nuclear cases, is in some sense that which is "imagineable in the mind's eye" independently from its other features of time, location, instrumentality, and so on, even though we know that for a real action to exist, it must also have these attributes. These attributes which are nonessential to the "inherent mechanics" of the ACT are the incidental cases. It should be emphasized that the term "incidental" does not imply that these cases bear only incidental significance in subsequent analysis by the memory. We will see in fact that the information communicated by incidental cases is sometimes more significant than the nuclear action itself.

This distinction between nuclear and incidental cases is not made in "standard" CD. However it is quite useful here, since it bears directly on the data structures which store actions in the memory. Fig. 2-1 defines the conceptual cases and indicates their CD graph notation. In the table, "A" stands for an ACT, "X" indicates where the case filler is attached. Case links are members of the larger set of dependency links, soon to be described.

| NAME | SYMBOL | DESCRIPTION |
|---|---|---|
| ACTOR | X <===> A | The "main link", denoting the actor-relation. The actor case filler must be animate. |
| OBJECT | A <-o- X | The conceptual object of an action. All primitive ACTs except MBUILD govern a conceptual object. |
| RECIPIENT (to-from) | A <-R-> Xt, Xf | The donor (Xf) and recipient (Xt) of an action involving the abstract transfer of an entity. |
| DIRECTIVE (to-from) | A <-D-> Xt, Xf | The beginning (Xf) and end (Xt) points of an action which changes an entity's mental or physical location. |
| INSTRUMENT | A <-I- X | The action (X) by which another action (A) occurs. X further specifies A, and must always be an action. |
| TIME | X <===> A | All time aspects (X) are noted above the main actor-action link. Section 2.2.10 describes the various aspects in CD. |
| LOCATION | A <-L- X | The physical location of an action. Any physical object can be a location in this context. |

**Figure 2-1. The conceptual cases.**

#### 2.2.1.2 SCOPE OF THE ACTION PRIMITIVES

Before describing the ACTs, a short aside is in order. The action primitives about to be described are not intended to account for all of language. This does not mean that, when pressed, an expert could not render some approximation to just about anything by using only these action primitives; judging from experience, he probably could. But that is not the issue. The real issue concerns not the primitives themselves, but rather what they mean to the system, how they are combined, and what they predict and explain concerning language processing in humans. We must again take care to distinguish the *specifics* of the CD formalism from its *theory*.

The choice of particular action primitives and their resulting descriptive potential constitute the formalism. Their adequacy, saliency, or even correctness is always subject to question; hence, they are always subject to revision. This is not disturbing, since the real substance of the theory as an approach to language understanding transcends the particular choices of primitives, and, although there has certainly been the same kind of developmental relation between the formalism and theory of CD as between the formalism and theory of memory, the particulars of the formalism, as with the memory, are malleable.

This is merely a caveat, not an apology for the specific primitives posited by CD theory. In fact, after a while, one gains an intuitive feeling of their adequacy, correctness and tremendous descriptive power in the domain of humans' day to day interactions and discourse. This domain is small enough to explore in depth and work with; yet it is large enough to be interesting, because it touches most of the real issues of language. These primitives constitute a powerful core from which we might expand.

### 2.2.2 THE PRIMITIVE ACTS

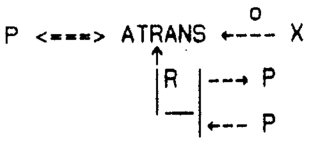
In this description of the 11 primitive ACTs, the following "typed" objects will be used:

| | |
|---|---|
| P | a person (something capable of acting, possibly personified) |
| CON | any non-atomic conceptual graph (a complete conceptualization, as opposed to a simple concept) |
| X | a physical object (a person can be a physical object) |
| L | a location (any person or physical object can be a location) |
| M | a "mental" location (explained later) |
| B | a bodypart |

(?) will denote that, although the case is present, its content is unknown; (??) will denote a query.

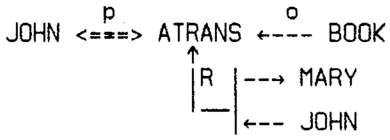
Each ACT will be defined by (a) describing in English its conceptual meaning, (b) specifying its "skeleton" conceptual template of nuclear cases, and (c) illustrating a typical usage. Recall that the incidental cases exist for every ACT; they are simply not shown here. Although the representation of time will be described later, we will need the symbol for "past time" in some of the following examples. This is simply a "p" situated over the actor-action main dependency link.

## ATRANS

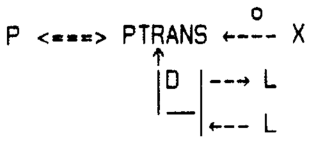



The "abstract" transfer of possession of an object from one person to another. This ACT accounts for verbs of "giving" and "taking", and underlies all verbs which have as a feature this abstract notion of change of possession. It is important to note that ATRANS generally leads by inference to a change of possession, but this change is not properly a part of the ACT itself (eg. "John gave Mary a book, but she refused it.")

EXAMPLE: "John gave Mary a book."

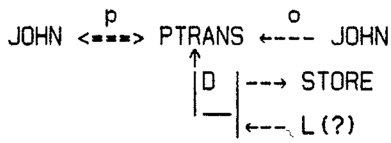



## PTRANS

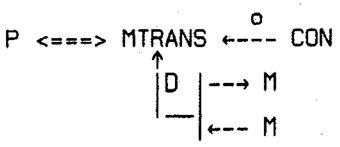



The "physical" transfer of an object from one location in space to another. PTRANS underlies verbs of "going" (the object is a person), "handing" or "moving" (an object), etc. Although PTRANS and ATRANS are independent, PTRANS is frequently the instrumental ACT for an ATRANS. Notice PTRANS governs the directive case, whereas ATRANS governs the recipient case. Just as with ATRANS, PTRANS does not guarantee that X ends up at the location specified in the directive case (eg. "John went to the store, but he got sidetracked."). That X ends up at the location toward which the PTRANS occurred is, however, a highly probable inference.

EXAMPLE: "John went to the store"

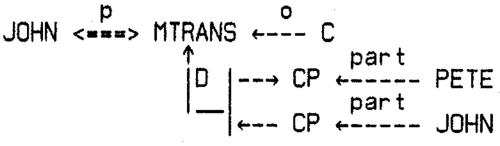


## MTRANS



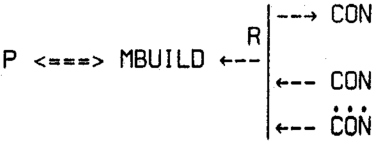

The movement of "mental" objects from one "mental location" to another. MTRANS underlies many verbs of thought (cognitive functions) and communication. Examples of verbs whose central idea is MTRANS are: "tell", "remember", "recall", etc. MTRANS actions involving more than one person frequently have SPEAK as the instrumental ACT. MTRANS actions which involve only mental locations within one individual have no instrumentality in Conceptual Dependency. Although the ACT of MTRANSing does strongly imply that the mental object starts existing in the mental location to which the transfer occurs, the ACT of MTRANSing does not guarantee this in itself ("John told Mary he was going to the store, but she wasn't listening.")

EXAMPLE: "John told Pete he went to the store."



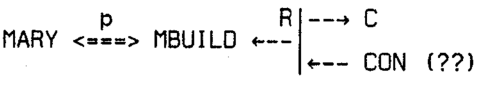

(where C is the graph in the PTRANS example and CP is a person's "conscious processor")

## MBUILD




The synthesis of a new mental object from one or more old ones. This ACT underlies many verbs of thinking, problem-solving, deciding, reasoning, etc. The recipient case symbolizes that the new CON (the topmost one) "receives" its existence from the CONs represented beneath it. These are the thoughts which played a part in the synthesis, and are frequently unspecified by language and not surmiseable. Although MBUILD instrumentality is generally uninteresting in the CD framework, it is always either a CONC or some form of MTRANS, usually to P's CP (conscious processor).

EXAMPLE: "How did Mary figure out that John went to the store?"




(where C is the graph in the PTRANS example)

## CONC

P <===> CONC <---(o) CON

The "conceptualizing" of a mental object. CONC indicates that CON is the focus of the active thought process in P, and underlies verbs like "notice", "be conscious of", "be aware of", "realize", etc. it is distinguished theoretically from CON simply having mental location CP in that it implies the spontaneous existence of CON in P's conscious processor, whereas MLOC(CP) implies that an MTRANS caused CON's existence in P's conscious processor (hence that CON did not arise spontaneously).

EXAMPLE: "Bill was aware of John's going to the store."

BILL <---(p)---> CONC <---(o)--- C

(where C is the graph in the PTRANS example)

## ATTEND

P <===> ATTEND <---(o)--- S

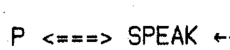
The "attending" of a person to one of his sense organs. ATTEND underlies verbs of perceiving, sensing, etc., and normally does not stand alone, but rather is the instrumental ACT by which a CONC occurs. The sense organs are: EYE (look at), EAR (listen to), NOSE (smell), SKIN (feel), TONGUE (taste), and are implicitly part of P.

EXAMPLE: "John saw Mary giving Bill a coat."

JOHN <---(p)---> CONC <---(o)--- C
<br>↑
<br>|I
<br>|
<br>JOHN <---(p)---> ATTEND <---(o)--- EYE

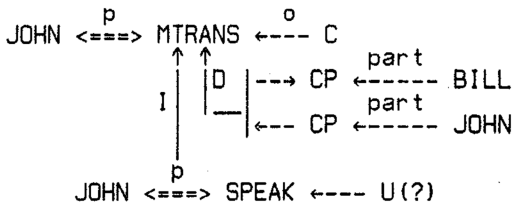
(where C is the graph for Mary ATRANSing a coat from herself to Bill)

## SPEAK



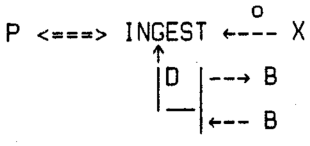

The uttering of sounds. U is the sound string (word sequence). SPEAK underlies verbs of speaking, saying, conversing, yelling, etc., and rarely stands alone. Instead, it is normally the instrumental ACT for an MTRANS between P and another individual.

EXAMPLE: "John verbally informed Bill of his departure."



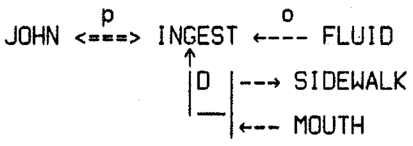

(where C is JOHN's PTRANSing from wherever he and Bill are to somewhere else, and where it can be predicted that U is something like "I am going", or "Bye")

## INGEST

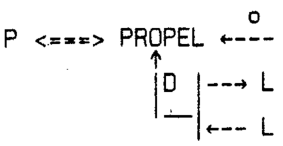



The act of moving an object, X, into or out of an internal bodypart. INGEST is distinguished from forms of PTRANS because the movement is effected by natural and internal bodily functions rather than by explicit "external" actions in the world. INGEST underlies a very diverse class of verbs, examples of which are: "breathe", "eat", "cry", "sweat", "swallow", "belch", etc. The directionality of the ACT determines whether the action is inherently an "ingest", "expel" or internal movement of an object or fluid. Bodyparts commonly referenced by INGEST are STOMACH, LUNG, MOUTH, NOSE, EYE.

EXAMPLE: "John expectorated on the sidewalk."

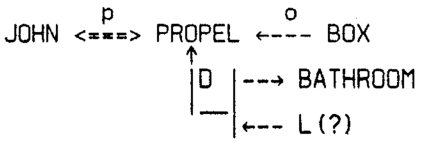


## PROPEL

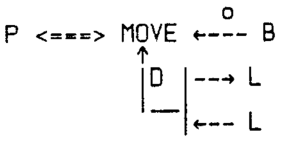



The application of a mechanical force (in a certain direction) to an object. This ACT underlies verbs of throwing, hitting, pushing, pulling, etc. The instrumental action for PROPEL is either another PROPEL, a MOVE or a GRASP-MOVE-UNGRASP

EXAMPLE: "John pushed the box into the bathroom."

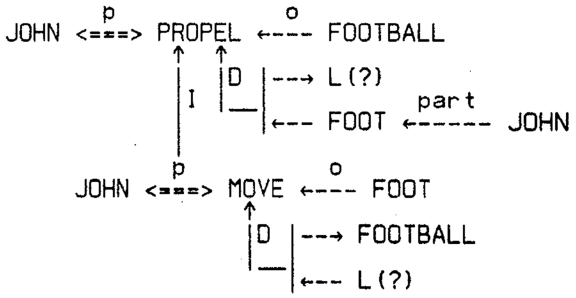



## MOVE

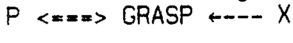



The movement of a bodypart. As with INGEST, this ACT is distinguished from forms of PTRANS because it arises from an internal body capability and has no causal involvement with the outside world. MOVE is typically found as the instrumental action for PTRANS and PROPEL, and is essential to verbs such as "hand to", "touch", "kick", "nod", etc. It has no instrumental case for the purposes of CD analysis.

EXAMPLE: "John punted the football."




## GRASP




The grasping of an object by the hand. GRASP underlies verbs such as "pick up", "clutch", "grab", "let go of", and frequently appears as an instrumental action (in conjunction with MOVE) of PROPEL. The action of ungrasping is a GRASP which ceases.

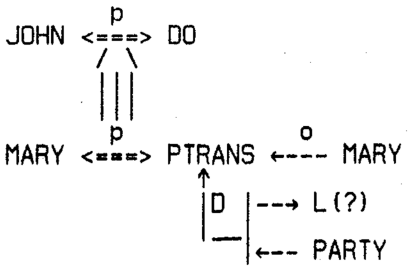
EXAMPLE: "John let go of the apple."



#### 2.2.2.1 THE DUMMY DO

In addition to these 11 specific actions, CD utilizes a "dummy" action to stand for some unknown action, that is, an action which must must exist, but whose exact nature is not explicit. This place-holder is denoted "DO", and takes only an actor:

EXAMPLE: "John caused Mary to leave the party."




(the triple-barred arrow between the dummy DO and the PTRANS being the CD causal link, to be explained shortly).

In the memory, actions are stored by structures of the forms

```
(ACT ACTOR OBJECT)
(ACT ACTOR OBJECT FROM TO)
```

### 2.2.3 PICTURE PRODUCERS

In this description of the primitive ACTs, I have made implicit use of all sorts of entities in the world: JOHN, BOOK, STORE, FLUID, etc. These are clearly representatives of an open-ended set of real world ideas and concepts. Since, when we hear the name of one we are immediately able to conjure up an "abstract" or "idealized" image, objects in this open-ended set are called *picture producers* (PP's) in CD terminology. PPs bear a close correspondence with dictionary *word senses*, in that two vastly different concepts may happen to have the same name in a language. Where there is ambiguity, we should technically write PP's with a subscript to clarify which "picture" we are trying to elicit by the word.

One other point deserves mention here. For the purposes of CD representation, it is

adequate and desirable to stop at the "picture" stage. That is, if conceptual *representation* were the final goal, it would suffice to represent, say, the concept "John" as the PP "JOHN1" (a male human whose name is John), without knowing *which* John in the real world is the target of the reference. However, this determination of real references is a very important task for a conceptual memory, and can sometimes be crucial to a conceptual analyzer's ability to construct the best possible conceptual graph in context. (For example, "John's pitch was foul" should come out one way if the analyzer knows the John being referenced is a roofer, whereas a completely different conceptual graph should result if John is a door-to-door salesman!) This task of determining the referent of PPs in context is discussed principally in chapters 4 and 8.

### 2.2.4 MENTAL LOCATIONS

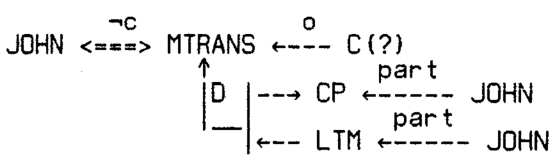
Certain of the primitive ACTs and states make reference to "mental locations". CD's expressive power regarding verbs of thought and communication is couched (together with the primitive mental ACTs) upon three abstract mental locations in humans: the "conscious processor" (CP), "immediate memory" (IM), and "long-term memory" (LTM). In addition, any information-bearing entitity may be personified as a mental location. (This includes books, computers, meters, etc. Thus, for instance, to read a book, one MTRANSes the information whose mental location is the book to his CP.)

The notions of CP, IM and LTM drag along with them such an entourage of psychological overtones that I will not attempt to justify them as psychological realities. In fact, this is not their purpose in CD; in CD they exist simply as intuitive abstractions which provide expressive power and latitude when used with the primitive mental actions. The CP is where ideation takes place-- the focus of thought, the locale of one's conscious awareness. IM is what "surrounds" the CP, representing knowledge which has recently been active or which has been associatively "drawn in to peripheral consciousness" by the activity in the CP. (Section 4.3 will suggest how an effective definition of this idea can be framed.) The LTM is the inactive storehouse of knowledge which may be drawn into the CP or IM. By convention, existence of a conceptualization in LTM means that that conceptualization is believed; existence in the CP implies that the conceptualization is being "thought about".

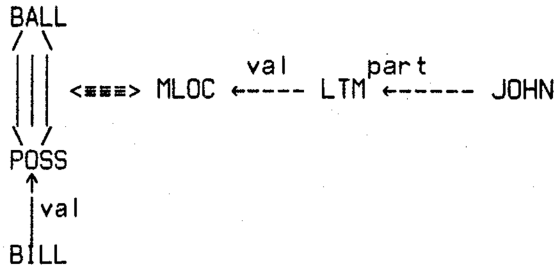
The following two examples suggest the potential expressiveness of these mental abstractions:

"John can't remember."

(John is unable to transfer something, C, from his LTM to his CP)




"John believes that Bill has a ball."

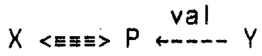



(where the two-headed, three-barred arrow is the attributive link, about to be described).

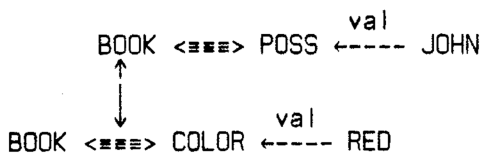
### 2.2.5 STATES AND ATTRIBUTION

In opening, I mentioned that the three notions most fundamental to the Conceptual Dependency theory are actions, states and causality. The primitive ACTs have been discussed, and the CD notion of causality will be examined shortly. We are interested here in the notions of states, statechanges and statechange scales.

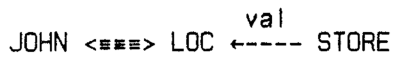
A state (sometimes called an attribution or conceptual feature, depending on what is being focused upon) is represented in CD by the *attributive* link, <≡≡≡>. The interpretation of X <≡≡≡> Y is that "X has the property, or is in the state of Y". Since states frequently are relations involving at least two PPs, the conceptual *value* link,



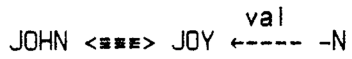

frequently occurs to denote the "value" of X along the "dimension" P -- X's value with respect to relation P. For instance, to represent "John has a red book" (or "A red book is possessed by John"), we write



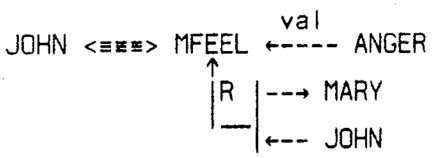

that is, "a book (such that the book has color, and that color is red) is possessed by someone, and that someone is John". (The two-headed, single-barred arrow is the "REL" link, defined in section 2.2.7.1.) To say that John is at the store, we write

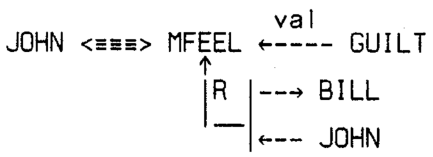



and to say that John is depressed, we write




which is "John is at some negative value on his JOY scale". To say that John is angry at Mary, guilty with respect to Bill, we write




respectively. That is, "John is feeling an emotion, this emotion is (anger/guilt), and it is directed toward (Mary/Bill)." The recipient link is the same one used as a case for ATRANS, but here it is not properly called a case. The last two examples relate to emotional scales which will be described shortly.

These few examples characterize the notions of states and attributions in CD. Although the number of states required to describe the world is quite large, the following handful of state predicates listed below (in their memory format) are, empirically, the workhorses of CD. This list does not include scale-related states, since these are discussed in the next section.

```
(WANT P X)        some state, statechange or action, X, is desired by P
(POSS X P)        an object, X, is in the possession of person P
(LOC X L)         an object, X, has some physical location, L
(MLOC X L)        a conceptualization, X, has some mental location, L
(MFEEL P1 E P2)   P1 feels emotion, E, toward P2
(BE X)            an object exists in the world
(INVOLV X Y)      concept X is involved in some way in conceptualization Y
```

It should be pointed out that the use of the state predicate WANT represents a deviation from "standard" CD theory. In CD, "P wants X" is represented by a structure of the form "P believes that X's occurrence or existence would cause P to increase in joy." Since, as we will see, the notion of WANT is so fundamental to conceptual memory processes, this state has been made primitive. For communication with the language analyzer and generator (which use the strict CD pattern), the memory intercepts and re-synthesizes this WANT pattern at the interfaces with these programs.

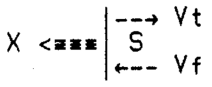
Time and duration are obligatorily associated with all states, although these associations are not called cases, since this term refers to actions. Locations are never associated directly with states; in order to express a thought such as "John was sick in Peoria", we write the state of being sick with starting and ending times t1 and t2, such that the interval t1-t2 overlaps with the interval during which John was located in Peoria. That is, what this really means is "John was sick *while* he was in Peoria": one state (sickness) existed during the time of another state's (location) existence. Although it is possible to represent the location of actions in the same way, it is more convenient to use the notion of a location case for actions. (Since actions can usually be viewed as instantaneous, there are no ambiguous overlappings of intervals.)

### 2.2.6 STATE SCALES AND STATECHANGES

Some actions in the world cause new, discrete states to come into existence; an example of this is the new location achieved by an object which has been PTRANSed away from its former location. These are the states I have just discussed. However, there are many other states which are in some sense continuous. Two examples of reference to continuous states are seen in the sentences:

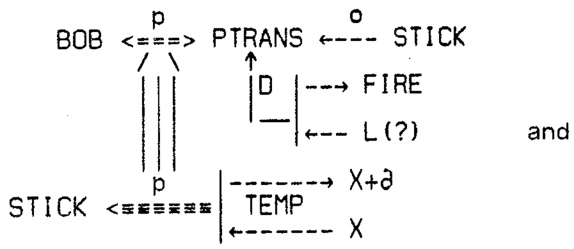
1. Bob heated the stick by putting it into the fire.
2. Mary cheered John up.

(1) involves the notion of statechange along the temperature continuum, or *scale*, (2) along a psychological continuum, the "joy" scale. In CD, such a change along a continuous state scale is denoted by the *statechange* link, which relates a PP, X, to values on the scale, S:

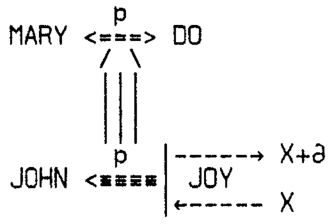



(STATECHANGE X S Vf Vt)

In the memory this construction is represented by the STATECHANGE predicate illustrated to the right. By use of this conceptual link, and the notion of state scales, the two sentences above can be represented, respectively, by the graphs:




and




respectively. (Again, the single-headed, triple-barred link denotes the CD causal relation -- section 2.2.8).

The interrelationship between points on psychological and physical scales and statechanges along those scales should be clear: to say that John is happy is to predicate that John lies at some positive point on the JOY scale; to say that John *became* happy is to say that he underwent a statechange to some positive value on the JOY scale. Points along psychological scales, and other scales for which there is no obvious metric (absolute temperature, for example, has an obvious metric) are defined as integers lying between -10 and 10. Roughly speaking, negative values are "undesirable", positive ones "desirable"; positive changes are good, negative ones bad. Thus, some scales have a negative orientation: to become more angry is to undergo a negative change on the ANGER scale. Some of the more common CD scales (shown with their orientations) are:

| | |
|---|---|
| JOY | positive |
| ANGER | negative |
| FEAR | negative |
| GUILT | negative |
| PSTATE | positive |
| HEALTH | positive |
| AWARE | positive |
| BENEFIT | positive |
| RELSIZE | positive |

Much research concerning the exact interdependencies among scales, remains to be done, particularly for the psychological ones. Clearly, they are not independent, but the nature of their interdependence remains to be made explicit.

In order to relate conceptual state scales to language, the conceptual analyzer has "standard" mappings from words and constructions in the onto points along the scales. For instance, "John is happy" becomes "John is at point +2 on the JOY scale"; "John infuriated Pete" involves "Pete changed state to -4 on the ANGER scale", and so on. This kind of oversimplification makes possible the efficient and effective analysis into, and generation out of CD, and it is adequate for these purposes. However, the assignment of a specific point on a scale to some language construction is more often than not ludicrous, and somewhat arbitrary for capturing the real meaning of an utterance in context. To avoid these problems, the memory acknowledges scales' inherent fuzziness by transforming statechanges into one of four "statechange" predicates: POSCHANGE, NEGCHANGE, BIGPOSCHANGE, BIGNEGCHANGE, based on the absolute numbers predicted by the analyzer. As with WANT, these forms are transformed at the interfaces with the analyzer and generator to make the memory compatible with these processess. Occasionally, fuzziness is not an issue, as in

John was euphoric. ----→ (JOY JOHN +10)
Pete died. ----→ (STATECHANGE PETE HEALTH X? -10)

so that absolute points are sometimes useful.

In summary, the statechange- and scale-related notions to which the memory is sensitive (expressed in memory notation) are:

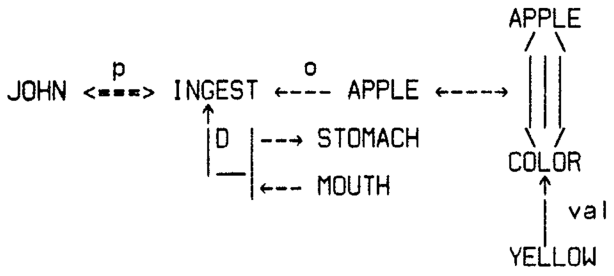
```
(STATECHANGE PP SCALE V-FROM V-TO)
(POSCHANGE PP SCALE)
(NEGCHANGE PP SCALE)
(BIGPOSCHANGE PP SCALE)
(BIGNEGCHANGE PP SCALE)
(<scale> PP VALUE) (a point on a scale)
```

### 2.2.7 ADJECTIVES AND RELATIVE CLAUSES

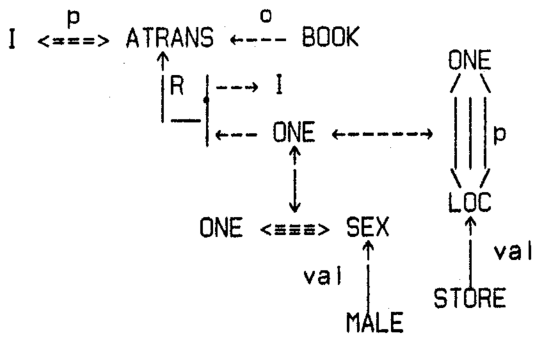
The underlying conceptual representations for "true adjectives" and most relative clauses in surface language are the same: both make use of the CD *REL* link (←→), which denotes additional conceptual attribution: conceptual features which are peripherally communicated about objects in an utterance. By "true adjective" I mean that the adjective really predicates a *conceptual* feature about the object, rather than simply having adjectival form in the sentence. Often, sentential adjectives have no relation with conceptual adjectives, as in the sentence "Mary gave John a bad beating." Here, although "bad" is sententially an adjective modifying "beating", it conceptually predicates the intensity of a hitting action, and hence is conceptually adverbial rather than adjectival.

#### 2.2.7.1 THE REL LINK

The REL link associates a PP with a complete conceptualization *in which that PP occurs.* The interpretation of this association is that the PP has the additional conceptual feature denoted by the conceptualization. Thus, to represent "John ate a yellow apple.", we write




And to represent a relative clause attribution such as "I took the book from the man who was at the store.", we write




Notice that, in general, more than one additional attribute can be associated with a PP.

### 2.2.8 CAUSALITY

Causality is a deep and many-faceted notion. This section will simply describe the types of causality used in CD, and show how they are used to achieve broad expressibility, without arguing for their correctness or adequacy.

In representing causality as it is used in language, we are not concerned with "correctness". That is, causality, as a language user employs it, is not necessarily the "real" causality in the world. For instance, we may *assert* that two physical events are causally related, even though, within a particular model of the world, there may by no explicable causal relation between them. Hence, language assertions of causality can exist independently from their reality within a world model. This difference defines one interesting task for a memory: one subprocess of understanding is to reconcile causal relations *communicated by language* with causal relations *in the memory's world model.* This issue is addressed in section 5.5.

There is, in addition, the deeper issue of whether or not the notion of causality expressed in language should be represented in the same way as the notion of causality which explains cause and effect relations in the world model which deals with language. Hopefully, from the standpoint of a language-understanding program, these two uses of causality can be thought of as referencing the same underlying notion: what may be "real world" causality in one person's model may either be reduceable to smaller units, or inexplicable in another's. That is, if we view the causality expressed in language as directly reflecting some alien model of the world, then we

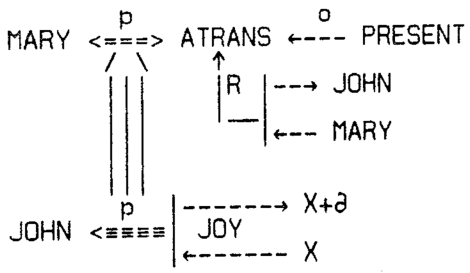
can view language causality and model causality as one and the same. Still, some record of what is internal to the model and what enters via language must be kept, for the memory must be capable of distinguishing what it holds to be true from what it has perceived via language.

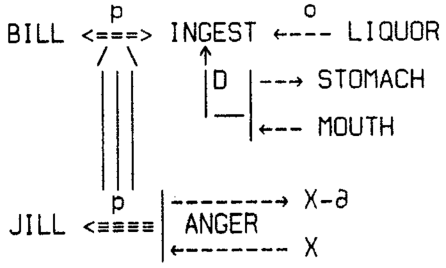
#### 2.2.8.1 THE CONCEPTUAL CAUSAL LINK

Causality in CD is denoted by the causal link, <≡≡≡. To assert that "X caused Y", we write

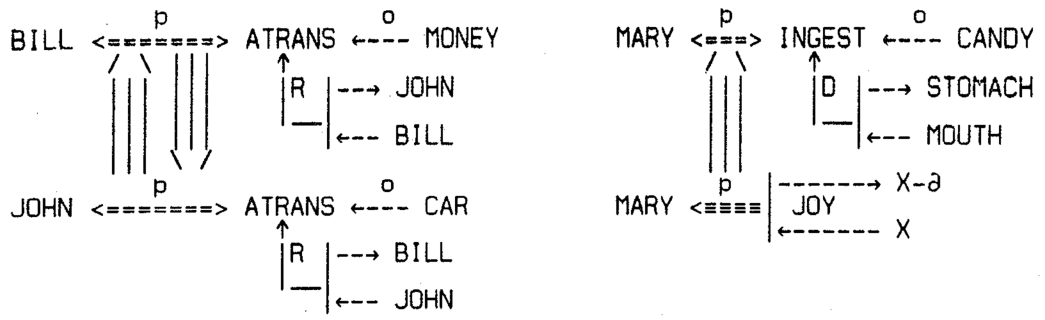



the interpretation being: X and Y both occurred, and Y occurred *because* X occurred. Thus, to represent "Mary made John happy by giving him a present.", and "Bill's drinking angered Jill", we write, respectively:



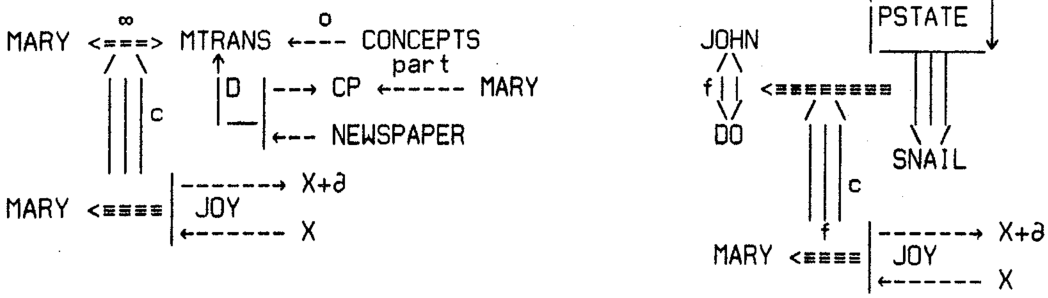

In addition to serving in this explicit capacity, the causal relation is frequently implicit in the underlying CD representation of individual words. Examples of this are with the verb "buy", as in "Bill bought a car from John", and "dislike", as in "Mary disliked the candy", which are represented, respectively, as:




(double, or mutual, causality underlies many two-party verbs like "buy" and "sell").

#### 2.2.8.2 CONDITIONAL CAUSALITY

The simple causal link expresses the causal relation between events which actually occurred. But language makes frequent use of probable or *conditional causality* for expressing the *potential* for causal relationship between two events. In CD, conditional causality is denoted by a causal link with a "c" beside it. Thus, to represent "Mary likes to read newspapers", and "John could please Mary by killing the snail.", we write




(the "infinity" mark over the MTRANS main link denotes timelessness, "f" marks future time.)

Simple and conditional causality form the central core of CD's ability to represent causality. However, there are many potential combinations of these two forms with negation and capability markers on the events they relate, and on the causal links themselves. Although it is possible to *enumerate* all such forms, many issues concerning their exact meanings and effective mapping procedures from language onto them remain to be researched.

The notions of simple and conditional causality are represented in the memory by the structures:

(CAUSE X Y)

(CANCAUSE X Y)

meaning "X causes Y" and "X could cause Y", respectively. Much more will be said about the uses to which these predicates are put in chapters 5 and 6. Much of the memory's knowledge of causality in the world is stored in the form of programs rather than in passive structures which make explicit use of these predicates. The relationship between these predicates and the programs which implicitly store causal knowledge will become clearer in subsequent chapters.

### 2.2.9 INSTRUMENTALITY

The conceptual instrumentality of an action is the specific *means* by which the action occurs. Although it is difficult to define and often hard to distinguish from causality, *conceptual* instrumentality is nevertheless quite different from the more common notion of *linguistic* instrumentality. Whereas linguistic instrumentals are syntax forms, frequently signalled by "by", and usually associated with some surface verb, conceptual instrumentality can be communicated in countless ways, and always serves to further the description of an underlying action, X, by making explicit (via another action, Y) the *means* by which X occurred. In a sense, then, the instrumental ACT, Y, makes the main ACT, X, more specific, even though there is no intrinsic heirarchy of specificity among the primitive ACTs.

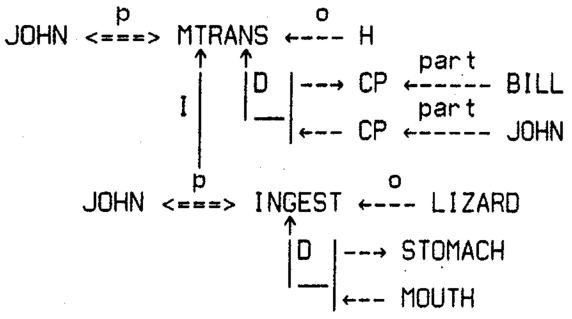
At a very abstract level, one could argue that conceptual instrumentality is only a fiction. Indeed, if a representation were "utterly primitive", that is, it described the world solely in terms of the "real" physical primitives of atomic particles and their laws of causality, perhaps there would be no need for instrumentality. Everything would be described "as it was", and this description would be devoid of any bias or interpretation. However, the moment we impose an interpretation on some *combination* or *sequence* of these utterly primitive events, such as "that sequence of sound-production was an act of communication" (MTRANS), we abstract the situation out of the domain of utter primitives to a higher-level interpretation of what happened. It is higher-level because it then characterizes a very complex event by an "off-the-shelf" higher-

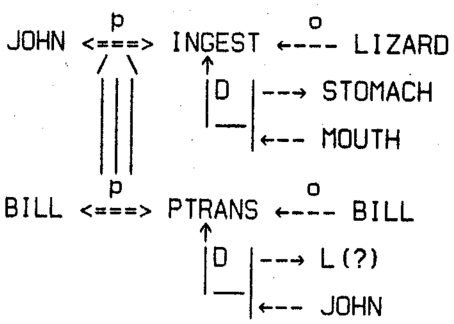
level pattern, and tantamount to this pattern classification is the loss of information. There must therefore be some means in this higher-level system for *selectively adding back* some of the information that was lost in the process of interpreting. This is the job of an instrumental.

This information loss occurs in the CD representation, since its "primitives" are actually very high-level abstractions. The instrumental case allows us to replace lost information in the abstracted interpretation by saying, for example, "that was an MTRANS action, and furthermore, it occurred by acoustic means, namely, a SPEAK". Here, the SPEAK puts back information which was lost in the process of classifying the action as an MTRANS.

Conceptual *instrumentality* is usually distinguishable from *causality* on the basis of "microtimes". An instrumental action is always contemporaneous with its main action, whereas two actions which are causally-related usually occur sequentially. This is not a universal truth, but rather a rule of thumb. If the actions occur at the same time, and one further describes the other, there is probably an instrumental relationship. Otherwise, the relation is probably causal in nature.

To illustrate, contrast "John communicated his hunger to Bill by eating a lizard" with "John drove Bill away by eating a lizard". The first is underlied by a true instrumental relation, whereas the second illustrates a causal relation:




(where H stands for the graph "John is hungry")

Also, within any framework of specific actions and states, there is another obvious rule for distinguishing causality from instrumentality. For verbs which are underlied in the theory by a state or statechange (rather than an action), actions which might appear to carry instrumental

modification are in reality carrying causal information about the underlying state: states and statechanges simply have no instrumentality! They occur by causality. For example, "John pleased Mary by singing" relates "sing" with "please" causally, because conceptually, "to please someone" is not an action at all, but rather is underlied by a "do-cause-statechange" on a psychological scale. Since "to please" is not underlied by a primitive action, it cannot have instrumental specification.

In CD, the link

$$\xleftarrow{\quad I \quad}$$

is used to denote that X occurs by instrumentality Y. In the memory, this is stored as (INST X Y). Instrumental actions, viewed as information-bearing subpropositions, constitute an important source of information from which to generate inferences. Also, by predicting (filling in) unspecified instrumentality, important lines of inferencing can result which would not otherwise occur from an input.

### 2.2.10 TIME

The time aspects of a conceptualization are noted above its "main link". Although, strictly speaking, only actions and states can have time aspects, the time of an entire causal structure is commonly associated with the causal relation itself, rather than the events it relates. The interpretation of this notation is that the causing action occurred at the specified time, and the caused conceptualization occurred immediately thereafter.

It is possible to represent the following time aspects in CD: ("NOW" refers to the time of utterance, CON refers to the conceptualization to which the time modifications are attached)

**NULL** (no time marking) CON is occurring NOW

**p** CON occurred or was in progress at some (indeterminate) time before NOW

**f** CON will occur or be in progress at some (indeterminate) time after NOW

**t=x** CON occurred or was in progress at time x

**ts=x** CON started at time x

**tf=x** CON finished at time x

$\infty$ (timeless) CON is a time-independent statement of fact

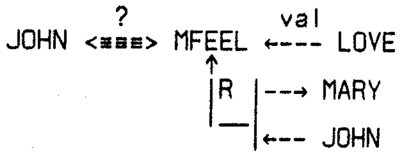
Explicit relations among the times in a graph are noted separately from the graph. Notice that durations are specifiable by their endpoints, which can be represented by TS and TF. Thus, for example, to represent "X occurred *while* Y", we assign X a time, t, such that t is greater than Y's starting time and less than Y's ending time.

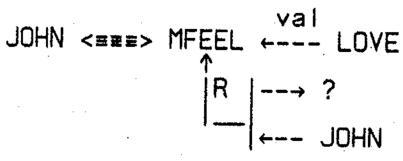
Neither CD nor the memory deal with more complicated time considerations such as frequency; this and other more complex time aspects require more research. However, the time aspects listed above seem (empirically) to account for a fairly large portion of time predications in ordinary language, and permit us to do interesting things. Furthermore, the feeling is that there is a very small number of these higher level time relations like frequency, pseudo-continuous states (ie. where a state is continuous, except for several "discontinuities"), and so on. If this is the case, the main burden is not on the representation, but rather lies in what an intelligent program *does* with that representation.

Section 3.6 describes how time information is stored in memory, pursuing some of the details of how time concepts are created and stored, and how deictic time references like "yesterday" and "last year" are handled.

### 2.2.11 INTERROGATIVES

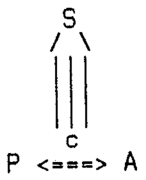
Interrogatives are denoted by a question mark (a) associated with some conceptual link, or (b) in the place where a PP would normally occur. The first form denotes a yes-no question about the validity of the conceptual link, whereas the second form denotes a request for some unknown information. Thus, to represent "Does John love Mary?" and "Who does John love?", we write, respectively,



### 2.2.12 CAPABILITY AND NEGATION

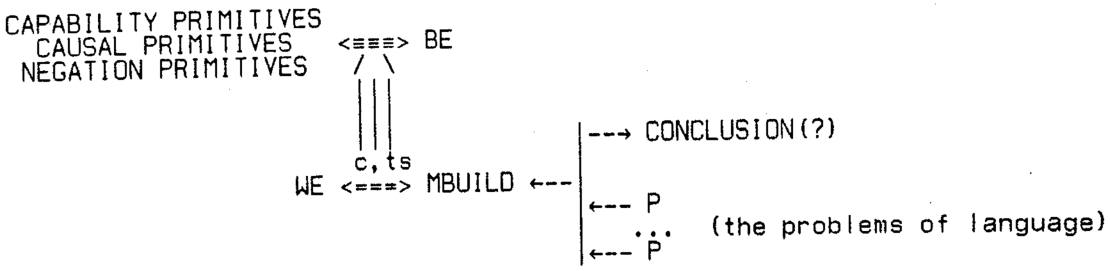
The ability of an actor to perform an action is denoted in CD by a "c" situated over the main actor-action link of a conceptualization. The lack of ability is denoted by a "¬c" symbol. In the memory, these modifications assume the role of "main conceptualization" by the forms (CAN X) and (CANNOT X), X being some action. The respective interpretations are that the actor does or does not have the ability to perform the action, X. The reasons for his ability or inability are represented as the causes of the CAN or CANNOT proposition.




that is, state S enables actor P to perform action A.

Negation is denoted in CD by a "slash" through a state, action or causal link. The interpretation of a negated causal is that two events occurred, but they bore no causal relation to each other. A negated conditional causal indicates that one event is incapable of directly causing another event.

It should again be pointed out that the CD coverage of capability and negation, taken in various combinations with causation, is in need of considerable elaboration. However, what there is of these notions enables us to get on with some interesting issues of language:



## 2.3 CONCLUSION

Conceptual Dependency is a modest but solid foundation upon which to develop a comprehensive, language-free theory of language. It allows us to represent the underlying meaning of utterances in a way which is independent from the form of the language string which communicates those utterances. This has a very appealing practical value because it allows cognition to be framed in a theory which is independent from any particular language: the memory will function equally well in Chinese and Swahili, assuming suitable conceptual analyzers and generators exist.

But more important, by employing a conceptually primitive meaning representation, we remove one very tenacious level of complexity from each utterance *before* the memory begins its analyses. This leaves the theory and program of conceptual memory free to get more directly to the deeper issues of cognition.

# CHAPTER 3

## REPRESENTATION: THE CONCEPTUAL MEMORY

The *storage in a memory* of conceptual graphs and the objects they reference introduces theoretical dimensions of representation which are not addressed by CD theory. Most of the new issues concern effective organization, referenceability, the ability to distinguish tokens of concepts, and inferenceability -- in short, all those things which integrate the "passive" conceptual graphs into a more "dynamic" format. These considerations constitute part of the interface between language and memory, and comprise a separate theory of their own. This chapter addresses issues of *representation* which arise at and beyond this language-memory juncture, and the next chapter describes the interface more from the standpoint of how the information is *processed.*

## 3.1 WHAT NEEDS REPRESENTING IN MEMORY

In order to discover *how* to represent world knowledge in a conceptual memory, we first ask *what* needs to be represented. There is a clear need for being able to represent conceptual dependency graphs in conceptual memory. But there are other many other requirements which are logical extensions from CD into the domain of memory. The principal ones are the following:

1. *concepts and tokens of concepts*, like "John", "John's hat", "love", "the man who was here yesterday", "person"

2. *events (actions and states)*, like "John gave Mary a book" and "Bill is depressed"

3. *features of concepts and tokens*, like "John is a person", "the hat is red" and "the car is owned by Mary", "a butcher is a person who cuts meat for a living"

4. *features (conceptual modifiers) of actions and states*, like "John saw Mary AT THE BEACH" and "John was here YESTERDAY AT 5PM"

5. *conceptual patterns*, like "books are normally used for reading" and "John is generally at work on Tuesday morning" and "Mary likes red books". These comprise a knowledge of what is normal in the world.

6. *time information*, like "John was at the store for three hours", "Bill washed the car while Mary mowed the lawn", "before John came ... "

7. *dynamic processes*, for example inferences such as "if a person hits another, then he was probably mad at him" and "if a person wants an object, he is likely to go somewhere to acquire that object"

The first six categories are represented by "passive" data structures and will be discussed in this chapter. The last category represents an extremely large and important class of data structures which can be executed as LISP programs. These constitute the main core of the theory of information processing within the memory, and are the subjects of chapters 5, 6, and 7.

## 3.2 DESIGN CRITERIA

The first six categories suggest the need for two distinct types of entities to represent "passive" (non-procedural) knowledge of the world: (1) *objects*, and (2) *relations* among objects. Before we try to define data structures for these entities, it will be useful to put into focus some desirable attributes of any memory. There are six important principles to which we would like the conceptual memory to conform:

**referenceability** It should be possible to distinguish abstract concepts from *instances* of those concepts, and it should be possible to accomodate arbitrarily many instances of any concept. Every concept and instance of a concept which could conceivably be referenced from language (either by name, or by a description of its conceptual features) should be directly referenceable in the memory. Identical objects and notions in the world should be represented by the same entity in the memory.

**flexibility** It should be possible to store arbitrarily many conceptual features of an entity. There should be as few structural constraints as possible, and the conceptual features themselves should be separable, discrete and individually referenceable. It should be possible to store *features of relations* as well as features of simple objects. The introduction or learning of new features should be easy and should not upset the existing feature structure of an entity. It should be simple to create and link new entities into the memory, and to merge two entities together when the need arises.

**homogeneity** There should be as few "local" structural anomalies in the data structures as possible. Everything should in theory be representable within the same paradigm, even if some things are, in practice, stored in other ways for computational efficiency on a

computer. It should be easy to add to entities new fields, tags, etc. which would extend or improve the control structure of the memory as the theory evolves.

**retrievability** The memory should be a *connected* and *fully inverted* structure. We must first learn how to retrieve and manipulate information to which there is *perfect access* before attempting to model a less perfect memory. The memory should accomodate associative searches through propositional information, as well as associative retrieval of that information. It should be possible and convenient to locate an entity from a description of its conceptual features, and, conversely, to locate the entire feature set of an entity from the entity itself. All associations (links) should be referenceable and accessible entities which can eventually accomodate "degrading" functions associated with imperfect retrieval and forgetting. Information should not be "distributed", but rather centralized around the entities it describes.

**independence from language** There should be no reliance upon the words of any particular language. The names (if any) of an object should simply be conceptual features.

**psychological validity** The memory should conform to at least introspectively available evidence about how people seem to store and use information as it relates to language. There should be no strict requirement at first that the memory be an accurate analytical model of experimental psychological data, however.

The memory I will describe fulfills all these criteria for the most part. How it meets the last four criteria will become evident. However, the notion of referenceability is one of considerable theoretical importance, and deserves elaboration.

### 3.2.1 REPRESENTING KNOWLEDGE: PROBLEMS OF TOTAL REFERENCEABILITY

In devising data structures for storing conceptualizations in memory, one criterion seems to be far more significant than others. This one concerns referenceability: that every component detail of information associated with each conceptualization should be identifiable and referenceable as a discrete unit. That is, if people can talk about some part or aspect of a conceptualization, then that part must in some sense be separable from the conceptualization. What this seems to indicate is that all information must be reduced to very basic units, which can then be stored discretely and interrelated to form the larger thoughts. A very useful test for

discovering what is psychologically a basic unit of information is what I will call the "*fact that*" test.

Consider the sentence "Yesterday, Farmer John surrepticiously gave Mary a turkey for tax writeoff purposes." Among others, it is certainly possible to reference all the following information units within this conceptualization:

1. The fact that it was JOHN who gave Mary the turkey ...
2. The fact that it was a TURKEY that John gave Mary ...
3. The fact that John gave the turkey SURREPTICIOUSLY ...
4. The fact that it was MARY to whom John gave the turkey...
5. The fact that it was YESTERDAY that John gave Mary the turkey...
6. The fact that it was TAX CONSIDERATIONS which CAUSED John ...

Each restatement causes an important shift in emphasis which we should be able to capture. If we were to store the larger composite information units (actions and states) in some large, rigid vector notation such as

(<action> <actor> <object> <time> <location> <cause> <manner> <instrument> ... )

many of these smaller units would not be referenceable independently from the rest in the same way the entire vector is externally referenceable as a unit. They would be "buried"; their relation to the composite information-bearer would be implicit in their *position* in the vector, rather than explicit. Aside from the undesirable local anomaly in representation which would be required to reference "the fact that X is in position Y of vector Z", to store relations in long, comprehensive positional vectors presupposes we have decided upon all the slots. How could we ever be certain that, say, 27 slots could account for every aspect of any conceivable event, relative to varying contexts!

In addition, we will see in the next chapters how all the various aspects of a conceptually complex sentence must be able to stand alone in order to contribute independently to the processes of inferencing. For example, in suitable contexts, the fact that it was *yesterday* that farmer John gave Mary the turkey could overshadow all the other information conveyed by that sentence.

I conclude that a fixed vector representation lacks generality and is undesirable: (1) it is

unrealistic to believe that any one fixed vector notation would be flexible enough to account for all possible conceptual forms, (2) that information in the vector becomes isolated and not directly accessible for references.

It should be clear that using even "typed" associative links around a central event or state node as in Fig. 3-1 does not fully solve the problem of independent referenceability, although it exhibits the desirable looseness and flexibility of attaching features and aspects to an entity in the memory.

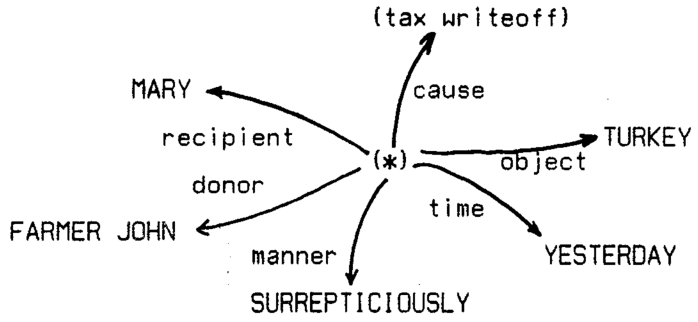



**Figure 3-1.**

The problem with referenceability still exists here because the links themselves are not referenceable as objects in the system: they serve in a higher capacity as relations. That is, a link is both an association and an implicit information-bearing relation: it predicates not only the existence of a relationship between two entities, it also specifies the *substance* of that relationship. It would be better to separate the notion of a link as a simple, *untyped* association (which is truly unreferenceable) from the notion of a link as an *information-bearing relation* (which can be referenced). This distinction is shown in Fig. 3-2 (the more desirable scheme is shown to the right). Although this distinction may seem quite esoteric, and have the appearance of splitting hairs, it is in fact a very important distinction.

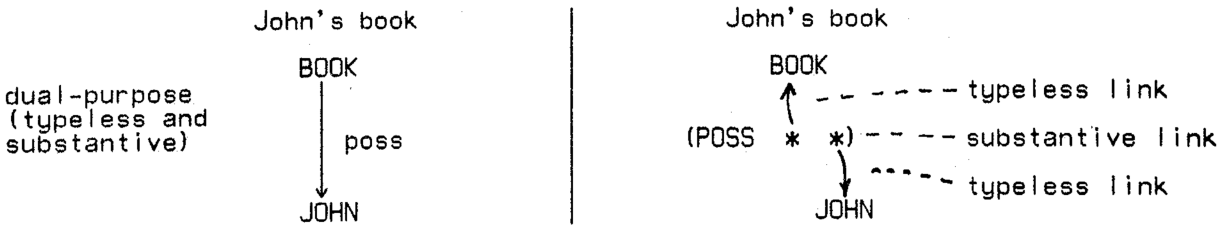



**Figure 3-2. Separating typeless links from substantive links.**

Thus, rather than represent all actions by some closed vector notation such as or even in some more general link scheme in which links serve the double purpose of denoting both an untyped association and an implicit relationship, I have tended to store all referenceable relations as separate units which are then associated with the units they relate, and with the larger information unit of which they are a part, by typeless associative links. Fig. 3-3 illustrates this technique for the Farmer John example. In the diagram, pound signs stand for referenceable units in the memory.

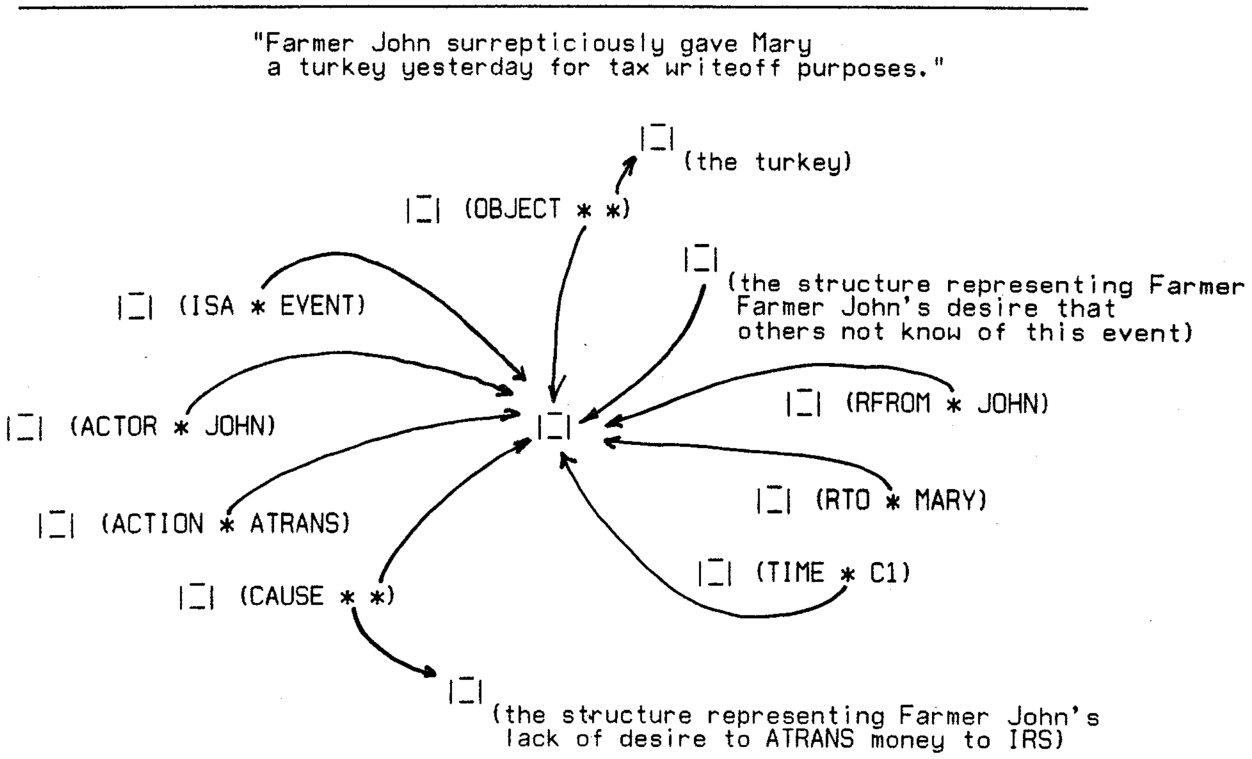



**Figure 3-3.**

In Fig. 3-3, C1 is some time token representing a point during "yesterday", RTO and RFROM representing the donor and recipient cases for ATRANS, respectively.

This results in a system which contains two basic types of objects: concept objects (this includes events and states), and information-bearing objects (those objects which relate concept objects and other information-bearing objects). That the information-bearing objects in the diagram are in fact "psychologically primitive" (in the sense we desire) can be verified by trying to apply the "the fact that" test to a few them.

While this characterizes the general philosophy for storing conceptual links in referenceable ways, as might be expected, reducing *everything* to this level is both cumbersome and inefficient in the realities of today's programming techniques. Furthermore, such "purity" is not necessary for the solution of many interesting problems of conceptual processing. As I describe the data

structures, deviations from the prescriptions of this section in the implemented program will be evident. Section 3.4.4 will summarize them.

## 3.3 CONCEPTS AND TOKENS

**The smallest units in a conceptual memory are CONCEPTS and TOKENS. What are these notions, how should they be represented, and how should they be organized?**

"Simple" objects were discussed as components of the Conceptual Dependency system of representation. There, the simple objects are "picture-producers" (PPs). A PP like "John" can produce a mental picture of a person named John, a PP "book" can cause the hearer to imagine a bound pile of paper which bears information, and so on. Recall, however, that there is no simple one-one correspondence between the words of a language and the PP's the language is capable of referencing.

In the memory, simple objects are *concepts* and *tokens*. These entities symbolically represent real objects and ideas in the world. What is their relation to PP's? Just as there is a lack of one-one correspondence between words and PPs, there is in general no one-one correspondence between PP's and objects in conceptual memory. A PP is an abstraction which stands for an entity with a certain set of features. But a potentially infinite number of real objects in the world can be categorized as instances of each PP.

For instance, the PP "JOHN" stands for any entity, X, which satisfies the abstract conceptual topology:

```
X is a person
X's name is "John"
X is of sex type male
```

However, there are many entities in the world which satisfy this topology: John Smith (the guy who lives down the road), John Smith (the butcher across town), John ("Ding Dong") Jones, the guy who ran for mayor last year, and so on. There must therefore be the potential for representing all these different Johns in the memory. There, any X which stands for the person John Smith in the real world is an example of a *token* of the *class concept*, "person". I will often refer to class concepts as simply *concepts*.

What should the X which represents, say John Smith, the guy who lives down the road, look like in the memory? Since it stands for a single entity about which many unique facts may be known, and which is unique itself, we want a unique entity in the memory to represent it. I have called the LISP construction which embodies this entity a *superatom*. A superatom is a discrete object to which we may point when referencing the entity for which it stands. But, in the absence of any defining conceptual information, a superatom is no more than a place-holder. That is, a superatom *to which no information is attached* is simply "something" if we must reference it by language. In the program, superatoms are just LISP atoms which arise via the LISP sequential symbol generator. Because of this, a superatom will often appear externally in the examples as something like "C3749".

All conceptual information *about* an entity is associated with that entity's superatom in the memory. This association is via the property called the *occurrence set* ("ASET" for historical reasons) of the entity. The occurrence set is a set of pointers to all conceptual information in the memory in which the superatom is involved. (The form of conceptual information in the memory is the topic of the next section. Suffice it to say here that every piece of information in the memory is also identifiable by a unique superatom.) This entity/occurrence set association can be viewed from two perspectives: it can be thought of either as the defining set of features for the entity ("feature" here meaning *any* conceptual information known about the entity), or as a set of pointers to all other points in the memory where the entity occurs. There is of course no material difference between these characterizations.

> **The occurrence set for a concept is therefore a catalog of everything known about that concept; it is a bundle of conceptual features. The superatom IS the concept, but the occurrence set defines its essence.**

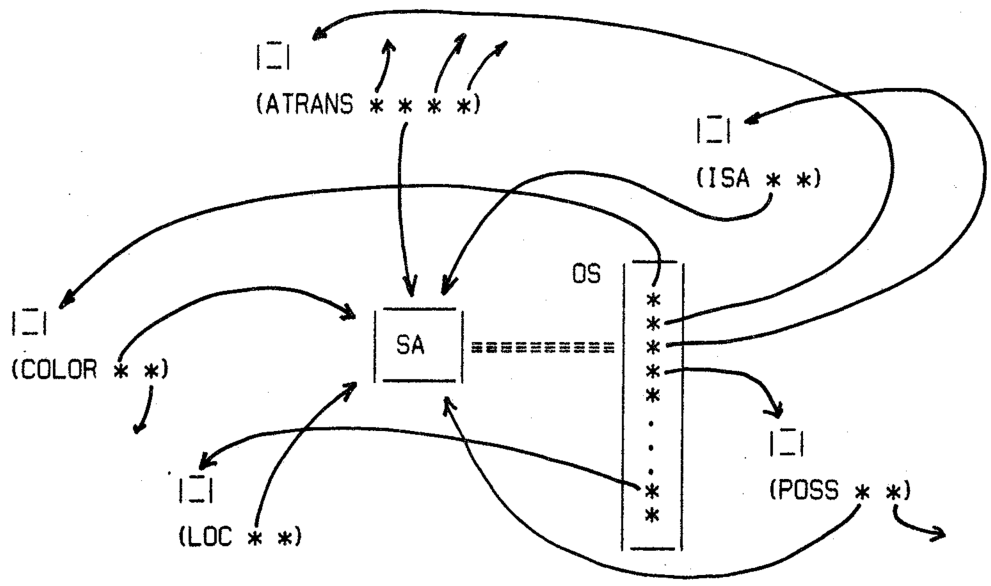



**Figure 3-4. The relationship of a superatom, SA, to its occurrence set, OS, and to its conceptual features.**

Fig. 3-4 schematically illustrates this data structure for simple objects in the memory. Stored in this kind of structure, our friend John Smith, the butcher who lives down the street, and who, among other things, possesses a car, loves Mary Jones, and was at the grocery at 5pm yesterday, is represented by the superatom, say C0431, and occurrence set illustrated in Fig. 3-5. The specific superatoms there were of course arbitrarily chosen for the purposes of illustration.

```
C0431

ASET: C3726          C3726: (ISA C0431 #PERSON)
      C0213          C0213: (NAME C0431 JOHN)
      C9771          C9771: (SEX C0431 #MALE)
      C7823          C7823: (SURNAME C0431 SMITH)
      C3254          C3254: (RESIDENCE C0431 C5613)    (C5613 is where he lives)
      C0003          C0003: (POSS C0823 C0431)         (C0823 is a car)
      C6541          C6541: (MFEEL C0431 #LOVE C0017)  (C0017 is Mary)
      C2188          C2188: (LOC C0431 C1792)          (C1792 is the grocery)
      C7437          C7437: (PROFESSION C0431 #BUTCHER)
       ...
```

**Figure 3-5. Part of the occurrence set for some John Smith the memory might know.**

Not shown in Fig. 3-5, the time of C2188 was furthermore 5pm yesterday:

(TIME C2188 C3214)

where C3214 is a time token representing this time. Also, C5613 is at a location which is "down the street":

(LOC C5613 C2819)

where C2819 is whatever this location actually happens to be. ("Down the street" is not a conceptual relation, but rather is simply one way of expressing John's location relative to our own.)

It should be clear that this superatom-occurrence set structure is fully-inverted. That is, it is possible both to locate an entity from any conceptual information which involves that entity, and to retrieve all conceptual information about an entity starting from the entity itself. Furthermore, all "links" are (a) untyped and (b) explicit. They are untyped because links merely serve to tie together an entity with its defining conceptual information. The *substance* of that information does not exist in the link, but in the conceptual information it points to. Links are explicit because each link is an identifiable object in an occurrence set.

### 3.3.1 COMMENTS ON NOTATION

I have been using, and will make further use of the notation:

# <letter>+ <digit>*

that is, a "pound sign", followed by a word, possibly followed by some digits. (#JOHN3, #LOVE, #PERSON). This notation stands for a superatom in MEMORY, and is no different from superatoms looking like "C1373". As we have seen, a superatom does nothing more than give us a way to point at collection of conceptual features. This notation just allows us to identify some concept or token when we need to talk about it without enumerating its feature set every time. Thus, the form of the symbol #JOHN1 is not a concern of any memory process, and might just as well be stored and accessed in the memory as, say, C4893, a concept among whose features might be found:

```
(NAME C4893 "JOHN")
(SURNAME C4893 "SMITH")
(ISA C4893 #PERSON)
(POSS C3825 C4893)
   ...
```

(where C3825 is, say, a token of a car). Often, when illustrating memory structures graphically, I will write a pound sign to stand for some superatom, then enumerate a set of features which describes it. But bear in mind that, although I am listing the defining conceptual information explicitly, all that is stored with the concept in memory is a set of pointers to other superatoms at which such information about the entity is stored.

### 3.3.2 THE LOGICAL ORGANIZATION: TWO IMPORTANT RELATIONS

The previous chapter characterized the nature of useful predicates in the conceptual domain. However, two relations are more important than most because they bear directly (a) on the logical organization of the memory, and (b) on a significant aspect of the memory-language interface. These are NAME and ISA, which relate an entity to language, and to the rest of the memory's internal taxonomy of concepts, respectively.

#### 3.3.2.1 "NAME"

Any concept or token can have a NAME feature. NAME is the principal means of interface between internal concepts and tokens and the words of one (or more) language, and a concept or token need not have any NAME, or it may have one, or many. Conversely, objects in the memory which are NAMEs of concepts and tokens may serve to name more than one concept or token (senses of a word, instances of class concepts).

In a "pure" system, names would be #WORD concepts whose conceptual values are the strings of letters (or more correctly, morphemes) which comprise the word. We have no use for this level of detail of information, so the structure has been "cauterized" at a slightly higher level: the second argument of the NAME predicate simply points to an "ordinary" atom, which is like a superatom except that its LISP print name is significant. Had names been specified "to the edge of the model", the type of construction shown in Fig. 3-6 would have arisen.

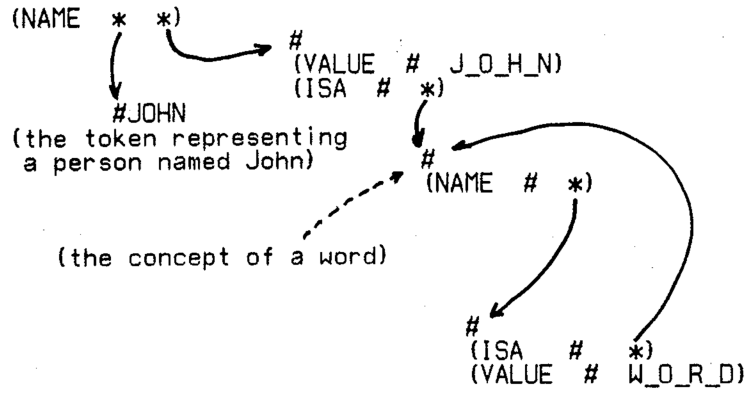



**Figure 3-6.**

To illustrate how a word concept in the memory relates to its senses (concepts) and, further, to tokens of concepts, consider the NAME structures which might surround the atom "BILL". Fig. 3-7 depicts how word concept "BILL"'s occurrence set might look in memory.

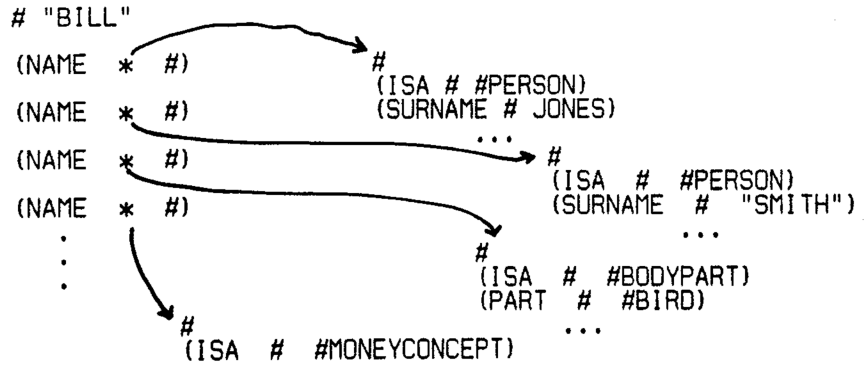



**Figure 3-7.**

A very fundamental reliance on some sort of recency criterion (as well as the processes of inference), is required to keep track of the most likely senses of words at any given time. [R2] discusses the notion of word sense promotion in considerably more detail.

3.3.2.2 "ISA"

The conceptual predicate ISA relates tokens to the abstract concepts (which are in turn just superatoms with bundles of conceptual features) of which they are instances, and also relates abstract concepts (like "person", "guilt") heirarchically. A concept or token can have no more than one ISA feature. The interpretation of (ISA X Y) is that, in addition to all the features on X's occurrence set (besides the ISA relation), X also has all the features of on Y's occurrence set. In Fig. 3-5 for instance, this means that C0431 has, in addition to those features explicitly on its occurrence set (C3726 ...), the multitude of implicit conceptual features associated with the superatom #PERSON, since C0431 is an instance of a person. Likewise, #PERSON has in addition to its occurrence set all features of #ANIMAL, which is *its* superset, and so on. (A common, but fairly trivial example of this concerns NAMEs of tokens. More often than not, there will be no NAME associated with a token: its NAME is normally stored as a feature of the idealized concept of which it is an instance. Hence, in order to *express* a given token in language, the memory must frequently ascend one or more levels until a name is found. Then it must locate some distinctive features of the token to distinguish it from other tokens of the same concept. It can then use those features, expressed by relative clauses and adjectives in the sentential expression. An example would be: "the red ball which John had ...".) In general, all processes in the memory which ask "does X have conceptual feature Y?" must be prepared to ascend X's ISA set sequence in search for feature Y.

When X is "almost a Y", except for feature Z ("an ostrich is a a bird, except that it can't fly"), we write (ISA X Y) and (LACKS X Z), Z being a pointer to the conceptual feature of Y which is not a feature of X. Thus, the "total feature set" for an entity, X, consists of everything on X's occurrence set, everything on Y's occurrence set, where (ISA X Y), and so on, except for those features for which a LACKS relation exists.

I should say more about intent of the ISA relation in this theory. We want to take special care not to "overspecify" one concept by heirarchically (ISA) associating it with too-specific another concept. The ISA relation should be reserved for associations between a token and its "least biased" classification. For instance, to characterize John, the butcher, by (ISA #JOHN #BUTCHER) would be an overspecification, since it places a special focus, or interpretation, on John which is not of *general* utility or interest. Rather, John always ISA #PERSON, and if he

happens to cut meat for a living, we should write something like (PROFESSION #JOHN X), where the X points to the bundle of features which define the essence of a butcher. The point is, John's profession is only a very small characterization of a man who might also happen to be a father, a good golfer, a rabid political right-winger, and so on. To characterize him as any one of these is to introduce a bias which could make it hard to interpret him differently in different circumstances.

Also, classifications which overspecify tend to oversimplify. That is, we call John a rabid right-winger because of the things he does and says; but we might also call our friend Bill a right-winger, even though he does and says completely different things from John. To say that (ISA #JOHN #RIGHT-WINGER) and (ISA #BILL #RIGHT-WINGER) is to predicate that they both have the features of this abstract class. Yet there may be have nothing at all in common in the details of what they do. On the other hand, they are both #PERSONs whose individual actions and beliefs can be contrasted on a one-one basis. Although the same *label* might evolve for both in our model, they are still complex people who can be interpreted quite differently outside the political domain.

### 3.3.3 STRUCTURAL PROPERTIES OF CONCEPTS AND TOKENS

It would seem that there are certain aspects of concepts, tokens and information-bearing units in a human language user's memory over which he has no direct control. These are things which are more closely related to the *mechanisms* of the brain than to the *data* the brain stores.

By attaching to a superatom other properties besides its occurrence set, it is possible to associate aribitrarily much information (of other types than conceptual) with each entity in the memory. Although the occurrence set defines all of an entity's *conceptual* features in the memory, other properties are useful for associating certain other information with superatoms for other memory functions "above" the conceptual data structures.

The "recency of activation" of an entity (reference to it, either by language directly or by some internal thought process) is an example of a property which would seem to be more related to a mechanism than to the substance of the entity itself. It would seem proper to view aspects such as this as part of the brain's "wetware": they are part of its unconscious control structure rather than part of the information this structure stores and operates upon.

Of course, no one yet knows exactly what the processes of the control structure are, much less which of them can be though of as involving "tags" on entities in memory. Nevertheless, use has been found for three structural properties which are related to language understanding:

1. RECENCY
2. TOUCHED
3. SEARCHTAG

These are stored as LISP properties of superatoms.

RECENCY keeps a record of the time each concept or token was last legitimately accessed by the reference mechanism. "Legitimate" means that an explicit decision was made that the concept or token was the referent of some language construction ("John Smith", "the dog with three legs", "love", "the second time we were in the meadow", etc.), rather than simply "passing over" the entity while searching for another one. By use of this tag, many potential problems of ambiguous reference can be avoided or solved. As we will see, the reference mechanism prefers the most recently accessed candidate for a reference in cases where there is a significant difference in recencies among the candidates, or where inference fails to solve the problem. RECENCY plays the same role for references to events ("the time we were in King City").

TOUCHED is also a recency tag, but records the time an object was last "touched" or drawn into, the processing by internal processes (inferencing), as opposed to having been referenced directly from language. We will see later how implicit references of this sort can be vital to understanding. As that section will illustrate, the set of objects in MEMORY with recent RECENCY and TOUCHED tags captures the Conceptual Dependency notion of *immediate memory*.

SEARCHTAG is of less theoretical utility than the other two tags. It simply provides a foothold for associative searches through MEMORY.

#### 3.3.3.1 TWO STRUCTURAL PROPERTIES RELATED TO THE CONCEPTUAL ANALYZER AND GENERATOR

Two additional properties, XFORM and CASES, are associated with concepts which are conceptual predicates (for example, ATRANS, MFEEL, POSCHANGE, ISA, NAME) in memory.

For a predicate concept, P, XFORM stores the Conceptual Dependency structural template which will express (in CD, not language!) memory structures which use P. This is purely a

transformation of *form*; it is the beginning of the memory-generator interface which allows any information-bearing structure in the memory to be assembled into a Conceptual Dependency graph for expression in language. Several examples of the XFORM property are shown in Fig. 3-8.

| | |
|---|---|
| *ATRANS* | ((ACTOR X1 <=> (*ATRANS*) OBJECT X2 FROM X3 TO X4)) |
| ISA | ((ACTOR X1 <≡> (*CLASS* VAL X2))) |
| CAUSE | ((CON X1 <≡ X2)) |
| BIGPOSCHANGE | ((ACTOR X1 <≡>F X2 <≡>T X2) INC (4)) |
| WANT | ≡ EXPRESS_WANT<br>((CON ((CON X2 <≡C ((ACTOR X1 <≡>F (*JOY*) <≡>T (*JOY*))<br>INC (2) TIME (T1))))<br><≡> (*MLOC* VAL (*LTM* PART X1 REF (*THE*))))) |
| *HEALTH* | ((ACTOR X1 <≡> (*HEALTH* VAL X2))) |

**Figure 3-8. XFORM templates.**

In the templates of Fig. 3-8, Xi is interpreted as argument i in the memory bond notation, ≡ indicates that a special function is to be applied to the template after it has been instantiated to perform special details of the transformation which are not conveniently notated in the passive template. (In the WANT template, for instance, this amounts to correct location of the times which are internal to the template.)

CASES stores a similar, structural transformation template at the *analyzer*-memory interface, and is a property only of primiticve ACTs. For ACT A, CASES stores the list of the nuclear conceptual cases for A, in the order in which they appear in memory bonds. For example, the ATRANS CASES property is (ACTOR OBJECT FROM TO), and the CASES property for GRASP is (ACTOR OBJECT). Section 4.5 describes how this information is used by the process which converts analyzed conceptual graphs into internal memory structures.

#### 3.3.3.2 THREE OTHER INFERENCE-RELATED PROPERTIES OF CONCEPTUAL PREDICATES

There has been no discussion yet of the inference mechanism and other active processes in the memory. However, it should be noted here that the organization of inferences hinges about conceptual predicates, and this involves the potential association of three LISP program modules

with each conceptual predicate. These associations occur via three structural properties of predicates' superatoms: "IPROG" for inference molecules, "SPROG" for specifier molecules, and "NPROG" for normality molecules. These will be defined later.

## 3.4 STORING CONCEPTUAL INFORMATION

**How should conceptual propositions be stored and organized? How should they interface with concepts and tokens?**

The story is, of course, not yet complete: I have yet to describe how *relations* among objects (conceptual information) are stored. Not much more than the structure for concepts and tokens is required, for we can view a unit of conceptual information as an object in the system in much the same way as we view a concept or token. That is, any information (a feature, action, state, etc.) can itself have an arbitrary number of conceptual "features": time aspects, who knows about it, what caused it, what it caused, what its location was and so forth. Viewing units of information as objects is convenient also from the standpoint of language, since all but the simplest utterances involve nested conceptualizations: one or more "sub"-conceptualizations can be referenced by the main one. From this standpoint, a feature of each sub-conceptualization is that it occurred in the context of the main conceptualization.

The main difference between information-bearing objects and simple objects concerning storage requirements is the obvious one: in addition to serving as a place-holder, with which arbitrarily many conceptual features can be associated via its occurrence set, an information-bearing entity must carry some intrinsic information.

### 3.4.1 BONDS

Conceptualizations are therefore stored as superatoms, replete with occurrence set and the RECENCY, TOUCHED and SEARCHTAG properties described for concepts and tokens. Their information content, a *bond*, is associated with their superatom under the LISP property "BONDVALUE".

Bonds are positional lists which relate other conceptualizations, concepts and tokens. The first member of a bond list is always the conceptual predicate (action, state, causal, etc.).

Predicates are simply concepts which bear special meaning to the processes which operate on them, and are stored just as any other concept. In the current implementation their occurrence set typically consists of just an ISA relation such as (ISA *MTRANS* #ACTIONPRED). Bonds which represent actions consist of the nuclear cases of the action, always in the order ACTOR, OBJECT, FROM, TO. State and causal relations are stored in the obvious ways as illustrated throughout the previous chapter.

As an illustration of bonds, consider the utterance "John believes that Bill sold his car". The left of Fig. 3-9 shows the internal memory structure which would result from this utterance, the right illustrates the structure graphically. Time aspects have been omitted, and the superatom numbers were of course chosen arbitrarily.

```
C3764
  BONDVALUE: (MLOC C3765 C0018)
  ASET: null

C3765
  BONDVALUE: (DUALCAUSE C3766 C3767)
  ASET: (C3764)

C3766
  BONDVALUE: (ATRANS C0021 C7641 C0021 C0027)
  ASET: (C3765)

C3767
  BONDVALUE: (ATRANS C0027 C5321 C0027 C0021)
  ASET: (C3765)
```

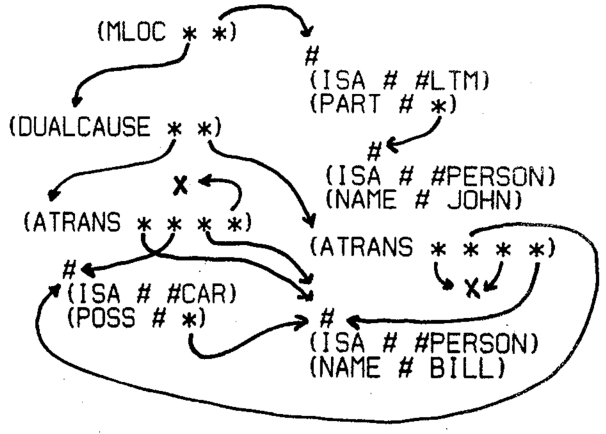



**Figure 3-9.**

In Fig. 3-9, C0021 is the token which represents the LTM (long-term memory) of the John who believes this, C0021 is Bill, C0027 is the (unspecified) person to whom Bill sold his car, C7641 is the car and C5321 is the money which was exchanged for it.

Information-bearing superatoms (those which store bonds) also have the structural properties RECENCY, TOUCHED and SEARCHTAG. But in addition to these, they require some additional structural properties. For instance, does the memory *believe* C3765 in Fig. 3-9? This leads us to two notions about bonds which do not exist for concepts and tokens.

### 3.4.2 STRUCTURAL INFORMATION ASSOCIATED WITH BONDS: CONTEXTUAL TRUTH

What information the memory is *capable of representing* and what it actually *believes* at any given time are quite different issues. There must an effective and efficient means of distinguishing them. Otherwise, for example, we could tell it that John believes Bill sold his car, then ask it whether Bill sold his car and receive an unmitigated affirmative response, even though the car may not in fact have been sold. (Of course, the memory *could* decide to believe this also, as well it might, but this is not the issue here.) Even more absurdly, we could tell it "John couldn't go home", then ask it whether John went home. Finding the structure (PTRANS JOHN JOHN L HOME), a simple-minded memory would blithely reply "Yes", doing so simply because it had not paid attention to the surrounding structure, in this case, (CANNOT (PTRANS ...))! "Surrounding structure" here means the conceptualization's occurrence set.

The problem, then, is keeping a record (or at least being able to reconstruct one) of what the memory holds to be true. It would clearly be possible for the memory, by applying special heuristics to enough surrounding structure, to decide of a unit, X, whether or not X is held to be true. For instance, even though it is possible to find a (PTRANS JOHN JOHN L HOME) bond, if it occurs in some mitigating environment such as (CANNOT (PTRANS ...)) then it is certainly not to be believed, *except in that environment.* This is an obvious, but important observation, and I will call it the *principle of contextual truth.* This principle says that the information in the bond of an information-bearing superatom cannot be assumed to bear truth of its own. Rather, is true *only in the context of its entire occurrence set.* Furthermore, there exist effective procedures which can decide whether an information unit is believed on the basis of its occurrence set.

In practice, to have to make such decisions too frequently would be timewise-unwieldy. One of the lowest level functions of the memory is to locate units of information which are true in the memory's model. There would be no time for higher level functions if, each time MEMORY located information, it had to decide again whether or not it was believed. More important, this would seem to be totally counterintuitive to the way people seem to store information. Some level of automatic assessment of the believability of each new information unit as it is stored would seem to be the rule rather than the exception in human language users. Even information whose truth has not yet been decided is at least "tagged" as such. (There is one important exception to this, and it concerns time. In particular, the temporal truth of an uncompleted state or protracted

action is subject to reevaluation each time its truth is important to an inference or other process. This is necessary since many states come to an end after "fuzzy", but predictable, periods of time, and is discussed in section 6.8 as a form of inference.)

### 3.4.2.1 THE PROPERTIES "TRUTH" AND "STRENGTH"

To give this principle of contextual truth some tangible and efficient realization in MEMORY, two additional properties are associated with each superatom which stores a bond: TRUTH and STRENGTH. The former stores the value TRUE or FALSE and indicates whether the bond (in the context of its occurrence set) bears any truth in the model, or whether it is simply a non-truth-bearing part of a larger structure. At any time then, only those bonds bearing a TRUTH=TRUE constitute what the memory itself believes (to some degree or another). As we will see, one byproduct of inferencing is to change TRUTH and STRENGTH markers, so that, if John says he believes something and the memory's characterization of John is that he is trustworthy and not given to hallucinations, the memory can start believing John's belief too simply by tagging it as true.

STRENGTH indicates the approximate degree to which a bond which is believed; hence, TRUTH and STRENGTH are not independent. The strength is a real number in the interval 0 to 1. By convention, a strength of X below 0.5 means that there is reason to believe that X is *possible*, but *not likely*. (For some predicates, this can often be interpreted to mean that X's negation is believed to be likely.) Inference molecules propagate strengths from antecedants to the inference they generate by individual heuristics within each inference in the molecule. In the current model, the propagation occurs by simple multiplicative factors.

The whole notion of strength is a fuzzy one: the numbers themselves mean very little. Instead, their significance manifests itself in the *effects* the numbers have on the inference process: as long as their effects are appropriate, the numbers themselves are inconsequential. This is a point where representation and the process which operates on it are truly inseparable. The two conceptualizations, C1 and C2, below illustrate this quite poignantly: even though C1 and C2 below may both have the same low strength of 0.10, the interpretation of the strength relative to the *substance* of each conceptualization makes one quite significant, the other much less so:

C1: Mary's husband is cheating on her.

C2: John wants to buy the car.

Although its likelihood is the same as C2's, in C1, even the *suspicion* of such dastardly behaviour is quite significant. C2 has much less flare: it's just not very likely that John wants to buy the car, and that's that.

It should be reiterated here that, beyond this convention of low strengths, problems of negation have not been explored in any depth in the memory.

### 3.4.3 PRESERVING CONNECTIVITY IN INFERENCE SPACE: "REASONS" AND "OFFSPRING"

People not only can solve problems, they are aware of, and can describe *how* they solve them. A restatement of this phenomenon is to say that a person is (subconsciously) capable of preserving information about *why* he believes each piece of information in his memory. By doing this, everthing has an explicit internal justification: an implicit dependence relation with the antecedants from which it arose. This information can be used not only to answer questions like "What makes you believe that?", but it also provides a means of predicting and propagating the effects of altering the truth or strength of some information in the memory: if information X played a part in generating Y, and X's credibility falls under serious doubt, then so might Y's. In addition, in sections 5.6 and 6.6, it will be shown how REASONS and OFFSPRING are essential to at least two very vital classes of conceptual inference.

In the memory this means that, in addition to *conceptual* connectivity among information through bonds and occurrence sets, some sort of connectivity in inference space is essential. That is, whenever X is inferred from Y1,...,Yn, we should make this dependence explicit. To implement this, two other structural properties are associated with every information-bearing superatom in MEMORY: "REASON"s and "OFFSPRING".

These two properties are inverses of each another. The REASONS property for information-bearing superatom, X, is a list of other information-bearing superatoms in the memory which some inference molecule used in order to generate X, its inference. A null reason list for X means simply that X is believed: there are no reasons for the belief. In a more formal system, we might call this an axiom.

The property OFFSPRING for X is a list of other information-bearing superatoms in memory whose existence in some way relies on X. It is in a sense the inverse of REASONs. While every bond with TRUTH=TRUE must, by convention, have a REASONs list, the OFFSPRING property is of course not a requirement.

Inference molecules, which are the fairly complex LISP programs which make inferences, are responsible for supplying a REASONS list to the inference monitor along with each inference as it is generated. The following scenerios illustrate typical usages of the REASON list:

> John has just asked Bill for the New York Times on the table. MEMORY infers that it is likely that John wants to read it. This inference is generated, and the reason list consists of the following three units of information: (1) the NY Times is a newspaper, (2) a newspaper is printed material, and (3) the normal function of printed material is that it be read. Section 6.1 illustrates a similar example and includes a computer example

> Mary hated John. Bill hit John. The memory infers that it could be likely at that point that Mary feels a liking for Bill. This inference is generated, and its reasons are (1) Mary feels a negative emotion toward John, (2) John suffered a negative change, (3) it was by Bill's action that John's negative change occurred.

#### 3.4.3.1 REASONS AND OFFSPRING VS. CONCEPTUAL CAUSALITY

REASONS and OFFSPRING should not be confused with the CAUSE relation. The CAUSE relation is part of the data which the memory stores, has access to, manipulates, and uses to generate inferences. A CAUSE relation is stored in a bond, and hence can have REASONs and OFFSPRING. Section 5.5 discusses how and why causal relations are maintained by CAUSE bonds in bond-occurrence set framework. REASONS and OFFSPRING relate to the supervisory functioning of the memory.

### 3.4.4 DEVIATIONS FROM THE "PURE" REPRESENTATION

It should be clear that the theory of representation as proposed in section 3.2 has served

as the guiding doctrine, if not dogma, of the implemented program. That is, with only a few exceptions (motivated by the pragmatics of implementing a large system), every information-bearer which is potentially referenceable from language is indeed referenceable as a discrete object in the memory. The noteworthy exceptions are threefold, and we have seen two of them:

First, conceptualizations are stored in the form of positional lists which only implicitly specify the case relation of the concepts to the action or state predicate. Hence, it is not possible to reference information units like "the fact that X is the actor of action Y" or "the fact that X is the recipient in action Y". This type of unreferenceability is, however, restricted to a very small, well-defined, domain: namely the nuclear case relations for actions and the "nuclear" arguments of states relations.

Second, there are certain counts, tags and relationships maintained for conceptualizations which are unreferenceable as units in the system. The rationale for such a auxiliary constructs should be clear. Although these features *could* be framed in the main data structure, it is more intuitively correct to keep them separate.

The third main exception will become evident when we discuss the nature of inference, specifier, and normality molecules and atoms in chapters 5-7 (these store the active inference processes in memory): while these objects are discrete entities, referenceable as entire units, their internal components (their conceptual contents) are not individually referenceable. Section 7.3.4 discusses this problem and a potential solution to it.

Each of these exceptions could have a direct representation in a "pure" system in which we had the luxury of unlimited amounts of storage. The fact is, the amount of space consumed to make these units of information referenceable is simply not justifiable in terms of the number of new capabilities they enable. Although certain basic processing assumptions have been based on these "impure" forms, their conversion to operate on pure forms is easily imagineable.

## 3.5 SUMMARY OF MEMORY DATA STRUCTURES

We are now in a position to formulate concise structural definitions for the memory structures I have been describing.

**DEF:** A "concept", C, is a LISP atom which has the property ASET (*occurrence set*) and which does not have the property BONDVALUE. A "token" is a concept, X, such that there is no relation (ISA Y X) in the memory.

The value associated with the property ASET of C is a list of superatoms which store bonds $\{\beta 1, \ldots, \beta n\}$ such that C occurs in $\beta i$, i=1,...,n, and only in those bonds in the memory.

**DEF:** An information bearing unit, U, is a LISP atom which has the properties ASET and BONDVALUE.

The ASET is the same as for concepts. The value associated with the property BONDVALUE of U is a list of atoms (other concepts and bonds) of the form (P X1 X2 ... Xn), where P is some predicate and X1,...,Xn are its conceptual arguments (nuclear cases for primitive action predicates).

In addition, information-bearing superatoms in the memory have the following auxiliary properties: RECENCY, TOUCHED, SEARCHTAG, TRUTH, STRENGTH, REASONS and OFFSPRING. Concept and token superatoms have only the first three. Also, each concept which is a conceptual predicate in the memory has five additional properties: CASES (ACTs only) which specifies the case mapping from graph slots internal bond positions XFORM which is a template specifying the Conceptual Dependency structure which will express in CD format any memory structure which involves that predicate, and IPROG, SPROG, NPROG, which store inference-related LISP program modules associated with that conceptual predicate.

## 3.6 REPRESENTING AND STORING TIME

**How should time information be represented in conceptual memory? What time entities and relations are needed?**

There are three general approaches to the problem of how to represent and manipulate

time in a model: (1) ignore it, (2) make it an implicit part of the control structure (say, by partitioning memory on the basis of time, or by carrying along "states of the world"), or (3) make time an explicit aspect of all data, representing it in the same general structures as everything else. I have chosen the third approach because it appears to be potentially the most general.

Any model whose main goal is to cope with natural language and what that language communicates must be prepared to deal in considerable detail with time. This includes keeping track of times of events and states, maintaining relationships between the times of events and states, and supplying proof procedures to interrelate the various time predicates of the model, and sensitizing to time aspects all conceptual inferences for which time is a critical dimension. Furthermore, it is an important realization that much of what we would term our "knowledge of normality in the world" bears heavily on the maintenance of time relations in the system. We will later see a type of conceptual inference which is wholly concerned with the maintenance of time relations in the memory. It is the purpose of *this* section to describe the philosophy of the memory's sensitivity to time from the standpoint of representation. This involves describing what time predicates and relations there are, and how they are used in the model.

While the memory is not strictly an analytical psychological model, I have emphasized that a reflection of psychological intuition in the model is highly desirable. For some limited tasks, it might be adequate to maintain a "state of the world" type data base where every unit of information in the memory, and only those units, is viewed as true at the present time. But this is simply not the total picture of the way people (successfully) deal with time. The major inadequacy in a model which handles time this way is that once information about the modeled world becomes false or is superceded by a newer piece of information, the old information is forever lost. In a model for which there is a well-defined task to be accomplished, and this task is sensitive only to the current state of the world (say, a factual question-answering system, or a model of traffic flow) this approach works nicely. The problem of understanding the conceptual content of natural language utterances is not such a simple domain. There must be ways to distinguish past, present and future not only because they are commonly refelected in language, but because most inferences are sensitive, in varying degrees, to time. Also, when the memory needs to communicate with the outside world via the conceptual generator, it must be able to express detailed tense information. This information must somehow be deriveable from time structures in the memory.

At the other end of the spectrum from "state of the world" models is the approach whose tenet is the following: *remember everything.* That is, propositions should not be thrown away simply because they become out of date or irrelevant. This is not to say that a *real* (human) memory does, or should remember everything. There are important psychological, and above all practical, arguments against such a claim. However, the criteria for removing a proposition from memory are quite a bit more complex than simple truth or falsity at a given time of the world. It is not my intent to discuss in any depth the problem of forgetting. Becker [B1] has some interesting ideas on this subject.

I have taken this alternative approach by having the model *remember everything.* A forgetting function is viewed simply as an addition to the system at a higher level. To claim that the lack of forgetting or retrieval-degrading functions has any bearing on the algorithms which maintain time is ludicrous: there must still be processes which are capable of working on whatever information *is* available: in particular, *perfect* information.

### 3.6.1 TIME TOKENS AND RELATIONS: THE REQUIREMENTS

So much for the philosophy; What do we need to do the job? There must clearly be *time tokens* and *time relations.* A time token represents either a point on the model's absolute time scale, or a duration on this scale. (In the current model, this scale is the number of milliseconds the LISP core image has run. It would more appropriately be the number of seconds since some starting date.) A time point is simply a token, X, whose occurrence set contains an (ISA X #TIME), and probably some other time relations which I will describe. A time duration is a concept, X, whose occurrence set contains an (ISA X #DURATION) and a specification of its length, N, in scale units: (TVAL X N).

To illustrate what kinds of time relations the memory must record, let's examine an extremely simple story. Consider the following two-liner:

John had a book.
He gave the book to Ellen.

Assume the time of utterance of the first line is #NOW, that is, the present. For the purposes of

illustration, I will use #NOW as though it were a point in time -- a time token. In reality, each time the conceptual analyzer completes a conceptual graph and sends it to MEMORY, a special function is evaluated which creates a new time token with which it then associates the current numerical value of the model's clock. For LISP reasons, this number is stored as a structural feature of the token under the property TVAL instead of simply adding another conceptual property (VALUE X <number>) to the time token, X's, occurrence set. By creating a time token which uniquely identifies the time of utterance, all other time references in the utterance have a concrete and unique pivot.

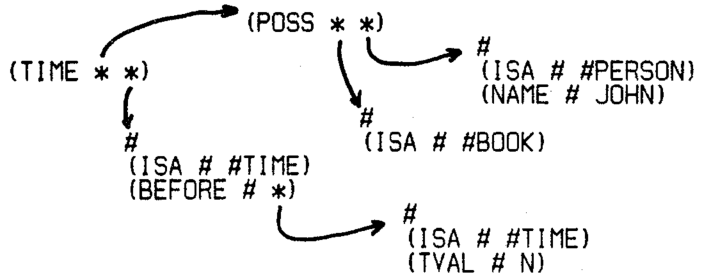
In this story, the first sentence predicates that it was true at some point in the past of #NOW that John possessed a book. The conceptual graph MEMORY receives from the analyzer has the form shown in Fig. 3-10.

```
((ACTOR (BOOK) <=> (POSS) VAL (JOHN)) TIME (TIM01))

TIM00: NOW
TIM01: (BEFORE TIM00 NIL)
```

**Figure 3-10. "John had a book."**

The memory will create a POSS bond to stand for the state, a token to represent the point in time at which the state is being predicated to be true, then will add all known relations about this time point to its token's occurrence set. In this case, the only known relation is that it was *before* another time token -- the one created to represent #NOW, the time of utterance. Since time is a conceptual requirement for any event or state which occurs in the world, by convention, any proposition in MEMORY stored with no time predications is a *timeless* statement -- a belief about the world which is invariant with time.

The internal structure which results from the analyzed descriptive form is depicted in Fig. 3-11.




N is the numerical "now" on MEMORY's internal time scale.

**Figure 3-11. "John had a book", internally.**

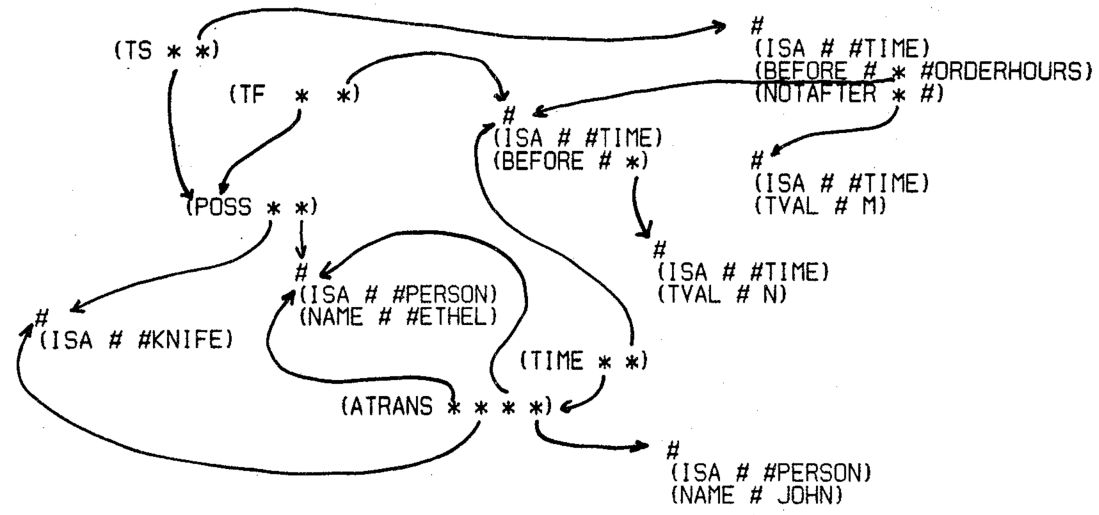
The second line of the story is analyzed into the following descriptive unit:

```
((ACTOR (JOHN) <=> (ATRANS) OBJECT (BOOK REF (THE)) FROM (JOHN) TO (ELLEN))
 TIME (TIM01))

TIM00: NOW
TIM01: (BEFORE TIM00 NIL)
```

Assume the references to the correct book and John are found, and that this information has been processed into internal form in the same way the first line was. Also assume that, among others, three inferences subsequently made are (1) that John in fact had the book at the time he gave it, (2) that after he gave it, he no longer had it, but rather (3) Mary started having it. These three new facts (the second sentence, and two inferences which arise from it) augment the existing structure as shown in Fig. 3-12. Notice that the information communicated by the first sentence remains.

### 3.6.4 TIME EXAMPLE

The following example will serve to tie together these simple ideas about time: "Ethel had a knife for several hours yesterday before she gave it to John." This sentence results in the structure shown in Fig. 3-13. Again, only enough occurrence set to distinguish each token is shown there.




M, N are the absolute times delimiting yesterday, a duration.

**Figure 3-13.**

## 3.7 COMMENTS ABOUT THE MEMORY

It is probably accurate to say that this memory is a paltry fraction of what will ultimately be required comprehend and use language. And, although it satisfies the six criteria laid out in section 3.2, these criteria are merely the ones which seem important today, relative to specific tasks of understanding language, especially with regard to conceptual inference. I have had to ignore many important issues, and to idealize and simplify many others just to get started. But the memory structures I have defined appear to be simple enough to accomodate most any future extensions: some sort of homogeneity indeed has been achieved. All that would seem to be required for new information forms are the conventions for their storage.

For instance, there are well-defined places to store structural information and "real" information. If, as an example, the notion of negation is found to be more fundamental than most notions, it can be elevated to the structural level, where its presence can exert more direct influence on the memory's control structure. If direct word-word or concept-concept free-associations (as contrasted with only associations *through conceptual information*) are found to be vital to understanding (as undoubtedly they will), there are obvious ways of implementing them in the memory. If "reasoning by analogy" is to be pursued, there are also some obvious ways to approach it in a graph memory such as this.

It has not been a primary objective of this research to implement efficient associative retrieval algorithms. Rather, the emphasis has been more on issues of logical organization and flow of information in response to language. All these lofty goals are assumed to be underlied by efficient retrieval algorithms. Of course, the problem has not been ignored completely, since the program does function (if somewhat inefficiently), and the data structures described were designed with timewise-efficient associative retrieval in mind (at the expense of storage requirements). Furthermore, some retrieval functions are discussed as part of the theory. Section 4.3 discusses some uses of associative searches through conceptual information, and chapter 7 covers a few more points about retrieval.

### 3.7.1 PARTITIONING THE MEMORY

It should be made clear that I have chosen not to partition the memory artificially into functionally separate units (say, CP, IM, LTM). Again, I am not primarily concerned with modeling the *hardware* of the brain, but rather its topology from the standpoint of the *logical* flow of information within it. This is not to say that to discover the *physical* flow as well would not be interesting and useful. But to do so would, for instance, draw us into issues of what enters and leaves CP, why and when. Answers to questions of this type would certainly augment the theory nicely, and would give insights about how to limit searches. But much can be done *without* partitions, and as we will see in the next chapter, RECENCY and TOUCHED provide a rudimentary form of logical partitioning.

## 3.8 HOW IT ALL HANGS TOGETHER: AN EXAMPLE OF AN INTERNALIZED CONCEPTUALIZATION

I will conclude this chapter with an example of an utterance which is relatively complex with respect to this theory. I will show (1) how CD represents it, (2) what the resulting memory structure will look like (in schematic graph form), (3) what the LISP representation of the CD graph which MEMORY receives from a conceptual analyzer looks like, and (4) the final memory structure in a "virgin" memory.

The example is:

> Mary didn't give Bill the red book which Pete had given her because Bill had aggravated her by choking her friend John."

The conceptual dependency graph which represents this utterance is shown in Fig. 3-14.

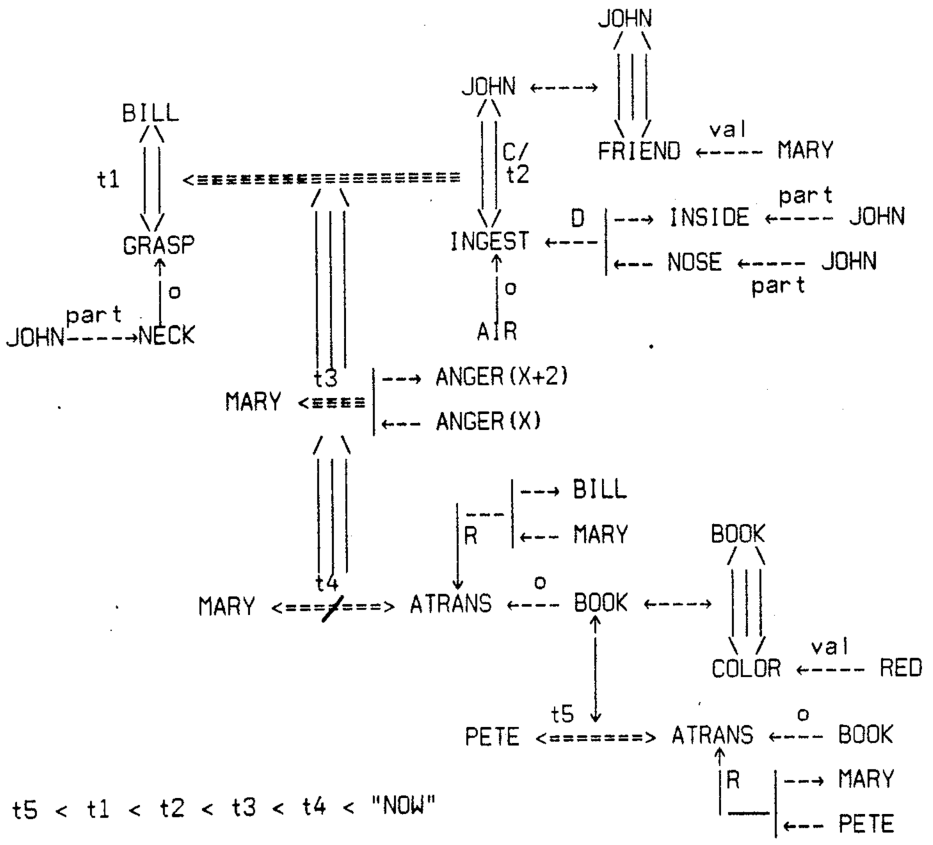
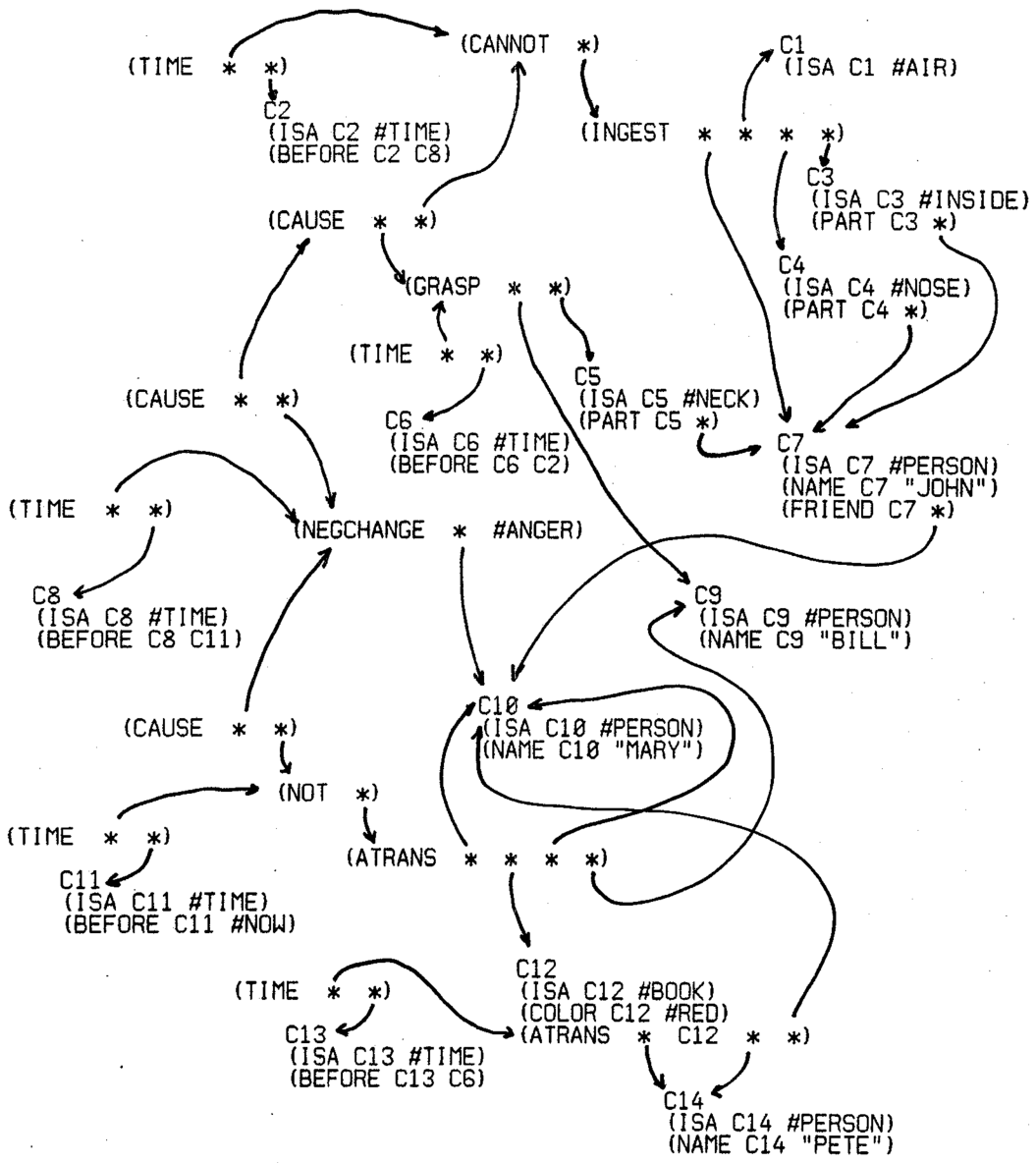



**Figure 3-14.**

(The REL link in Fig. 3-14 specifing that John and Mary are friends is in fact repeated at each occurrence of "JOHN" in the graph. However, the analyzer creates just one object for this John who is Mary's friend, then uses pointers to this object in the LISP version so that anywhere the PP "JOHN" appears above, it is in fact just a pointer to this one object. Similar remarks apply to all PPs which occur at more than one point in the graph: PPs which have the same name in this graph are guaranteed to have EQ LISP pointers in the computer analysis.)

This analyzed meaning graph will ultimately map onto the memory structure whose pictorial graph representation is shown below. In general, only small portions of each token's occurrence set are evident in the graph.




Here, the BEFORE time relations have been specified by atom names rather than pointers simply because connecting them would render the illustration illegible. Recall that each proposition as well as each concept is stored as a superatom and, as such, can be referenced by arbitrarily many other propositions. Also recall that all links are two-way: they have been

drawn one-way for clarity in the graph. It is important to remember also that the #<NAME> entities in the graph are also pointers to other concepts which are no more than sets of conceptual features. The names used merely serve to identify those sets for the sake of illustration.

*How* this graph is internalized, and how all the information it contains is extracted are the essence of the next chapter. The following is the actual computer structure in which this example results.

## REPRESENTATION COMPUTER EXAMPLE

The following sequence shows the LISP form of the meaning graph which the memory receives from a conceptual analyzer. Section 5.5 describes the syntax conventions for representing CD graphs in LISP S-expressions. This particular input to MEMORY is a retouched and augmented output of Riesbeck's conceptual analyzer.

```
MARY DIDNT GIVE BILL THE RED BOOK
WHICH PETE HAD GIVEN MARY BECAUSE
BILL HAD AGGRAVATED MARY BY CHOKING
MARYS FRIEND JOHN
```

This is the input sentence as the conceptual analyzer receives it. Below is the conceptual analysis which the analyzer sends MEMORY.

```
LEGEND OF ROLES

CON      takes an entire conceptualization
ACTOR    the actor of an action
<=>      the ACT of the conceptualization
OBJECT   the conceptual object
PART     a modifier asserting a part-of relationship
VAL      the value of some concept with respect to the predicate which VAL modifies
<≡>      the attributive relation, taking as rolefiller some relation
<≡       the causal relationship. Causal forms are represented as (CON X <≡ Y).
↔        the REL dependency, used to specify concepts further (takes a complete CON)
REF      the nature of the determiner used with a concept
TO       the directive or recipient "to" case for certain actions
FROM     the directive or recipient from case for certain actions
TIME     a modifier for complete conceptualizations
MODE     a modifier for complete conceptualizations, taking as rollfiller a list of
         modes. Examples of modes are *CANNOT*, *NEG*, *CAN*.
INC      the incremental amount by which a statechange occurred.
^        the conjnction of two complete conceptualizations
```

```
((CON
  ((CON
    ((CON
      ((ACTOR (BILL)
              <=>
              (*GRASP*)
              OBJECT
              (NECK PART
```

```
                    (JOHN ↔
                          ((ACTOR (JOHN)
                                  <≡>
                                  (FRIEND VAL (MARY)))))))
     TIME
     (TIM01))
    <≡
    ((ACTOR (JOHN ↔ ((ACTOR (JOHN) <≡> (FRIEND VAL (MARY)))))
            <=>
            (*INGEST*)
            OBJECT
            (*AIR* REF (*A*))
            FROM
            (NOSE PART
                  (JOHN ↔
                        ((ACTOR (JOHN)
                                <≡>
                                (FRIEND VAL (MARY))))))
            TO
            (*INSIDE* PART
                      (JOHN ↔
                            ((ACTOR (JOHN)
                                    <≡>
                                    (FRIEND VAL (MARY)))))))
     TIME
     (TIM02)
     MODE
     ((*CANNOT*)))))
  <≡
  ((ACTOR (MARY) <≡>F (*ANGER*) <≡>T (*ANGER*))
   TIME
   (TIM03)
   INC
   (-2))))
^
((CON
  ((ACTOR (MARY) <≡>F (*ANGER*) <≡>T (*ANGER*))
   TIME
   (TIM03)
   INC
   (-2))
  <≡
  ((ACTOR (MARY)
          <=>
          (*ATRANS*)
          OBJECT
          (BOOK ↔
                ((ACTOR (BOOK REF (*THE*))
                        <≡>
                        (COLOR VAL (RED))))
                ↔
                ((ACTOR (PETE)
                        <=>
                        (*ATRANS*)
                        OBJECT
                        (BOOK REF (*THE*))
                        FROM
                        (PETE)
                        TO
                        (MARY))
                 TIME
                 (TIM05)))
          FROM
          (MARY)
          TO
          (BILL))
```

```
      TIME
      (TIM04)
      MODE
      ((*NEG*)))))))

TIM00: ((VAL T-0))
TIM01: ((BEFORE TIM02 X))
TIM02: ((BEFORE TIM03 X))
TIM03: ((BEFORE TIM04 X))
TIM04: ((BEFORE TIM00 X))
TIM05: ((BEFORE TIM01 X))
```

Having received this as input, MEMORY integrates it into its data structures. Below is an intermediate structure which has undergone some transformations of form and which contains references to real world tokens in MEMORY.

```
((ANDX
  ((CAUSE
     ((CAUSE ((*GRASP* (#BILL1) (C0004))
                (TIME _ (C0007)))
             ((CANNOT ((*INGEST* (C0001) (C0016) (C0018) (C0021))
                       (TIME _ (C0008)))))))
      ((NEGCHANGE (#MARY1) (#ANGER))
          (TIME _ (C0009)))))
   ((CAUSE ((NEGCHANGE (#MARY1) (#ANGER))
              (TIME _ (C0009)))
           ((NOT ((*ATRANS* (#MARY1) (C0026) (#MARY1) (#BILL1))
                  (TIME _ (C0010)))))))))
```

MEMORY then completes the "internalization" of this input structure by creating bonds and superatoms to represent its various components. At the end of this internalization, the structure is represented by superatom C0044. We ask MEMORY to dump itself at this point.

```
C0044
```

C0044 is the superatom under which the entire input structure has been stored. Having internalized the input, we now request MEMORY to dump its contents. The input was received with MEMORY in a "virgin" state, so only this structure is present in MEMORY (along, of course, with the approximately 200 virgin structures.

```
C0001: NIL

ASET:
   C0033: (*INGEST* # C0016 C0018 C0021)
   C0023: (PART C0021 #)
   C0020: (PART C0018 #)
   C0006: (PART C0004 #)
   C0003: (FRIEND # #MARY1)
   C0002: (NAME # JOHN)
```

C0001 is the concept which represents the person named "John" in the input. MEMORY was purposefully initialized with two people named John to force the creation of this new token. Chapter 8 describes how MEMORY will return to this token after inferencing in an attempt to decide which of the two candidate "John" concepts C0001 references.

```
RECENCY: 8966

-----------------------------------
C0002: (NAME C0001 JOHN)
```

C0002 is the information that the name of C0001 is "John".

```
-----------------------------------
C0003: (FRIEND C0001 #MARY1)
```

C0003 is the information that C0001 is a friend of Mary. MEMORY was initialized to know of only one Mary, Pete and Bill to demonstrate how unambiguous references are processed.

```
-----------------------------------
C0004: NIL

ASET:
   C0031: (*GRASP* #BILL1 #)
   C0006: (PART # C0001)
   C0005: (ISA # #NECK)
RECENCY: 8916
```

C0004 is C0001's (John's) neck which Bill grasped.

```
-----------------------------------
C0005: (ISA C0004 #NECK)
```

C0005 is the information that C0005 is a neck.

```
-----------------------------------
C0006: (PART C0004 C0001)
```

C0006 is the information that C0006 is a part of C0001.

```
-----------------------------------
C0007: NIL

ASET:
   C0032: (TIME C0031 #)
   C0025: (BEFORE C0024 #)
   C0015: (BEFORE # C0008)
RECENCY: 8933
```

C0007 is the time at which action C0031 occurred. C0031 is Bill's grasping of John's neck. Notice that its relative position on the time scale is recorded in C0025 and C0015.

```
-----------------------------------
C0008: NIL

ASET:
   C0034: (TIME C0033 #)
   C0015: (BEFORE C0007 #)
   C0014: (BEFORE # C0009)
RECENCY: 8983
```

C0008 is the time of John's inability to ingest air, C0033.

```
-----------------------------------
C0009: NIL

ASET:
   C0038: (TIME C0037 #)
   C0014: (BEFORE C0008 #)
   C0013: (BEFORE # C0010)
RECENCY: 9116
```

C0009 is the time of Mary's becoming angry at Bill, C0037.

```
-----------------------------------
C0010: NIL

ASET:
   C0041: (TIME C0040 #)
   C0013: (BEFORE C0009 #)
   C0012: (BEFORE # C0011)
RECENCY: 9183
```

C0010 is the time at which Mary's giving the book to Bill did not occur. This non-event is structure C0040.

```
-----------------------------------
```

00011 is the time of utterance of the input

```
C0011: NIL

ASET:
   C0012: (BEFORE C0010 #)
RECENCY: 7366, TVAL: 6833

-----------------------------------
C0012: (BEFORE C0010 C0011)


-----------------------------------
C0013: (BEFORE C0009 C0010)


-----------------------------------
C0014: (BEFORE C0008 C0009)


-----------------------------------
C0015: (BEFORE C0007 C0008)


-----------------------------------
C0016: NIL

ASET:
   C0033: (*INGEST* C0001 # C0018 C0021)
   C0017: (ISA # #AIR)
RECENCY: 8966

-----------------------------------
C0017: (ISA C0016 #AIR)


-----------------------------------
C0018: NIL

ASET:
   C0033: (*INGEST* C0001 C0016 # C0021)
   C0020: (PART # C0001)
   C0019: (ISA # #NOSE)
RECENCY: 8983

-----------------------------------
C0019: (ISA C0018 #NOSE)


-----------------------------------
C0020: (PART C0018 C0001)


-----------------------------------
C0021: NIL

ASET:
   C0033: (*INGEST* C0001 C0016 C0018 #)
   C0023: (PART # C0001)
   C0022: (ISA # #INSIDE)
RECENCY: 8983

-----------------------------------
C0022: (ISA C0021 #INSIDE)


-----------------------------------
C0023: (PART C0021 C0001)
```

structure. Notice that the system clock has been recorded as its TVAL.

C0012, C0013, C0014 and C0015 are the relational information among the various times alluded to by this input.

C0016 is the air which John could not ingest. It makes sense to create this non-entity: MEMORY might encounter an input like "John couldn't breathe, but the air was poisonous anyway."

C0017 is the information that C0016 is a token of some air.

C0018 is C0001's nose.

C0019 is the information that C0018 is a nose.

C0020 is the information that C0018 is part of C0001 (John).

C0021 is C0001's insides.

C0022 is the information that C0021 is an "inside".

C0023 is the information that C0021 is part of C0001 (John).

```
-----------------------------------
C0024: NIL

ASET:
   C0029: (TIME C0028 #)
   C0025: (BEFORE # C0007)
RECENCY: 8200

-----------------------------------
C0025: (BEFORE C0024 C0007)


-----------------------------------
C0026: NIL

ASET:
   C0040: (*ATRANS* #MARY1 # #MARY1
                #BILL1)
   C0030: (COLOR # #RED)
   C0028: (*ATRANS* #PETE1 # #PETE1
                #MARY1)
   C0027: (ISA # #BOOK)
RECENCY: 9183

-----------------------------------
C0027: (ISA C0026 #BOOK)


-----------------------------------
C0028: (*ATRANS* #PETE1 C0026 #PETE1
                #MARY1)

ASET:
   C0029: (TIME # C0024)

-----------------------------------
C0029: (TIME C0028 C0024)


-----------------------------------
C0030: (COLOR C0026 #RED)


-----------------------------------
C0031: (*GRASP* #BILL1 C0004)

ASET:
   C0036: (CAUSE # C0035)
   C0032: (TIME # C0007)

-----------------------------------
C0032: (TIME C0031 C0007)


-----------------------------------
C0033: (*INGEST* C0001 C0016 C0018
                C0021)

ASET:
   C0035: (CANNOT #)
   C0034: (TIME # C0008)

-----------------------------------
```

C0024 is the time at which Pete gave the book to Mary.

C0025 is the information that the time of Pete's giving Mary the book was before C0007, the time of Bill's grasping John's neck.

C0026 is the red book which Pete gave to Mary, and which Mary did not give to Bill. (C0040 is modified by a NOT).

C0027 is the information that C0026 is a token of a book.

C0028 is Pete's giving of the book to Mary at time C0024.

C0029 is the information that Pete's giving the book to Mary occurred at time C0024.

C0030 is the information that the book is red.

C0031 is Bill's grasping of John's neck (in the choke action). Notice that it caused C0035, John's inability to ingest air.

C0032 is the information that Bill's grasping action occurred at time C0007.

C0033 is the ingesting action which John (C0001) was unable to perform at time C0008. Notice its CANNOT modification, C0035.

00034 is the information that the time

C0034: (TIME C0033 C0008)

of John's inability to ingest air was C0008.

---

C0035: (CANNOT C0033)

ASET:
C0036: (CAUSE C0031 #)

C0035 is the information that John's ingesting action was unable to occur. Notice that it was caused by C0031, Bill's grasping action.

---

C0036: (CAUSE C0031 C0035)

ASET:
C0039: (CAUSE # C0037)

C0036 is the information that Bill's grasping caused John's inability to ingest air. Notice that C0036 in turn caused C0037, Mary's incipient anger (aggravation).

---

C0037: (NEGCHANGE #MARY1 #ANGER)

ASET:
C0043: (CAUSE # C0042)
C0039: (CAUSE C0036 #)
C0038: (TIME # C0009)

C0037 is Mary's increase in anger which was caused by C0036, and which in turn caused C0042, Mary's not giving Bill the book.

---

C0038: (TIME C0037 C0009)

C0038 is the information that Mary's increase in anger occurred at time C0009.

---

C0039: (CAUSE C0036 C0037)

ASET:
C0044: (ANDX # C0043)

C0039 is the information that Mary's increase in anger was caused by Bill's choking of John.

---

C0040: (*ATRANS* #MARY1 C0026 #MARY1 #BILL1)

ASET:
C0042: (NOT #)
C0041: (TIME # C0010)

C0040 is the giving of the book to Bill which Mary didn't perform at time C0010.

---

C0041: (TIME C0040 C0010)

C0041 is the infomation that the time of Mary's unwillingness to give Bill the book was C0010.

---

C0042: (NOT C0040)

ASET:
C0043: (CAUSE C0037 #)

C0042 is the information that Mary's giving action did not occur.

---

C0043: (CAUSE C0037 C0042)

ASET:
C0044: (ANDX C0039 #)

C0043 is the information that Mary's anger caused he not to give Bill the book.

---

C0044: (ANDX C0039 C0043)

C0044 is the information that two events occurred. C0044 constitutes the complete input structure.

---

---

#BILL1: NIL

Finally, we have a look at Mary, Bill and Bate as they exist after this input. Innnn

```
ASET:
   C0040: (*ATRANS* #MARY1 C0026
               #MARY1 #)
   C0031: (*GRASP* # C0004)
   I0008: (ISA # #PERSON)
   I0007: (NAME # BILL)
RECENCY: 9183


-----------------------------------
#MARY1: NIL

ASET:
   C0040: (*ATRANS* # C0026 # #BILL1)
   C0037: (NEGCHANGE # #ANGER)
   C0028: (*ATRANS* #PETE1 C0026
               #PETE1 #)
   C0003: (FRIEND C0001 #)
   I0004: (ISA # #PERSON)
   I0003: (NAME # MARY)
RECENCY: 9183

-----------------------------------
#PETE1: NIL

ASET:
   C0028: (*ATRANS* # C0026 # #MARY1)
   I0202: (ISA # #PERSON)
   I0201: (NAME # PETE)
RECENCY: 6833
```

structures are in general those which were present in the virgin system.

# CHAPTER 4

## GETTING CONCEPTUAL GRAPHS INTO THE MEMORY: REFERENCE, WORD SENSE PROMOTIONS, INTERNALIZATION

In the last chapter, I discussed the main issues of representation for a conceptual memory. We turn now to more process-oriented issues: how does the meaning graph become a memory structure, and what effects does this have on the memory. There is again a multitude of issues here, and I have addressed those which seem to be most typical of the kinds of processing at this language-memory interface. The general issue is how the analyzed conceptual graph is transformed into internal memory structures. Within this main topic, we will cover the following five areas:

1. the identification of concepts and tokens from sets of conceptual features
2. the creation of temporary tokens for those tokens and concepts which cannot be identified
3. the activation of concepts and tokens, and the memory's interaction with the conceptual analyzer in this regard
4. the creation of bonds to store all the sub-conceptualizations contained in the conceptual graph. This includes the mapping of time references and deictic time concepts onto concrete time tokens.
5. the extraction of subpropositions from the graph to form the initial set of structures from which conceptual inferencing will begin

## 4.1 REFERENCING CONCEPTS AND TOKENS FROM LANGUAGE: DESCRIPTIVE SETS

**The words which appear in an utterance are gone by the time the memory begins processing. All that remains is conceptual information. What does the conceptual information which identifies a concept or token look like?**

Since most of the original words of an utterance are gone from the conceptual graph which is the product of the conceptual analyzer, it is a meaningful question to ask what becomes of the words, and how the memory uses what it gets to identify or create concepts and their tokens before other other processing begins. This section describes this interface.

For words in the sentence which might reference concepts or their tokens, the analyzer's job is first to choose the correct lexical *sense* of the word, or make a best guess based on its conceptual context. This identifies a conceptual PP ("picture producer"). This mapping of a word onto a PP (sense choice) makes available all conceptual knowledge about that PP. For example, when the word "John" is mapped onto the PP which is a male human with name "John", the conceptual features:

```
(ISA # #PERSON)
(SEX # #MALE)
(NAME # JOHN)
```

stored with this PP become available. The "poundsign" is used to denote the object being described. These conceptual features become the kernel of the language-referenced concept's *descriptive set.*

> **A descriptive set is an unordered list of conceptual features which describe some concept or token (or perhaps many concepts or tokens).**

Any other conceptual information about this object the analyzer can glean from the sentence augments this kernel. Such information typically comes from sentential adjectives and relative clauses which correspond to individual pieces of conceptual information.

For the PP, "the big red dog who ate the bird", the descriptive set shown in Fig. 4-1 would arise. Note there that I have written some internal memory concepts (#DOG for instance). The process of reference identification is recursive, so that they too have previously been referenced by other descriptive sets. Had these been shown in Fig. 4-1, #DOG, for instance, would be replaced by the descriptive set { (ISA # #ANIMAL) (NAME # DOG) }.

```
X: { (ISA X #DOG)
     (COLOR X #RED)
     (RELSIZE X #LARGE)
     ( Y: (INGEST X
                  { (ISA # #BIRD) (REF # *THE*) }
                  { (ISA # #MOUTH) }
                  { (ISA # #STOMACH) }
          )
             (TIME Y { (ISA # #TIME) (BEFORE # #NOW) } )
     )
     (REF X *THE*)
  }
```

**Figure 4-1. Descriptive set for "The big red dog who ate the bird".**

Notice in Fig. 4-1 that the fourth member of X's descriptive set (the INGEST feature) has a time modification. In general, any descriptive set element can have its own additional modification. This is the general form of a descriptive set member; the first members (ISA, COLOR, RELSIZE) are simple cases (they have no nested modifiers). Since, empirically, language rarely ventures beyond these two levels of nesting in this respect, the program is equipped to deal with only this secondary level of modification.

Fig. 4-2 shows the general form of a descriptive set which can be processed by the memory.

```
{ ( <feature> <modifier> ... <modifier> )
                    .
                    .
                    .
  ( <feature> <modifier> ... <modifier> )
}
```

**Figure 4-2. Syntax of a descriptive set.**

### 4.1.1 MULTIPLE OCCURRENCES OF A CONCEPT IN A GRAPH

The conceptual analyzer guarantees that, for two PPs or conceptualizations which (based on the analyzer's linguistic knowledge) reference the same real world concept or event, not only will their descriptive sets be identical, but they will be LISP "EQ" in the graph. That is, all occurrences of them in the graph will point to the same physical descriptive set. Thus, in the

conceptualization "John gave Bill the apple" (Fig. 4-3, left), although "John" occurs in the conceptual graph twice, in reality, each occurrence is a pointer which references the same descriptive set (Fig. 4-3, right). This equality is also preserved for entire subconceptualizations which reference the same state or action.

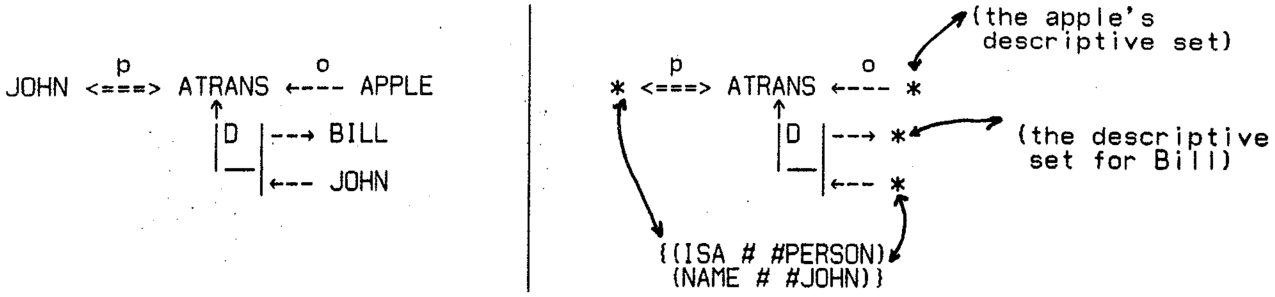



**Figure 4-3. "EQ" LISP pointers to identical references within a graph.**

During the internalization process, the memory maintains a list, !REFLIST, of pointers to all descriptive sets and subconceptualizations which have been processed (identified with internal memory units) up to that point in the internalization process. Before each new descriptive set or subconceptualization is processed, its membership on !REFLIST is first checked, and if it is found, the associated memory structure which resulted from previous references to it is used with no further processing.

### 4.1.2 WHEN REFERENT IDENTIFICATION IS PERFORMED

The descriptive set is the unit which memory receives from the analyzer for each concept and token referenced in an utterance. In theory, the memory may be called either *during* or *after* the conceptual analysis to attempt an identification of the token or concept referenced by some descriptive set. The former case typically will occur when the analyzer can predict that it would be useful to the analysis to have more *specific* knowledge about the concept. If the referent can be identified at that time, its entire occurrence set (everything known about it) will become available to the analyzer, and this newly accessible knowledge can subsequently influence the interpretation of the utterance or of future utterances. "John's pitch was foul" is such an example. In this utterance knowing more about John could solve the ambiguity of "pitch".

Although this form of analyzer-memory interaction remains to be exploited in the current implementation, the nature of the interaction is straightforward: the analyzer passes the memory a reference request in the form of a descriptive set (all the features collected about an entity so far), and the memory returns with one most likely referent, or a list of candidates if more than one referent is likely. Normally, because of the RECENCY heuristic I will describe, there will be only one. Notice that this is one place at which the solution of anaphoric references logically fits in the processing sequence, although much research remains to be done in this area.

In the case where the analyzer can get along with only PP's, rather than fully-identified memory concepts and tokens which have actually been identified, referents are not identified during the analysis. Rather, the entire graph is constructed, then passed along to the memory. This is the nature of the current analyzer-memory interface.

The process of reference establishment therefore occurs as part of the larger task of getting the analyzed graph -- which consists of descriptive sets connected by conceptual links -- into memory structures suitable for conceptual inferencing. I have called this process *integration*. In the next section, we will cover the first two tasks of the five listed at the beginning of the chapter.

## 4.2 THE REFERENCE MECHANISM: SEARCHING FOR REFERENTS OF DESCRIPTIVE SETS

**How are concepts and tokens actually identified from a descriptive set? What happens when a descriptive set identifies more than one concept or token?**

The process which identifies memory concepts and tokens from descriptive sets is called the *reference mechanism*, or just the "referencer". For descriptive set, D, its task is to discover to which of the memory's large number of concepts (and still larger number of tokens) D refers, so that occurrences of the descriptive set in the conceptual structure can be replaced by internal memory pointers to its referent. To do this is to gain access to occurrence set of some internal token or concept (all its conceptual features), and this access is crucial to the process of inferencing which we are leading up to.

### 4.2.1 ORDERING THE DESCRIPTIVE SET

The first step for the referencer is to reorder D, using a very simple "reference significance" heuristic. Before describing it, let's see why the ordering of the set has any significance in the first place. Consider the reference "Henry Kissinger, who is in Peking, ...". Suppose the descriptive set is given to the referencer with the order shown in Fig. 4-4, and suppose the memory knows Dr. Kissinger (has a token for him), but didn't happen to know he is in Peking. If the memory were *first* to search for an X, such that X is in Peking, Kissinger would not be in the set thus located. Yet it is patently obvious to a human language user that this new information is indeed new, and that the name of X alone serves as a positive identification. In other words, new information has been communicated *via a descriptive set*, and we don't want this in general to disrupt the identification of an otherwise obvious reference. Computers are dumb, so we have to help out; this is the goal of this reordering.

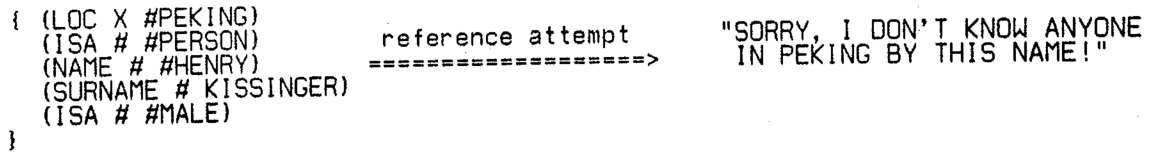



**Figure 4-4. A dumb reference mechanism.**

The ordering is simply a heuristic measure upon which all system predicates are rated according to their nominal usefulness to reference establishment. For example, NAME, SURNAME, SEX, and ISA all have very high values because they are very powerful and concise units of information from which to identify referents, while ACT and STATE predicates (such as the LOC in this example) have lesser values. This ordering will force the search to look first at those conceptual features which are usually very critical to the correct identification of a referent.

It should be clear that this ordering is not necessarily from most-specific to least-specific. Rather, it is designed to increase the odds that the intersection search which we will describe shortly will not fail simply because the descriptive set contains some obscure or new feature which would eliminate the correct referent, C, from the search because that relatively insignificant feature was not previously known about C.

Hence, for the decriptive set above, we would like the reordering shown in Fig. 4-5.

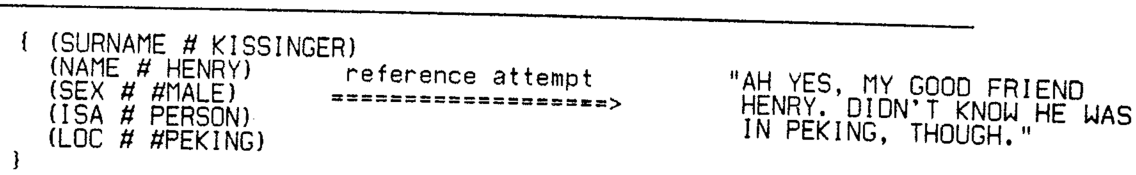



**Figure 4-5. The same dumb referencer, working with a reordered descriptive set.**

Thus, if the memory hadn't previously known that Kissinger was in Peking, the LOC element of the descriptive set would not interfere in the identification of Kissinger.

Also, it should be pointed out that this reordering does not buy any theoretical power. It is merely a guess about what is likely to be the optimal order in which to perform the intersection search which locates referents. It also aids in deciding when a descriptive set describes some token or concept "closely enough" to match, and when to augment the matched entity's occurrence set with those descriptive features of lesser importance which were not successfully used in the identification. As we will see, when the intersection search can unambiguously locate a referent from this reordered descriptive set *before the set is exhausted*, descriptive features in the remainder of the set stand a chance of conveying new information about the identified referent. As such they should be checked, and if they are *new*, they can provide one source from which to generate conceptual inferences.

### 4.2.2 THE INTERSECTION SEARCH

The intersection search is straightforward, and occurs as follows. Starting with the first feature in the reordered descriptive set, D', the memory locates all entities in the memory which satisfy this feature. These entities are placed on a list as the remaining candidates. Each candidate is then tested for the second feature in the descriptive set, and those which survive become the remaining candidates. This process continues until one of the following occurs:

1. D' has been exhausted
2. exactly one candidate remains
3. the next feature in D' would cause the candidate set to become null

In a large, information-rich memory, such as that of a language user, at least one object will be found on most occasions.

In the memory, if exactly one is found, it is assumed to be the referent. It augments !REFLIST and will replace occurrences of the descriptive set in the graph. When more than one object is found in the intersection, each one is examined for RECENCY, which stores the value of the system clock at the time a successful reference to that object was last established. If this information breaks the tie, the most recent object is selected. Otherwise, each candidate's TOUCHED property is examined. Recall that TOUCHED is like recency, except that it records *implicit* references to a concept or token which have been drawn out by other memory processes rather than by language directly. If one of the candidates has a more recent TOUCHED value than the rest, it is selected. We will see later how TOUCHED can be of extreme significance to this process.

If the intersection search *begins* to locate a candidate set, but some feature on the descriptive set would cause the next intersection to rule out *all* candidates, the search is suspended. The remaining features which have not yet played a part are then scanned, keeping a tally of how many are satisfied by each candidate. If one candidate satisfies more features than the other candidates, it is chosen, and the features it did not satisfy are assumed to be new. For a candidate, C, to "satisfy" a feature simply means that C possesses that feature *exactly*. However, I would eventually like to make this notion of "satisfy" looser.

If no one candidate can be selected over the others, no decision is made at that time. More will be said about this later.

### 4.2.3 ADDITIONAL HEURISTICS -- AND PROBLEMS

It might be pointed out that, in addition to information explicitly contained in the descriptive set, there are other heuristics which the memory could use in selecting a referent. Among them is the *hearer's modeling of the speaker*. For instance, if John refers to Bill while speaking to Sue, Sue (and the memory, should it also hear John) may usefully assume that this Bill lies in the intersection of Sue's and John's acquaintences. That is, the descriptive set can be augmented by this modeling information:

```
#: ( (NAME # BILL)
     (ISA # #PERSON)
     (ACQUAINTED # #SUE)
     (ACQUAINTED # #JOHN)
```

However, this type of modeling is not presently performed.

Besides this absence of modeling, there are many other subtle problems with this process of intersection searches for referents. The basic one is deciding *which* features of the descriptive set to ignore (selectively) in case the intersection of *all* of them turns out to be null. The problem is: which *combination* of features is likely to be useful when all features together yield a null intersection? To make this process more intelligent than it currently is, some fairly subtle heuristics will be needed to avoid the combinatorics of features taken N at a time. This is one point at which more theory remains to be developed for descriptive sets and the identification process.

However, in practice, the algorithm I have described will be successful enough of the time so that this is not a major issue in the total picture of the memory. We might conjecture that the reason for this is that speakers tend to include in descriptive sets exactly what they feel is most important to the hearer's correct and expedient identification of the entities being referenced (in the context in which they are referenced). This is an important facet of how the *speaker* models the *hearer*. How he chooses to identify X when speaking to P1 can be entirely different from how he chooses to identify X when speaking to P2, and these choices are based on his models of what he believes P1 and P2 know about X. As we will see in the following chapters, some types of conceptual inference rely on a rather crude ability to model other people's knowledge, but none of these involve quite so subtle a problem as this.

### 4.2.4 HANDLING UNIDENTIFIED REFERENCES

When the reference search algorithm

(a) fails to locate any candidates for the referent of a descriptive set, D, or
(b) locates several candidates, but none can be selected over the rest,

the referencer *creates a new token*, T, to represent this unidentified referent of D. In case (a), D

becomes T's starting occurrence set. That is, the memory doesn't yet know what it has, but it is willing to go along with it at this point. In addition T is recorded on the list !REFNOTFOUND to note this reference failure.

Quite often, conceptual inferences will arise which will contribute new features to T's occurrence set, and these new features will be of use in determining the referent after the inference mechanism has finished. In other cases, the referent simply will never be determined: a new concept or token has in fact been introduced, or an existing one has been referenced in such an obscure way that the reference is impossible to establish. These cases (members of !REFNOTFOUND) provide one source from which the memory can generate questions at a later time.

An example of case (a) above (no candidates can be located) is:

John ate a green frob

where the memory would be incapable of locating something whose NAME is "frob", the concept of which this green object is a token. It would therefore create a new concept as shown in Fig. 4-6. Having done so, it could then create a token of this new concept which is green and which John ate.

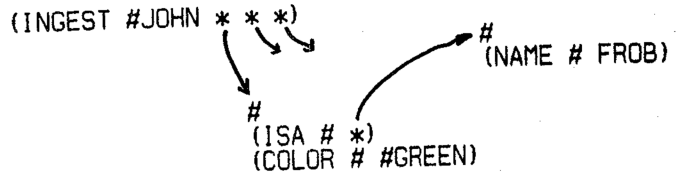



**Figure 4-6.**

#### 4.2.4.1 WHEN SEVERAL CANDIDATES ARE LOCATED

In case (b) above -- when *several* candidates for the referent are located -- a temporary token, T, is also created. It is then associated with its candidate set:

( <new token> <candidate> <candidate> ... <candidate> )

and this association is added to a special list, !REFDECISION, which will be used later after inferencing.

**In other words, the choice of referent is deferred for the time being.**

The hope is that, by generating new information about T during the process of inferencing, new features of T will be turned up which can either narrow the candidate set or choose one candidate unambiguously. Since chapter 8 is devoted to this reference-inference interaction, I will not go into it here.

This new T which stands for one of the candidates receives as its beginning occurrence set all those conceptual features which lie in the *intersection* of the features of all its candidates. This will be at least the descriptive set, D, from which the candidates were determined in the first place, but it may also include *other* features which are common to all the candidates. By locating these common features and associating them with the temporary token, T, the chances for making important inferences involving T increases, since many inferences will be dependent upon T's partial conceptual features.

Fig. 4-7 illustrates this ability to defer reference decisions until inferences have had a chance to contribute new information. The example shows a simple case involving two candidates for some person named "John".

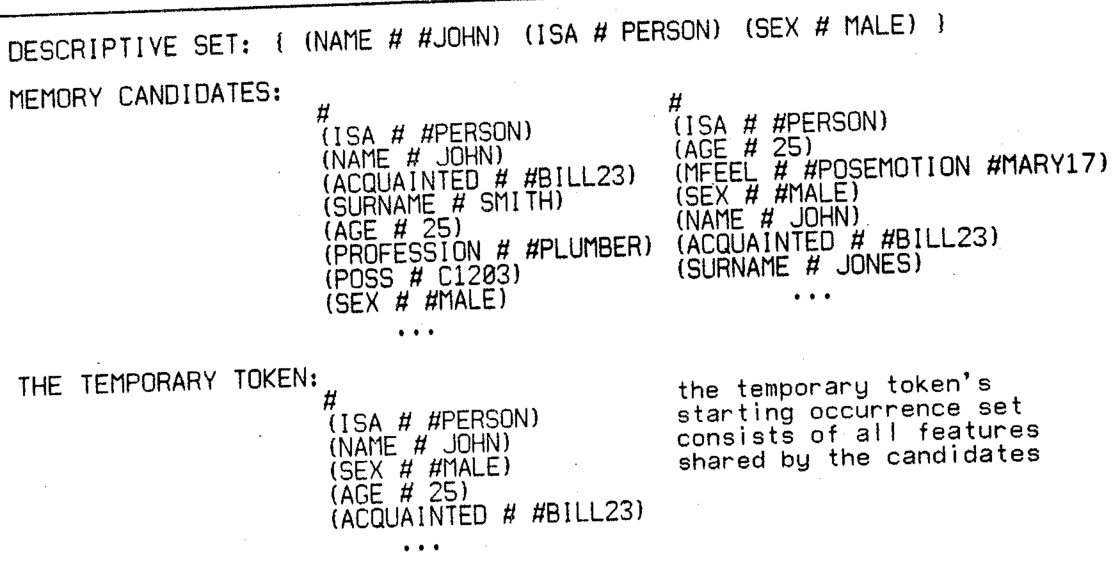



**Figure 4-7. Deferring reference decisions.**

### 4.2.5 REFERENCE SIGNALS AND THE SPECIAL PREDICATE "REF"

**How are references to tokens of a concept distinguished from references to the concept itself? That is, what conceptual information from the analyzer signals these different cases, and how does the memory use this information to locate or create concepts or their tokens?**

The descriptive set of a concept or token which is gathered from an utterance by the conceptual analyzer typically consists of a (NAME X Y) feature, perhaps a (REF X Z), and usually two or three other conceptual features of X which were either explicitly found in the utterance or inferred by the analyzer, using language-specific knowledge, knowledge about conceptual case restrictions, and so forth. It is the reference-finder's task, given this set of propositions, (a) to arrive at the best possible identification of the concept at that point, (b) to note whether any decision was made in doing so, or (c) to note that the concept was not identified and consequently had to be created from the conceptual propositions given. This section describes the effects of the predicate REF on this process.

REF is a special kind of conceptual predicate: the information it conveys about a concept is

in the form of a *signal*, which indicates the kind of processing the reference-finder should perform in order to locate the object described by the descriptive set. In the current program, there are three forms of this REFerence signal: *A*, *THE*, and null (REF is absent). The effects of these three reference signals are of considerable significance to the manner in which the referencer treats the rest of the descriptive set in which the REF occurs. We will consider each of these three signals individually.

#### 4.2.5.1 REF *A*

**sample:** John gave Mary *a book about whales.*

**sample:** Bill bought *some spoiled milk.*

Consider the analyzed graph component which references "a book about whales" in the first sample. This will have the form shown in Fig. 4-8, namely, "a book in which are located concepts which involve the concept whale". This form directs the memory to create a *token* of the concept which is identifiable by the feature (NAME # BOOK), together with other conceptual features of this PP, BOOK, which are stored on its property list, and which distinguish it from the PP's to which other senses of the word "book" refer. For the PP BOOK, suppose this feature set consists of just one other feature: (ISA # #PRINTEDMATTER).

```
(BOOK REF (*A*)
      ↔ ((CON (*CONCEPTS* REF (*A*)
                        ↔ ((ACTOR (WHALE) <≡> (INVOLV VAL (*CONCEPTS*)))))
         <≡> (MLOC VAL (BOOK))
)
```

**Figure 4-8. A REF *A* signaled descriptive set.**

The memory must therefore first identify the referent of this *concept*, C, from the descriptive set:

{ (NAME # BOOK) (ISA # #PRINTEDMATTER) }

before it can be concerned with the particular *token* of this concept which is being referenced. As we have seen, C will either be uniquely located, or a temporary concept will be created to represent it. In this example, #BOOK will be located.

Having identified C as #BOOK, the memory must then *create* a new token, T, of this concept. Since an indeterminate reference is nominally a signal that a new token is about to be *introduced*, no intersection search to locate an existing token should be performed. Rather the token should simply be created and accepted as new. The T thus created will serve as the referent and will (correctly) never wind up on the list !REFNOTFOUND.

In this example, the descriptive set which defines T(C) will be the remainder of the original descriptive set, plus an (ISA # C) to indicate the concept of which this token is an instance. Since C will be #BOOK, the token's defining descriptive set will be

{ (ISA # #BOOK) (MLOC X #) }

where X stands for the token which represents these concepts about whales. This token will have been recursively created by the same mechanism described here. T will soon thereafter receive the additional occurrence set member (ATRANS #JOHN T #JOHN #MARY) during the internalizing process described in section 4.5. After the complete reference and internalization process, the resulting memory structure for this reference will be that shown in Fig. 4-9.

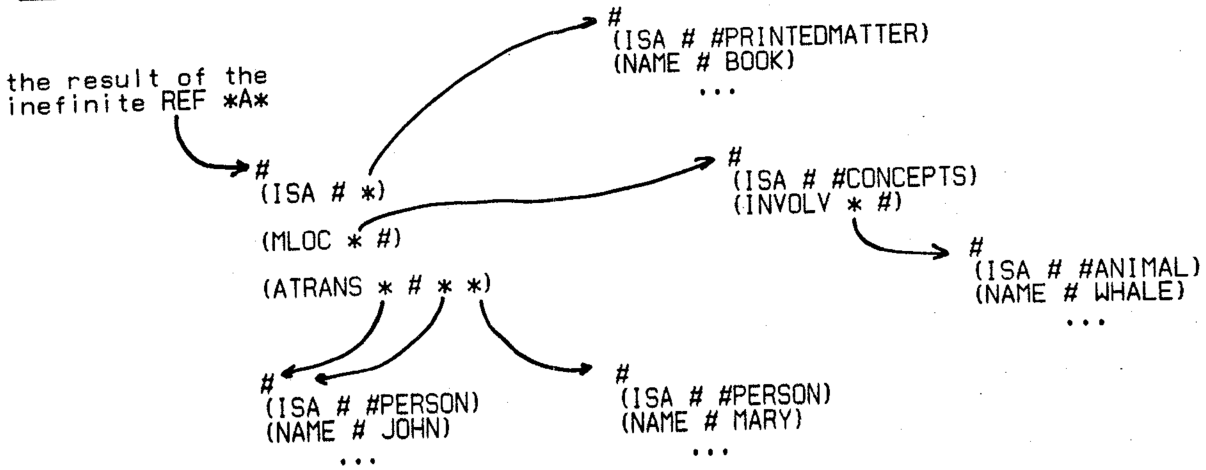



**Figure 4-9. The book about whales which John gave Mary.**

#### 4.2.5.2 REF *THE*

**sample:** Mary has *the book about whales.*

**sample:** *The nurses* were nice.

The reference signal *THE* conveys much the same information as *A*. That is, a *token* of a concept is the object of the reference. However, instead of *creating* this token, the memory is signaled to *locate* it: the definite reference in general presupposes the hearer to be able to identify some existing token with a recent RECENCY or TOUCHED tag. REF *A* on the other hand signals the introduction of what the speaker believes to be a new token to the hearer. This may not always turn out to be the case, and, although memory's response to REF *A* is always to create a new token, this token may later be identified with some existing token by the inference evaluation function described in section 7.5.

In case the reference mechanism cannot find the specific token referenced by this determinate descriptive set, a new token is created, and added to the special list, !REFNOTFOUND, to record that the referencer is concerned about its failure. Its presence on this list can later serve to frame a question of the form "What X?"

#### 4.2.5.3 NULL REF

**sample:** *Books* have pages.

**sample:** *Mary* likes *milk.*

Null reference is signaled by the absence of a REF predicate. Null reference indicates that the *concept itself* described by the descriptive set is to be the referent. Creation of new concepts which are unrecognized and decisions when the referent is ambiguous are performed as with REF *A* and *THE*. The identification of people by their names falls into this category: Bill refers to some concept which ISA #PERSON, and whose NAME is BILL, not to a token of something which ISA #PERSON and whose NAME is BILL.

#### 4.2.5.4 A SUMMARY OF REFERENCE SIGNALS

These three forms of reference signal cover most forms of declarative or imperative utterances. However, for references within interrogative utterances, the REF signal *A* requires a different interpretation. In this case, the concept to which it refers is assumed to be implicitly

existentially quantified. For instance, "Does Mary have *a* book?" means "Is there an X in Mary's possession such that (ISA X #BOOK). In this context, no token should be created. Rather the descriptive set should serve as a *template* to a pattern-matcher or proof procedure. This is not true for *THE* occurring in an interrogative form ("Does Mary have *the* book?"), since it still implied that some book exists, and that the hearer is supposed to know which it is. Hence, answering a question about it requires that it first be located by the referencer.

The memory's responses to these reference signals are still quite primitive. There are many counterexamples which form smaller theoretical classes of reference signals, and they are often quite interesting. Examples are:

*A clock* is a time-keeping instrument.

*The sky* is blue.

I play *the piano.*

But to process REF information in all its various subtle forms has not been a goal here. I want only enough capability at this time to permit the memory to get on with the other issues of reference and inferencing. However, a little more can be said concerning when the memory should be satisfied that it has enough information to "feel comfortable" that some newly-introduced token has been characterized enough.

### 4.2.6 A SPECIAL REFERENCE HEURISTIC INVOLVING REF *THE* SIGNALS

Under what circumstances does a human language user ask for more information about new tokens (of old concepts) which have been introduced to him via language, and, in particular, tokens which are *introduced* by the definite reference signal, *THE*? That is, even though the conventional way to introduce a new token is via a REF *A* signal, new tokens are frequently introduced by a *definite* reference. How is the memory to recognize when it "understands" the new reference, even though it has never heard it before? The answer to this question is very relevant to the processing involved in reference establishment in the memory, because without some heuristics, the memory would always be preoccupied with building up its store of knowledge about new tokens. What is the criterion by which a language user decides whether or not the new reference has been sufficiently described?

To illustrate this problem, consider the following two examples:

1a. Mary wants the clock.

2a. The man told me the way home.

Having heard either of these two utterances (in no particular contexts), a human language user could reasonably be expected to ask "What clock?", or "What man?" However, hearing either:

1b. Mary wants the clock which John gave Fred.

2b. The man I met in the candy store told me the way home.

the same language user would most likely not ask for more information about the clock, or the man. Why do (1b) and (2b) satisfy him, while (1a) and (2a) do not?

The heuristic which allows a language user to "be happy with what he's given" seems to be a very general one, independent of the particular feature topology of specific tokens. For if we examine enough cases where an *additional attribution* seems to satisfy the curiosity of the hearer, we must come to a very general conclusion: apparently, *almost any* additional attribution about a definitely referenced token will appease a language user's curiosity about its identity!

However, there seems to be one important proviso: the attribution must be one which could legitimately have been used *alone* to introduce the concept via a REF *A* signal. In these examples, the tokens "man" and "clock" are implicitly being introduced by the additional attribution. To illustrate attributions which do not satisfy this constraint, consider the references to a clock and a man in the utterances:

1c. Mary wants the electric clock. ---→ A clock is electric. (What clock??)

2c. The man with a mustache showed me the way home. ---→
A man has a mustache. (What man??)

Even with these additional attributions, our language user will still probably want to know more about the clock and the man because neither of these would satisfy this proviso: it is simply not possible to introduce either the man by saying "a man has a mustache", or the clock by "a clock is electric". On the other hand, it is quite possible to introduce these tokens by the phrases "I met a man", or "John gave Fred a clock".

(Of course, if the clock is a member of a known set of clocks, and it is the only electric one in the set, the problem of judging when the token is adequately specified is non-existent. In that case, the referent could be found unambiguously, and the questions we are posing here would have no meaning, since no new token is actually being introduced.)

The criterion by which the memory can judge whether or not any given attribution would adequately introduce a new token is thus the issue, and this criterion appears to be quite uninvolved. I will state it as the principle which solves this problem in the memory:

> **Any additional attribution which establishes *any* kind of connection with another *existing* concept or token in the memory will generally be sufficient identification of a new token which has been introduced by a REF *THE* signal.**

This answer turns out to be extremely simple. But this is precisely the type of problem which must first be solved before worrying about larger issues.

## 4.3 MODELING IMMEDIATE MEMORY: IMPLICIT WORD AND CONCEPT ACTIVATION

What does it mean to say that word X means Y "in the current context". That is, what is an effective definition of context as it relates to the choices made by the conceptual analyzer concerning the underlying meanings of words. More generally, how can the memory model the notion of an immediate memory which lies on the periphery of conscious thought, and how does this notion of immediate memory relate to "context" in the language sense. The answers to these questions will relate both to the analyzer's ability to choose the correct senses of words while analyzing, and to the memory's ability to establish references from descriptive sets. I have some tentative issues and solutions, and some ideas about others, and will present them in this section.

### 4.3.1 ACTIVATING IMPLICITLY-REFERENCED CONCEPTS

**sample:** John was run over by a truck.
When he woke up *the nurses* were nice.

**sample:** It's nice not to have to put the cats out tonight.
Do they know where *it* is?
Yes. (explained below!)

It is very common for speakers of natural language to leave much up to the (predictable)

imagination of the hearer. Realization of this is a recurring theme in this thesis, since much of the processing the memory engages in is designed to make explicit what is implicit (missing) conceptual information, and to elaborate upon what is already explicit. As I will try to show in this section, the ability to do this is often closely related to the process of establishing references.

Variations on the first sample above illustrates the idea of an implicitly referenced concept (or token). If a human language user hears "The nurses were nice" in the absence of any particular context, he is likely to ask "What nurses?" That is, a REF *THE* signal has given him concern that he is not able to identify this referent which the speaker believes he should. On the other hand, if he hears the sequence "John went to the hospital. The nurses were nice."; he will probably not ask this question, *even though there is still no explicit reference to nurses.* It can be argued that this mechanism is not difficult to explain, and I will tentatively agree with this by attempting to explain it. Yet, as the two samples above show, implicit references can be established by far more involved processes than even this hints at. There seems to be an extremely powerful reference-inference interaction which underlies this kind of ability in a human language user. What kind of mechanism can account for this phenomenon? Whatever it is, I want the memory to do it too! I will call it *implicit concept activation.*

#### 4.3.1.1 FREE ASSOCIATION AMONG WORDS

Our first conjecture might be that a system of free association between words of the language underlies this ability. By this explanation, hearing the word "hospital" activates the word concept HOSPITAL, and this activation automatically spreads a "charge" to its logical neighbors in this free association network, "setting" them for potential future reference. There is much compelling evidence that this is a real mechanism. But is it *adequate* for this relatively high-level language mechanism which seems to underlie our ability to cope with reference tasks as complicated as the two samples above?

I will argue that it is not, and for the following reason: although it is undeniably a real mechanism of human memory, simple free association among words is too unrestricted a phenomenon to explain most references of at the level of these two sentences. A human language user's brain simply does not resound with all the thousands of potential free

associations from HOSPITAL each time he hears the word. In the first sample, the mechanism is quite a bit more dependent upon the *meaning content* of the *rest* of the utterance in which "hospital" occurs. Contrast the first sample above with the following utterance:

> In the dark of the night, John had wallowed through the mud to the north wall of the abandoned animal hospital.

If the memory were to hear next: "The nurses were nice", and fail to ask "What nurses?", something would indeed be strange! This sentence simply does not establish a context in which we might expect to hear about nurses, even though it obviously contains the *word* "hospital".

#### 4.3.1.2 ASSOCIATION AMONG CONCEPTS THROUGH CONCEPTUAL STRUCTURES

A second conjecture which takes this failure into account goes as follows. There is still a type of free-association, except that, rather than spreading through *word associations*, it spreads through *conceptual features of the internal concepts* which the words reference (occurrence set members). In this scheme, the set of features of the particular hospital which has been referenced are assumed to have some sort of ordering from "most salient" to "least salient". Each time this particular hospital is referenced, the N most salient features would automatically be activated, and they in turn would activate other concepts they involved conceptually. This activation would proceed several levels away from the original concept.

This is a very attractive mechanism for the memory. It could be the basis of an effective definition of context (and perhaps even for such exotic phenomena as "iconic memory"). It would seem to have great potential for helping the conceptual analyzer choose senses of words in a contextually sensitive way. I will try to focus this issue a little more in the next section, but make no pretenses about having a solution or theory yet.

But for the process of reference, even this type of associative activation through conceptual structures seems to be too broad a process. In particular, what the "most salient" features are is in fact quite often governed by the meaning of the utterance in which the reference to "hospital" occurs, and to the surrounding context in general, in the same way relevant word associations are governed by meaning. For example, if we are talking about the construction of a new hospital *building*, we are not at all baffled by the reference "*The*

*cornerstone* was cracked", whereas in this context, "the nurses" is actually quite distant. On the other hand, if we are talking about John's surgery, a reference to the hospital building's cornerstone would be equally obscure.

#### 4.3.1.3 ASSOCIATION AMONG CONCEPTS THROUGH INFERENCE STRUCTURES

Because of this recurring failure to be sensitive to the surrounding *meaning*, I will make a third and final conjecture:

> **The activation which implicitly tags other concepts as having potential relevance to the "current context" spreads via the agent of conceptual inference. That is, implicit references are those concepts and tokens which are "touched" by meaning graphs which arise as conceptual inferences from the utterance in a particular situation.**

Until we explore the various types of conceptual inferences, this conjecture may seem vague. But it indeed gives the appearance of providing just the kind of restraining influence we need on this associative mechanism. The number of implicitly activated concepts and tokens will still in general be quite large, but they will have been filtered through a process which is inherently sensitive to the subtleties of the meaning content of each utterance in a particular situation.

And, as we will see, the implementation of this idea comes essentially free of cost, since the generation of conceptual inferences is a reflex response in the memory, and has many other goals besides this one. Although the memory is not yet large enough to gain a good insight into the ramifications of this approach, it appears to represent just the right tradeoff between too little and too much associative spreading of implicit references.

We can summarize this mechanism as a three step process:

1. Each new input triggers a relatively large number of spontaneous conceptual inferences
2. This new set of inferences "touches" new concepts which are *conceptually* part of the larger situation to which the utterance refers. This causes these concepts to be specially marked as having an *implicit recency*. I have called this implicit recency TOUCHED, and the marker is the value of the system clock at the time the conceptual inference which caused the concept to be touched was generated.

3. The reference mechanism recognizes these specially marked concepts as having been drawn out as part of some situation, and prefers them over other unmarked ones. Also, definite references to a concept which has been touched become understandable, because they then have points of contact with existing memory concepts.

#### 4.3.1.4 EXAMPLES

In the first sample above ("John was run over by a truck. When he woke up the nurses were nice."), the explanation of this mechanism goes as follows: having heard that John underwent a serious negative change in PSTATE, the inference arises that he may have been taken to a hospital, for the purposes of undergoing a positive change in PSTATE. Part of the algorithm by which this occurs is to be put in bed, and worked on by doctors and nurses. Notice that this is already quite a bit removed from the hospital's masonry cornerstone on the north corner of the building. Via these kinds of inferences, an implicit reference to some nurses (the ones which might be working on John because he might be hurt and in a hospital) has been made.

In this example, it may sound as though we have been forced into tracing through a quite tenuous line of inferences to arrive at this activation. This is perhaps partly the case. But some fairly strong arguments will be presented to support the claim that human language users perform a very large amount of often "tedious" processing from many different facets of the meaning content of each natural language utterance they hear. And although I am perhaps proposing that the memory has to go "too far" in a forward, predictive, direction in this example, it nevertheless seems to be that much of a language user's reasoning indeed "works forward" into hypotheses about surrounding situations, or what might happen next.

This idea requires much more research, and perhaps we must make the reference mechanism a little smarter to "meet this implicit activation mechanism half way". But the problems seem only to concern the *quantity* (depth) of inference, not the *quality* of this inference activation mechanism. In this example, the crucial step was made by an inference which drew the concept #HOSPITAL into the situation in a contextually meaningful way, namely, that it is where John went because he was hurt.

The process described there also relates to one important form of interaction between the conceptual analyzer and the memory as it concerns the context sensitive construction of an underlying meaning graph of an utterance. I will conclude this section with the promised explanation of the second sample at the beginning of this section:

```
It's nice not to have to put the cats out tonight.
Do they know where it is?
Yes.
```

Linda said the first line of the sample to Chuck one evening: she was communicating to him that she believed (a) that the cats didn't have to be put out, and (b) that it was nice that this was the case. The reason for (b) was obvious to Chuck. However, in order to understand (a), he had to ask himself "why is this the case?" To answer this was to answer the question "why do we put the cats out at night?" The answer was that if we didn't, it would lead to an undesired gift on the living room rug the next morning. Therefore, since it was no longer necessary to put them out, Chuck concluded that Linda had done something that would allow them to remain inside without messing things up. This reminded him that Linda had said she was planning to buy a litter box that morning. The inference was made that she in fact had, and that it was now in the house. Chuck was then able to ask what would have been a most obscure question without this ability of both participants to draw out implied references. They both knew immediately that the referent of "it" was the new litter box, and Linda was able to answer the question. It is hard to envision how we could account for something like this without some very powerful inference-reference interaction through "touched" concepts in the memory.

### 4.3.2 TWO MEMORY TASKS RELATED TO IMPLICIT CONCEPT ACTIVATION AND THE ANALYZER

At this point, I will describe two other processes which are logically part of the analyzer-memory interface, but which are ancillary to the main concerns of the memory. Neither has been implemented in any generality yet, but both relate to this idea of implicit concept activation I have been discussing. I want merely to point them out as useful and realistic memory tasks which are attendant problems of both conceptual analysis by the analyzer and reference establishment by the memory.

There should of course be *numerous* points of interaction between the analyzer and memory, perhaps even to the point where they blend into one process. The two I have chosen to discuss here are felt to be "typical" of the kinds of things which should eventually break this traditional analyzer-memory barrier. The first concerns the memory's role in helping the analyzer to *select senses of words* in a way which is sensitive to the context in which those words are used. No such interaction has actually been implemented in the current program, although the memory I am proposing offers a natural domain in which it could occur, and for this reason, I want to mention it. I have no general solution yet, and you are referred to [R2] for more ideas related to this subject.

The second interaction concerns the kind of processing which discovers the underlying relation between two concepts which have been associated with each other sententially. Although the interaction of this process *with the analyzer* has not been implemented this is an actual capability of the memory as it exists now.

#### 4.3.2.1 IMPLICIT CONCEPT ACTIVATION AND WORD SENSE PROMOTION

Consider the following four examples:

(1a) John asked Mary which piece of fruit she wanted.
Mary picked the apple.

(1b) Mary climbed the apple tree.
Mary picked the apple.

(2a) John was in a meadow.
The grass smelled good.

(2b) John was looking forward to getting high.
The grass smelled good.

Notice that the conceptual forms underlying "pick" are totally different in (1a) and (1b). Likewise, the PPs to which "grass" refers are also quite different in (2a) and (2b). Yet when a human language user hears any of these four sequences, he is usually capable of what appears to be an instantaneous choice of the correct sense of "pick" in the first example or "grass" in the second. How is this possible? The mechanism which underlies this ability is often called *word sense promotion.*

One of the tenets of a conceptual analyzer is that the analysis it performs be sensitive at all points to as much "context" as possible. The hope is that by doing this, most backtracking (undoing of wrong decisions) can be avoided, and what backtracking does occur will occur only because there is a genuine *conceptual* ambiguity. This appears to be the way people successfully analyze natural language. To understand how a human language user avoids backup is to gain a very important insight into the interaction of language and memory. In the above samples, avoidance of backup is synonymous with this automatic selection of the correct sense of "pick" and "grass" at the time of analysis. One would hope that the conceptual analyzer could exhibit a similar lack of confusion. We want here to examine two ways the memory could interact with the analyzer's choice of word senses.

There seem to be two related versions of this process of word sense promotion: one which relies chiefly upon implicit concept activation by inference, and another which seems to require an additional pattern-matching ability. Examples (2a) and (2b) appear to be explainable in terms of implicit concept activation: in (2b), the inference immediately arises that John WANTs to get high, and there is a class of conceptual inference specifically designed to predict a person's future actions on the basis of his current wants. In this case, that he may plan to ingest some sort of psychoactive drug is a very strong prediction. By generating these inferences, the concept "psychoactive drug" is implicitly touched by the inference process, and this activation can explain how the correct sense of "grass" could be chosen by a conceptual analyzer which is sensitive to the TOUCHED and RECENCY properties of memory concepts. Section 6.5 describes another type of conceptual inference which would draw out the desired concept for "grass" in (2a).

Notice though that, regardless of which sense of "grass" is intended, it will nevertheless be a simple PP -- all senses of "grass" are underlied by simple concepts, rather than complex conceptual *structures*. This is *not* the case in examples (1a) and (1b): both senses of the verb "pick" reference relatively complex underlying conceptual structures. Because of this, even though it might be quite possible to predict *conceptually* what Mary is likely to do next after the first utterance of (1a), or why she is climbing the apple tree in (1b), it is not clear how these inferred structures should exert an effect through this rather simple mechanism of implicit concept activation. That is, because "pick" is represented by a *structural pattern* of conceptual

information rather than by a simple concept, there must be some way of recognizing this pattern at some point as a prediction of what *words* might be anticipated next. This requirement sounds very much like the process of generation from conceptual structures back onto the words of a language. ([G1] gives a comprehensive account of the problem of generating language utterances from conceptual structures.)

I will describe two alternative approaches to this problem of choosing one word sense over another on the basis of inferred information patterns in the memory. The first stems directly from this observation of similarity between the process of word sense promotion and generation of language from conceptual structures.

## THE PROTO-SENTENCE APPROACH

Suppose we had a very fast and independent computer which ran concurrently with other memory processes. Its sole job would be the following: each time the memory generated a new conceptual inference, this program would generate from it an entity which was *almost* an expression of it in the language. It would not go all the way back to language, but instead would stop short, at the point *just before* lexical realizates for concepts in the conceptual structure were chosen. We might call this a "proto-sentence", because all that would be missing would be the particular *choices of words* which would express the language-expressible concepts which this partial generator has assembled into a proto-sentence of the language. For example, any of the words "pick", "decide", "choose", "select", could be realizates of the underlying conceptual ACT in the second sentence of example (1a) above.

By this conjecture, the explanation for our ability to expect the sense of "pick" as the one involving a reaching and grasping action goes as follows. The analyzer analyzes the first utterance "Mary climbed the apple tree". It passes the analyzed graph to the memory, where conceptual inferences arise from it. Among these are predictions about why Mary wants to be up in the tree. related inference classes). One prediction which arises is that she might desire to have an apple, and that she might be expected to perform this reaching-grasping (MOVE-GRASP) action because this could result in her having the apple. Along with all the other conceptual inferences the memory might generate from this utterance, this *predictive inference* (section 6.5), would be seen by the partial generator, which would map it into a a proto-sentence of language-

expressible concepts each of which could be expressible by a number of actual words. When in fact some word is *subsequently* perceived which could have expressed one of the concepts in this proto-sentence, it is preferred by the analyzer over others. Fig. 4-10 illustrates this analyzer-memory-generator tripartite interaction.

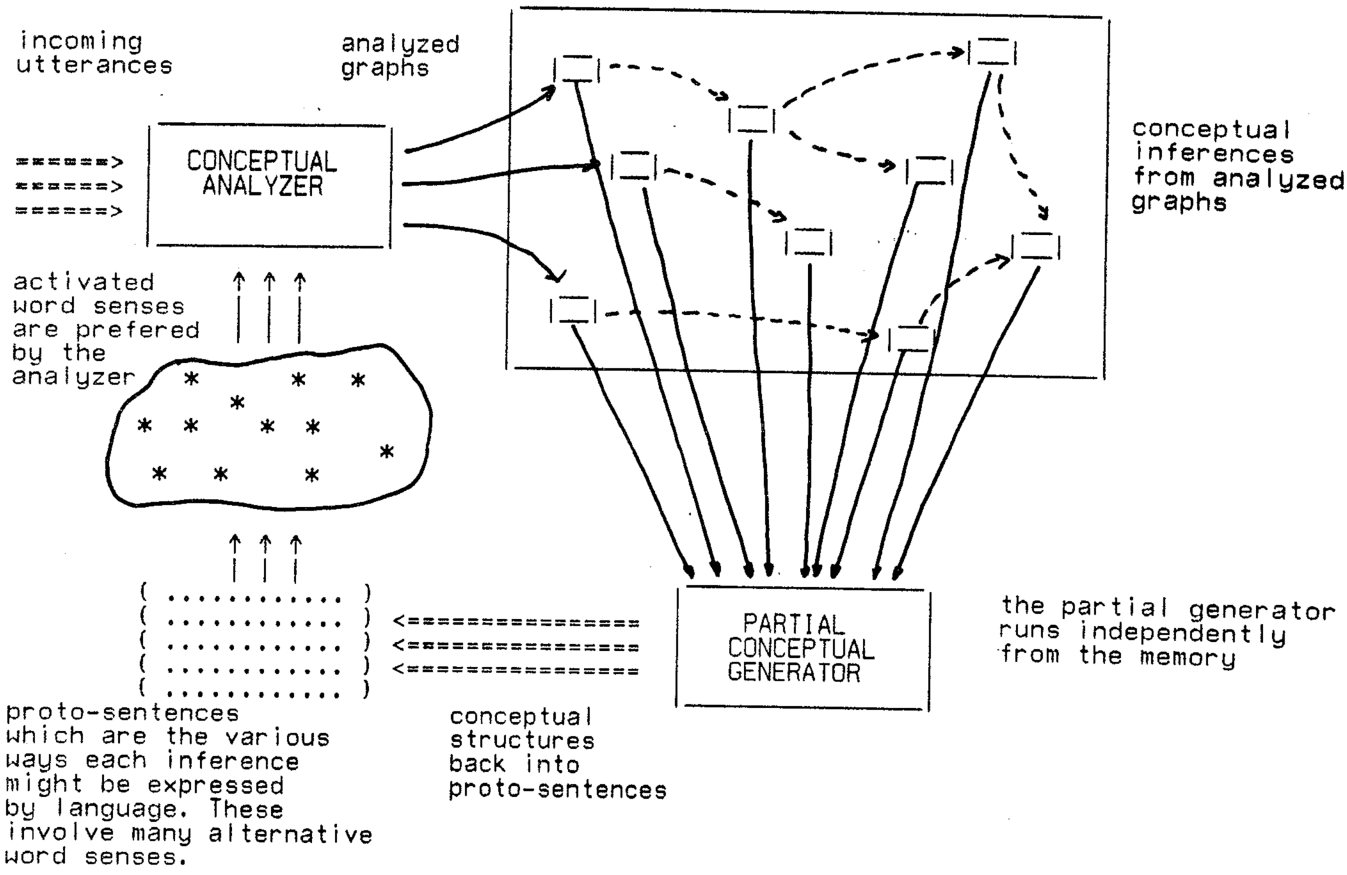



**Figure 4-10. Mapping inferences back into proto-sentences, activating many word senses.**

This approach is aesthetically very attractive, because it knits together completely the operations of the conceptual analyzer, memory and generator in a pleasing way. Of course, it would require a tremendous amount of computation. On the other hand, it is conceivable that such an interaction actually occurs in human language users: we frequently find ourselves "subvocalizing" in language that which we are thinking about conceptually. Quite often in fact, we

even catch ourselves "thinking" in well-formed language strings! It is not at all unreasonable to hypothesize that, hearing "Mary climbed the apple tree", we not only infer *conceptually* what she is up to, but that we also subconsciously put this inference (and others) back into proto-sentences which are strings of concepts, each of which could be expressed by a small set of words. As I have shown, this could be a powerful explanation of our ability instantaneously to understand word senses in subsequent utterances.

## THE CONTEXT-TESTING APPROACH

The second method by which the analyzer could be made sensitive to context in its choice of word senses is a more passive "on-demand-only" approach. In this approach, each sense of each word would have associated with it a package of memory-query tests. As each word whose lexical sense cannot be chosen on the basis of *linguistic* cues and expectancies is scanned, each test package associated with each sense would be executed. Each test would be a question asking whether certain conceptual patterns exist at that time in the memory. The conceptual patterns are precisely those memory structures which have been perceived before that time, and those which have arisen from them as conceptual inferences. Rather than a constant "sub-vocalization" of memory patterns as they arise, this approach would be more goal-directed because each sense of each word in the analyzer's vocabulary has specific tests which tell when that word sense, viewed as a unit of meaning, might be relevant. Since these tests inquire about the *meaning* environment in which each word occurs, by performing enough of them, it would seem possible to make the analyzer very wise indeed about choosing the correct sense of each word at each point in the analysis.

In our example, the tests associated with the two senses of "pick", PICK1 (to select, decide, choose) and PICK2 (to reach, grasp and pull, to pluck), might go as follows. (I am assuming that the analyzer has at least been able to decide that the "pick" is underlied conceptually by an action.) PICK1's tests look for a pattern of the form "has the actor, P, been requested to communicate to another person which of several alternative future states of the world would cause him the most pleasure?" If so, choose PICK1 as the probable sense. Otherwise, "does P desire a physical object which is currently attached to some larger object?" If so, then PICK2 is a likely candidate for the meaning of pick. In reality, a good analyzer would have to ask many more questions than these. But the idea of a "sense test package" should be clear.

This approach perhaps seems easier to implement, and far more frugal with the precious computation time of today's computers. It is a realistic approach, particularly since our conceptual formalism allows the prescribed test packages to be very concise. This approach, however, lacks the elegance of the first one which knits together the operations of the analyzer, memory and generator in a mechanically independent, yet logically very dependent way.

I simply do not know enough yet to decide between these two approaches. As with most other issues in natural language, the answer will probably not be of the all-or-none variety. Instead, it will probably prove useful, and be psychologically accurate, to employ both in some harmonious combination. Regardless of how word senses are ultimately made sensitive to context, I have a fairly specific and workable definition of context, and will conclude by summarizing it:

> **In the memory "context" is the set of all conceptual structures which have either been perceived directly, or have been generated through inference processes. This includes all concepts and tokens involved in (TOUCHED by) these structures.**

#### 4.3.2.2 RELATION PATHFINDING: ANOTHER SOURCE OF IMPLICIT CONCEPT ACTIVATION

**sample:** Mary's car is broken.
The car which Mary owns...

**sample:** The duck's bill is orange.
The bill which is a bodypart of the duck...

**sample:** John's grass needs mowing.
The grass which is part of the yard
on which the house which John is renting is located...

**sample:** John was dressed as a lion for the masquerade party.
He wagged his tail on the way out the door.
The tail which is part of the costume which John
is wearing...

**sample:** John and Mary were painting John's chairs.
Mary's chair was red, John's green.
The chair which Mary was painting...

We have been exploring the processes by which analyzed graphs and their components become structures in the memory. Another process which most language users take for granted when comprehending language is the process which infers the underlying conceptual relation between two concepts or tokens when some type of unspecified association between them is predicated sententially. Strictly speaking, the prediction the memory makes about the nature of

such relations is a form of conceptual inference which overlaps both the linguistic and conceptual domains. However, since it is more properly a subtask of *constructing* the original meaning graph for an utterance, rather than of comprehending an already-constructed meaning graph, I have chosen to discuss it here rather than in chapters 5 and 6: one of the main contributions of this process to comprehension is the implicit concept activation which arises from it. I have called this task *relation pathfinding.*

The task of relation pathfinding is the following: given two concepts or tokens which have been predicated by an utterance to bear some relationship to one another, discover the underlying conceptual relationship. Examples of this task are shown in Fig. 4-11.

| | | |
|---|---|---|
| John's hand | ------→ | the hand which is PART of John's body |
| glass factory | ------→ | a factory whose normal FUNCTION is to DO CAUSE glass exist |
| Andy's diaper | ------→ | the diaper which is LOCated on Andy's body |
| Bill's car | ------→ | the car which is OWNed by Bill |
| Mary's lawn | ------→ | the lawn which is PART of the property on which is LOCated the house which Mary OWNs. |

**Figure 4-11. Underlying conceptual relations referenced by concept pairs in language.**

This problem of determining the relation between two concepts has been dealt with in considerable detail by Sylvia Weber Russell [R5]. She has described effective procedures which attempt to combine semantic features of two nouns in permissible ways. To do this, she makes use of an abstract semantic feature system.

The memory accomplishes this same task by locating possible paths between the two concepts through conceptual information, both specific and in the form of simple patterns, rather than by a scheme of abstracted features. Much of this simple pattern knowledge is organized as the "normal function" of objects. For instance, the relationship between "glass" and "factory" in "glass factory" is ascertained by using an associative lookup to discover that the normal function of a factory is to produce a physical object. Since glass is a manufactured physical object, one possible relationship is "a factory which produces glass". Since a factory is also a physical location (a building), it might be that the composition of that building is being predicated by the noun pair. In this example, no path is found, since factories (in the memory's model) are not

constructed of glass. Concrete would, however, possibly result in an ambiguity. Resolution of this ambiguity is usually possible but involves language-specific information ("what is normally meant when 'factory' is modified by a manufactured product?").

All the samples above involve similar path-finding searches through conceptual propositions for their solution: a "hand" is found to co-occur with "person" in the conceptual proposition (PART #HAND #PERSON) and hence is interpreted as "a hand which is part of John". "Dress" is found to be an article of clothing (normally associated with a female), so "Mary's dress" receives the interpretation "the dress which Mary wears". "Bill's car" is similarly solved by discovering (NORMAL (OWN #CAR #PERSON)).

The path-finding technique which searches for a shortest-path connection between two concepts through conceptual structures in memory is a simple "expanding sphere" approach, in which the search expands simultaneously from the two points in the memory between which a path is desired. The search begins at each concept to be related, as shown in Fig. 4-12. It expands outward through the concept's occurrence sets, tagging structures (using the property SEARCHTAG) through which it passes until it encounters a tagged structure on a path from the other concept or token. The search is initiated from *both* concepts simultaneously for reasons of efficiency: if each sphere is thought of as a volume in a multidimensional conceptual association space, then, because volume is proportional to the cube of the radius, two smaller spheres which meet in the middle will generally occupy far less volume than one larger one whose radius must traverse the full concept-concept distance. Less volume means that fewer unfruitful concepts and tokens are touched, making the search more efficient.

If a path can be found, the set of propositions lying along this concept-to-concept path constitutes a possible relationship. For most problems of the nature described here, the path will normally be only a few structures long.

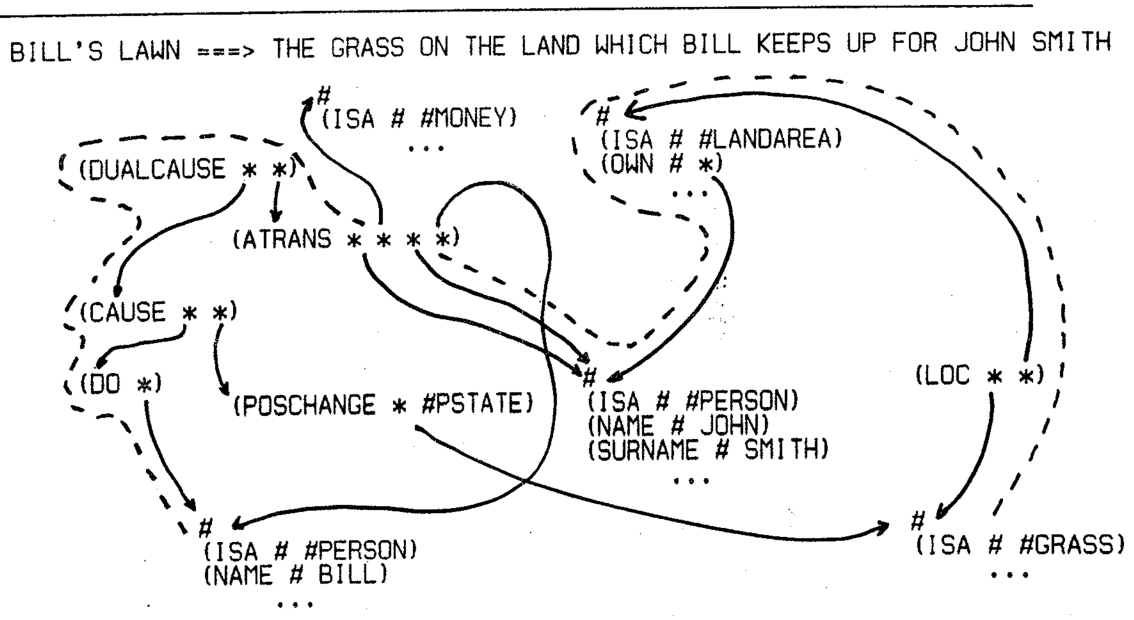



**Figure 4-12. Relation pathfinding by expanding spheres through conceptual information.**

Notice that, at the time path-finding takes place, for a pair such as "John's hand", neither the conceptual referent of "John" nor even the sense of the word "hand" may be known (the analyzer simply may not have collected enough descriptive or conceptual restrictions at that point). For this case, the path-finding algorithm must be slightly more generalized: it must keep track of N paths from possible senses of the first concept and M paths from the those of the other. In this way, solution of relations can also help the analyzer in its choice of word senses and referents. For example, for the pair "duck's bill", the pathfinder discovers a very short path through the conceptual proposition (PART #BIRD #BILL1), #BILL1 being the bill which is part of birds as opposed to a person, a unit of money, etc. However, by scanning through conceptual structures in an ever-changing memory, this could be overridden in the context established by the sentence "John's five dollar bill blew out of his hand in the park. A duck picked it up. The duck's bill ..." To override, the pathfinder would simply prefer structures with recent RECENCY and TOUCHED flags over other others along the path. The fourth and fifth samples above illustrate other such examples.

The intersection search is not a novel technique (see [Q2] for instance). What *is* distinctive

about the technique is the *nature* of the data structures through which the search occurs: in the conceptual memory, concepts never point directly at other concepts, but are always related to one another *through conceptual information.* The major advantage of this approach is that

> **Determining the relationship between two concepts by associative (pathfinding) searches through constantly changing conceptual structures permits this process to be fully context sensitive.**

By tracing out a path between the two concepts, other structures and concepts along the way will be TOUCHED, and can be implicitly activated. The third sample above illustrates this idea: suppose we hear "John's yard needed mowing. The house was in bad shape too." The average language user would rarely ask "What house?" in this situation. However, in the absence of the statement about the yard, he might well be expected to ask about the house. We again ask why this is, and seem to have a ready answer: the process of determining the relation between John and a yard has played a role in implicitly activating the concept for house. I cannot propose that it is solely responsible for this type of activation. There are undoubtedly other, more "iconic" associative mechanisms at work in the background which generate an image of a yard, house, bushes, sprinklers, and so forth. However, relation pathfinding seems to be one important source of implicit concept activation.

Let us now turn to another issue of the general process of internalization of new memory graphs.

## 4.4 SUBPROPOSITIONS

> **The average utterance is rich with information. That is, much more than the main thought is communicated. What are the sources in the meaning graph of this wealth of information? Why is it useful to extract? How is it used, once it is extracted?**

It is to the memory's advantage to recognize and extract all the sources of information within each graph in the hope that each bit will in some way contribute to the understanding of the entire conceptualization when conceptual inferences are later generated from it. This section will describe the notion of a conceptual information source by identifying three main sources within analyzed conceptual graphs. I will call any unit of information which can be extracted from

a conceptual graph a *conceptual subproposition.* The set of all subpropositions extracted from each conceptual graph will form the starting inference queue (a LISP list of memory structures, !SUBPROPS) from which conceptual inferences will be generated.

A conceptual subproposition is any unit of information which is conveyed directly by a conceptualization. The average utterance contains many conceptual subpropositions which a human language user makes unconscious but heavy reliance upon. For descriptive purposes, we can classify subpropositions into three categories:

1. explicit-focused,
2. explicit-peripheral
3. implicit

As we will see, both forms of explicit subpropositions are always complete conceptualizations, whereas implicit subpropositions correspond to single, isolated dependencies within the graph.

### 4.4.1 ILLUSTRATIONS

To illustrate these categories, consider the sentence:

> The engine of Mary's new car broke down while she was driving on the freeway late last night.

The explicit-focused proposition is: "a car engine broke down". This is the "main reason" for the conceptualization's existence, the major relation being communicated by the utterance. It will not necessarily always be the most interesting or important subproposition, however.

#### 4.4.1.1 EXPLICIT-PERIPHERAL INFORMATION

In this utterance, some of the explicit-peripheral propositions are:

1. the car is new
2. the car is owned by Mary
3. the time of the incident was late last night
4. the location of the incident was on the freeway
5. Mary was driving a car

These are additional facts the speaker thought essential to the hearer's understanding of the

conceptualization. They are "peripheral" (*dependent* in the conceptual dependency sense), and for the purposes of the conceptual analyzer. However, they frequently convey the most interesting information in the conceptualization.

#### 4.4.1.2 IMPLICIT INFORMATION

Some of the implicit propositions are:

1. the engine is part of the car
2. a car is owned
3. Mary is an actor (she performed an action)
4. the car was PTRANSed (i.e. it is moveable)
5. the car, engine and Mary were on the freeway (i.e. the actors and objects involved in an event have the event's location)

Briefly, these are very low-level propositions which conform to conceptual case restrictions, and which must strictly adhere to both the analyzer's and memory's knowledge of what is normal in the world. These typically lie on the borderline between what was said and what the hearer nearly always infers from what he heard. Although these very low-level units of information are generally uninteresting relations between objects *viewed as PP's*, they can be very interesting when viewed as *specific* relations among specific tokens and concepts in the world. For instance, although it is a fairly dull statement that a car has an engine as a part, to say that *Mary's* car has an engine is quite a different thing, because we may have thought her car was resting, engineless, on concrete blocks in her back yard. To ignore this low level source of information might be to miss this apparent contradiction. Chapters 5 and 6 will give examples of how even this low level information can lead to very important discoveries.

### 4.4.2 SOURCES

At what point in the processing, and from what points in a conceptualization are these three classes of subpropositions extracted? We will now look at each of them.

#### 4.4.2.1 EXPLICIT FOCUSED SUBPROPOSITIONS

In the sentence "Andy told Linda that Chuck went to McLean.", there are two explicit focused subpropositions:

(a) Chuck went to McLean

(b) Andy told Linda (a)

The eventual degree of strength to which the memory will believe these units of information is not a criterion for their being explicit focused subpropositions, although the memory is certainly interested in determining truth or falsity during inferencing. Conceptualization (b), being the main proposition of the thought, will automatically be examined, because it begins the inferencing for the entire graph. And, because it is nested in this top-level structure, conceptualization (a) will also be examined during inferencing. It's truth will always be dependent on its nesting in the main structure. Since the topmost structure will always be recognized as an explicit-focused information source, all other nested explicit-focused propositions will always be examined by the inferencer during the natural course of generating inferences from the larger structure in which they occur. Therefore, only the topmost explicit-focused structure is collected on !SUBPROPS, which forms the starting inference queue.

#### 4.4.2.2 IMPLICIT SUBPROPOSITIONS

In the sentence "John ate the hotdog" there are the four implicit subpropositions shown in Fig. 4-13. Each of these comes about because of a single dependency in the graph, and each is potentially the beginning of an interesting line of inference. However, subpropositions at this very low level are not actually extracted and placed on the starting inference queue. Rather, they are implicitly recognized by the inference process (molecule) which will generate inferences from the conceptualization in which they are contained.

In this example, conceptual inferences will arise that this #JOHN is an animal capable of INGESTing in the current context (for instance, he is alive and conscious at the time), that the object he ingested is INGESTable, and so forth. Typically, these inferences may have little effect on the understanding of the utterance. But they must nevertheless be made, because they are part of the rather complex event to which this simple utterance refers. By making them, the memory will stand a heightened chance of enrichening the relations among the information which the utterance conveys. For example, they may uncover unusual situations, help the reference mechanism clear up references, or generate more features of a newly-perceived concept or token. Furthermore, any contradictory information generated by conceptual inferences from

implicit subpropositions is potentially "serious", because it indicates that something unusual or incorrect has been perceived or inferred.

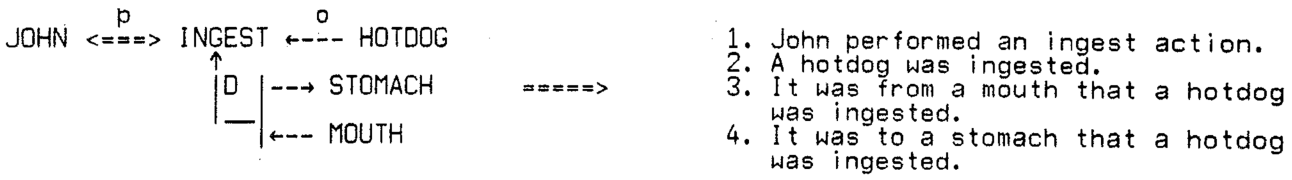



**Figure 4-13. Implicit subpropositions.**

These low level inferences are also of extreme importance in cases where a new concept has been introduced. By recognizing these implicit information sources, and by generating inferences from them, features of new concepts can be predicted. For instance, if the utterance is something like "John ate a delicious green frobifer", the memory can infer many important features of a frobifer which are not explicit in the utterance. While section 6.9.1 describes a class of inference (called feature inferences) devoted to this type of task, this is a pervasive task of *all* types of conceptual inferences.

#### 4.4.2.3 EXPLICIT PERIPHERAL SUBPROPOSITIONS

There are two sources of this subproposition type: REL-link-conveyed information, and main-link-modifer-conveyed information. Among the latter are TIME, LOCATION and INSTRUMENT.

As we saw, during reference establishment, memory concepts and their tokens are identified from analyzer-accumulated descriptive sets which are lists of conceptual features the analyzer extracts and predicts (using linguistic knowledge) from utterances. Within any particular descriptive set, there is likely to be one or more conceptualizations communicated via the REL link. Information communicated by this form provides candidates for explicit peripheral subpropositions.

Consider the sentence "John's car is red." which has the conceptual analysis shown in Fig. 4-14. The lower structure there is an example of a decsriptive set member which has been communicated via the REL link. It will be used as a descriptive set element during reference establishment to identify this particular #CAR which is being referenced.

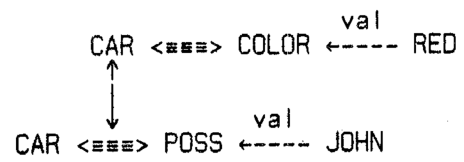



**Figure 4-14.**

At that time, one of three things can happen:

1. some token of a #CAR is unambiguously located
2. several #CAR tokens can be located from the descriptive set, in which case a new (possibly temporary) #CAR token is created
3. no concept satisfies this descriptive set (a new, possibly temporary token is also created).

For each REL-conveyed feature in the descriptive set, that feature either will or will not have played a role in the referent identification process. If it *did* play a role, then it must not be new information. Otherwise, it is likely to be a new feature from which potentially important conceptual inferences can arise. In this case, the feature, an explicit-peripheral subproposition, should be extracted for inference.

Therefore, we should recognize the following principle:

> **When some subproposition has been successfully used by the reference mechanism toward the identification of some unique concept or token, or toward some candidate set, that conceptual information (and presumably its inferences) must already be known. It should therefore not be extracted for inferencing.**

Specifically, the hope is that new information may arise which can serve to identify some existing concept uniquely as the referent of the descriptive set in case the identification failed before inferencing.

REL-communicated features constitute the first source of explicit peripheral subpropositions. Modifiers of the main conceptualization, such as TIME, TS, TF, LOC and INST, constitute the second source. For ACTs, these correspond to the incidental conceptual cases. These are collected on !SUBPROPS during the conversion process from analyzed graph to internal form (which is the topic of the next section).

Examples of where analysis of subpropositions arising from this source lead to interesting lines of inference are:

1. John went to work at 3AM. (the TIME aspect makes this potentially more interesting than it might otherwise be)
2. John woke up at 4PM. (TS, TF make this similarly of more potential interest than it might otherwise be)
3. John died on the moon. (the LOCation here is quite significant)
4. John let Mary know by tapping her on the shoulder. (here, INST will lead to potentially important inferences about John and Mary's spatial proximity)

### 4.4.3 CONCEPTUAL ADVERBS

Another example of how it can be important to begin lines of inference from many different starting points in each graph concerns "adverbial" modification of actions. For example, large number of inferences might be found to be applicable to an utterance such as "John walked down the corridor". But a new, *additional* level of inferences can also be applicable if instead we hear "John *tiptoed* down the corridor", where it is communicated not only that John performed (the same) underlying PTRANS action, he performed it in a certain *manner*. This additional information can be a most important *independent* information source within the graph. By organizing inferences, which cope with this additional information, around the conceptual predicate of the adverbial modifier, say QUIETLY, as in (QUIETLY ACTOR ACTION), then the additional inferences can arise from this structure *independently* from the central core of inferences about ordinary PTRANSing actions. This makes for a cleaner logical organization of the conceptual inference network, but it demands that seemingly innocuous information sources such as adverbial modification be recognized as independent information sources from which special lines of inference can arise.

## 4.5 STORING THE NEW CONCEPTUAL GRAPHS IN MEMORY STRUCTURES

**How is the conceptual graph which is the product of the analyzer physically integrated into the memory.?**

I have so far described (a) the reference mechanism which locates and creates concepts and tokens which are the targets of descriptive sets collected from utterances, (b) the processes of implicit concept activation, and (c) the potential information sources within each conceptual graph. These are three important components of the larger task of transforming a conceptual graph into an internal memory structure. Let us now have a look at the general flow of processing which coordinates this transformation.

The memory receives a meaning graph from the conceptual analyzer in a LISP S-expression whose form obeys the rough BNF description shown in Fig. 4-15, and Fig. 4-16 illustrates an example whose internalization we will follow.

```
<MAINGRAPH>  --->  <GRAPH> <TIMERELS>
<GRAPH>      --->  ( <MAIN> ) | ( <MAIN> <MLIST> )
<MAIN>       --->  ( <MLIST> ) | <ATOM>
<MLIST>      --->  <ROLE> <GRAPH> | <ROLE> <GRAPH> <MLIST>
<ROLE>       --->  ACTOR | OBJECT | MOBJECT | TO | FROM | <=> | <≡> | ...
<ATOM>       --->  JOHN | BALL | TIM01 | ATRANS | MLOC | LTM | ...
<TIMERELS>   --->  <ATOM> : ( <RLIST> ) | <ATOM> : ( <RLIST> ) <TIMERELS>
<RLIST>      --->  <TREL> | <TREL> <RLIST>
<TREL>       --->  ( VAL <ATOM> ) | ( BEFORE <ATOM> <ATOM> )
```

**Figure 4-15. The LISP form in which the memory receives conceptual graphs.**

MARY KNEW THAT JOHN'S FRIEND PETE HAD GIVEN JOHN A CAR.

```
((CON ((ACTOR (PETE ↔ ((ACTOR (PETE) <≡> (FRIEND VAL (JOHN)))))
        <=> (ATRANS) OBJECT (CAR REF (*A*)) FROM (PETE ... ) TO (JOHN))
       TIME (TIM01))
  <≡> (MLOC VAL (LTM PART (MARY) REF (*THE*))) TIME (TIM02))

TIM00: ((VAL T-0))          (T-0 is "now")      the ellipsis stands for a repetition
TIM01: ((BEFORE TIM02 X))                       of the ↔ link in the first
TIM02: ((BEFORE TIM00 X))                       occurrence of PETE in the graph
```

**Figure 4-16. An analyzed graph example.**

The internalization is performed by a procedure called CONVERT. Basically CONVERT is responsible for four things:

1. transforming the conceptual input syntactically from the form shown in Fig. 4-16 to the internal memory form. This includes a few instances of limited pattern matching to change the conceptual contents into more convenient memory forms.
2. creating new memory structures to store the new conceptualizations
3. calling the reference mechanism to establish references in the course of (2)
4. collecting all subpropositions on the list !SUBPROPS for subsequent inferencing

### 4.5.1 CONCERNING THE REFERENTIAL IDENTITY OF ACTIONS AND STATES

Before beginning the description of how the graph of Fig. 4-16 is internalized, it should be pointed out that the process of internalization makes no attempts to ascertain referents of *actions and states*. That is, if it internalizes "Pete gave John a car", it *always creates a new structure* to represent this action, even though the memory might in fact already know this from previous experience. In general, determining references to existing actions and states *at this stage* would be quite involved. Unlike references to concepts and tokens which *must* be established before inferencing can be of much use, unidentified references to past actions and states will generally (at worst) only duplicate knowledge the memory already has, and this duplication will be quickly detected by the inference evaluation procedure.

Thus, the task of recognizing the referential identity of incoming action and state structures is not handled at CONVERT time: each conceptualization is stored under a new memory node which may later on be detected as the same as some existing structure, and subsequently merged into it. The merge process which can do this is described in section 7.6.

### 4.5.2 THE EXAMPLE

The internalization procedure, CONVERT, goes about its task recursively. In the example of Fig. 4-16, CONVERT first locates the main link of the topmost conceptualization. Finding <≡> (an attribution), it then knows what other roles to expect to find in the conceptualization: namely the thing whose attribute is being given (ACTOR if it is a simple entity, CON otherwise), and a value, which is the rolefiller the role VAL which is nested within the <≡> rolefiller. It then extracts the

MLOC (main predicate) from the <≡> rolefiller position because, by convention, this is where the predicate is situated for all <≡> forms (POSS, LOC, etc.) Next, it retrieves the two expected rolefillers, in this case ((ACTOR (PETE) ...) TIME (TIM01)) and (LTM PART (MARY)), and calls CONVERT on each. The results which are returned (pointers to memory structures) are then plugged into the template (MLOC X Y). A new superatom is generated, and this MLOC structure becomes its BONDVALUE. This topmost structure will have no beginning occurrence set.

Having created a new structure, S, which stores this bond, CONVERT next examines the conceptual modifiers of the (X <≡> (MLOC VAL (Y))) main structure which has just been converted. In this case, it finds a TIME modification (the time at which MARY knew this), creates a new token (say C2316) to represent this time, TIM02, then associates C2316 with TIM02 and records this association on the list !TIMELIST. This association will be of use when the time relations in Fig. 4-16 are processed (to be described shortly). In addition, all modifiers of conceptualizations are placed on the starting inference list, !SUBPROPS. In this case, just this TIME structure is added.

After the modifier list has been fully processed, CONVERT finally gives this top-level MLOC structure REASONS = TRUE, TRUTH = TRUE (it believes everything it hears), and RECENCY = the value of the system clock which was recorded at the time the graph's internalization was undertaken. Notice that since the memory is not currently designed to know (or care) who the speaker is, this MLOC structure is the topmost structure of the conceptualization, and it is true simply because "that's the way it is" (REASONS = TRUE). This of course ignores many important issues which I am not addressing here. But to record who said it is simply to embed it within one higher level structure of the form (MTRANS speaker X CP(speaker) CP(self)).

As CONVERT exits at each level after having successfully converted some part of the input graph, it associates a pointer to the memory structure which was created with a pointer to the component of the graph which gave rise to that structure. It places this association on the list !REFLIST. Each time CONVERT is entered, before it begins processing it first checks to determine whether the subcomponent it is being called upon to process already exists on !REFLIST. If it is, this means that the analyzer had constructed EQ pointers and that the graph component is referrentially the same as the one already on !REFLIST. In this case, CONVERT does no further work, but simply returns the pointer to the associated memory structure created previously.

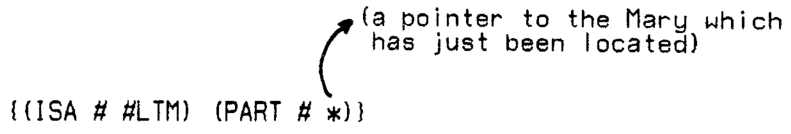
Having done all these things for this topmost MLOC structure, the task will be complete. But I must explain what happens during the two recursive calls CONVERT has made on itself to convert the CON and VAL rolefillers.

Consider the simpler of the two first: the mental location of the MLOC structure, (LTM PART (MARY)). CONVERT senses that this is a simple PP. It assumes that LTM is the NAME of some memory concept, and creates a simple descriptive set {(NAME # LTM)}. It then examines the property list of "LTM", which is the PP used by the conceptual analyzer, to locate any other conceptual features associated with this PP. In this case no other will be found. However, for a PP such as "JOHN", the additional conceptual features (ISA # #PERSON) and (SEX # #MALE) would be found. These features of the PP augment the descriptive set from which the concept is to be located. In this example, CONVERT then calls the reference-finding process, REFERENT, which in this case returns a pointer to the concept #LTM.

CONVERT next goes about collecting the modification of this PP. In this case, it finds only PART (MARY) and REF (*THE*). These tell the referencer that the concept #LTM is not the object of the reference, but rather that some *token* of an #LTM is. It therefore constructs a descriptive set which will identify the particular token. In this case, the set will consist of an (ISA # #LTM) relation and a (PART # X) relation. To determine what X is, CONVERT is again called, this time pointing to the structure (MARY). Again CONVERT senses a PP, creates a descriptive set

{ (NAME # MARY) (SEX # #FEMALE) (ISA # #PERSON) }

then calls REFERENT to locate this concept. The pointer returned becomes the X in the PART relation and at that point, the following descriptive set exists for this token of an LTM which is being referenced:




Having constructed this, CONVERT again calls REFERENT to locate the token for this Mary's LTM. The pointer thus returned is returned by CONVERT, and becomes the last slot in the topmost MLOC bond.

Recall that REFERENT will record on !REFNOTFOUND all references for which no candidates can be found. Since this would be undesirable for things like LTMs, bodyparts, or other low-level things like this to wind up on this list, there is a small number of heuristics in REFERENT to prevent this. Among them is: "when something which ISA bodypart, and which has a PART modification, cannot be found, just create it without noting it on !REFNOTFOUND."

The second slot in this top MLOC structure which CONVERT recursively calls upon itself to convert is:

```
((ACTOR (PETE ↔ ((ACTOR (PETE) <=> (FRIEND VAL (JOHN)))))
  <=> (ATRANS) OBJECT (CAR REF (*A*)) FROM (PETE ... ) TO (JOHN))
 TIME (TIM01))
```

Notice that this structure is itself an entire conceptualization, complete with TIME. The conversion process will hence be the same one which converted the MLOC structure. The first step is to determine the structural type of the conceptualization. In this case, an action is detected by the presence of the <=> main link. CONVERT thus knows to retrieve the <=> rolefiller, ATRANS, which is to become the predicate of the internal *bond* CONVERT is beginning to construct.

CONVERT next retrieves ATRANS's CASE property, which happens to be:

(ACTOR OBJECT TO FROM)

For each element of this CASE list, CONVERT seeks a matching role in the conceptual graph. There is no assumption of ordering. As each is located, its rolefiller will be isolated, CONVERTed, and collected on the ATRANS bond under construction. In case a required case cannot be located for some reason or another, CONVERT creates a new token to stand for the missing case, and marks it as being unspecified by placing the structure (UNSPECIFIED #) on its occurrence set. More will be said later about what will happen to this sort of structure during the inference process which occurs after internalization.

In this example, CONVERT will be called upon successively to convert the following graph components of the ATRANS action.

1. (PETE ↔ ((ACTOR (PETE) <=> (FRIEND VAL (JOHN)))))
2. (CAR REF (*A*))
3. (PETE ↔ ((ACTOR (PETE) <=> (FRIEND VAL (JOHN)))))
4. (JOHN)

2 and 4 will result in pointers to a token of a car, and to this person named John, respectively. 1 and 3 are in fact pointers to the same physical LISP structure, one of which will be converted trivially by finding it on !REFLIST. I will describe how this (PETE ↔ (...)) component is converted the first time it is encountered.

CONVERT again senses that a PP is being converted and begins constructing a descriptive set which will identify this Pete. On PETE's modifier list it detects a REL (↔) modifier. Since REL takes an entire conceptualization involving the concept it modifies, CONVERT again calls itself to process this conceptualization. It then substitutes "#" (symbolically -- there is no memory pointer yet) for occurrences of "PETE" in the converted result:

(FRIENDS PETE #JOHN) ---→ (FRIENDS # #JOHN)

and adds this REL-communicated feature of Pete to the descriptive set. The final descriptive winds up as the following:

{ (ISA # #PERSON) (NAME # PETE) (SEX # #MALE) (FRIENDS # #JOHN) }

CONVERT then calls REFERENT to locate this entity. If the identification successfully locates a candidate set for this descriptive set, then all these features must have been used in the identification, and hence, already known about the candidates. However, if no candidate could be located, this REL-communicated information is added to the list !SUBPROPS, and will thus become one of the starting structures for inferencing. The hope is that this might lead to inferences which would help establish the referent later.

CONVERT will then return a pointer to the entity for "Pete" thus located, and the ATRANS bond will be complete. Again, its modifications will be processed and will augment its superatom's occurrence set. Finally a pointer to this ATRANS structure will be returned to the MLOC level.

Subconceptualizations, such as the ATRANS structure in this example, are not assigned any TRUTH or REASONS, nor are they placed on the starting inference queue. This is because they will be examined anyway in the course of examining the top-level structure, and since their truth depends upon the higher structure in which they occur, no assumptions can be made at this point in the processing.

### 4.5.3 LINKING IN THE TIME RELATIONS

As each new TIME, TS or TF modifier is encountered as part of each subconceptualization's modifier list, a new memory token is created to represent this time, and the time token which appears in the graph (eg. things like TIM00) is associated with this new memory time token. This association is then recorded on the list !TIMELIST. For the TIME associated with the MLOC in this example, this association would look like (TIM02 . C3781) if the newly-created memory time token were C3781. As each new time is encountered in the graph, its existence on this list is first checked, and if it is found, the associated time token is used in the creation of the TIME, TS or TF structure which modifies the action or state structure. This is to insure time cross-references within the graph are preserved by this mapping onto memory tokens.

Although the syntax of analyzed conceptual graphs (Fig. 4-15) represents the time relation information as simply appended to the end of the graph, in fact, these relations are stored on the LISP property lists of the time atoms (like TIM00) in the graph. As each TIMnn atom is encountered for the first time, these relations are retrieved from its property list, and converted to memory structures themselves. These then become the occurrence set of TIMnn's associated memory token. Fig. 4-17 illustrates the time relationships in this graph as they will exist in the memory after internalization.

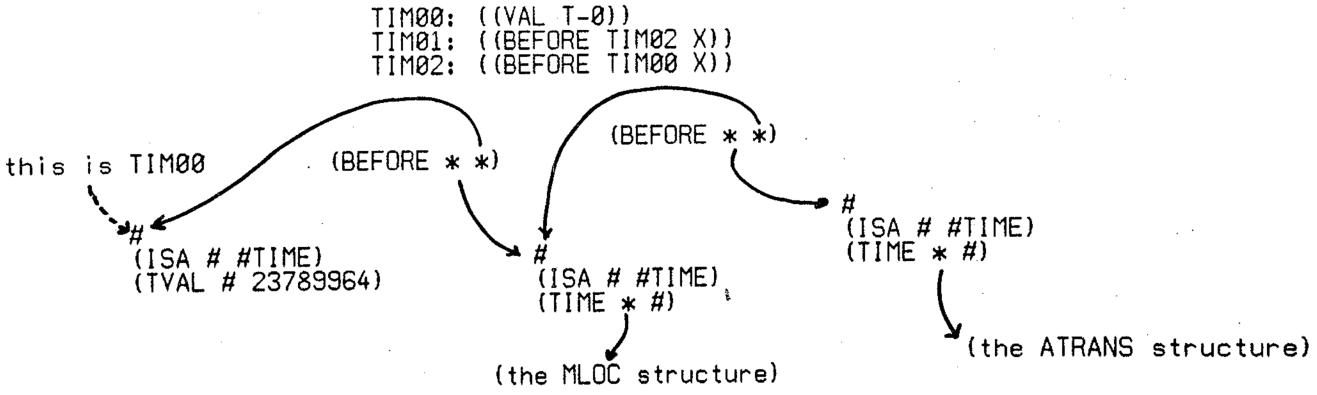



**Figure 4-17. The time tokens and their relations for the graph of Fig. 4-16.**

All deictic time references are denoted by VAL, as in

TIM00: ((VAL T-0))

Hence, as each VAL is sensed, a special procedure, associated under the property EVALTIME on the property list of the deictic time concept, is called to convert the reference to an actual point on MEMORY's internal time scale. In this example, T-0 is the only deictic time reference, and it represents the time of utterance. The actual numeric value thus obtained is associated with the memory time token via property TVAL.

## 4.6 A SUMMARY AND A PREVIEW

At this point, the internalized form is represented by a pointer to a single memory structure, and there exists a list, !SUBPROPS, of starting inference structures. In this example, these are the main MLOC structure, its TIME modification, and the REL information relating Pete and John if this was not used in identifying Pete. All TIME references have been converted into internal time tokens, and the memory is ready at this point to begin conceptual inferencing, which is the purpose of it all. Fig. 4-18 summarizes this processing which occurs between the time a graph is received from the analyzer and the time inferencing begins. Fig. 4-18 also includes a sketch of the flow through the inference and rereferencing processes as a preview of what the next chapters will be covering. At this point, it might be informative to reexamine the computer representation example which appeared at the end of the previous chapter.

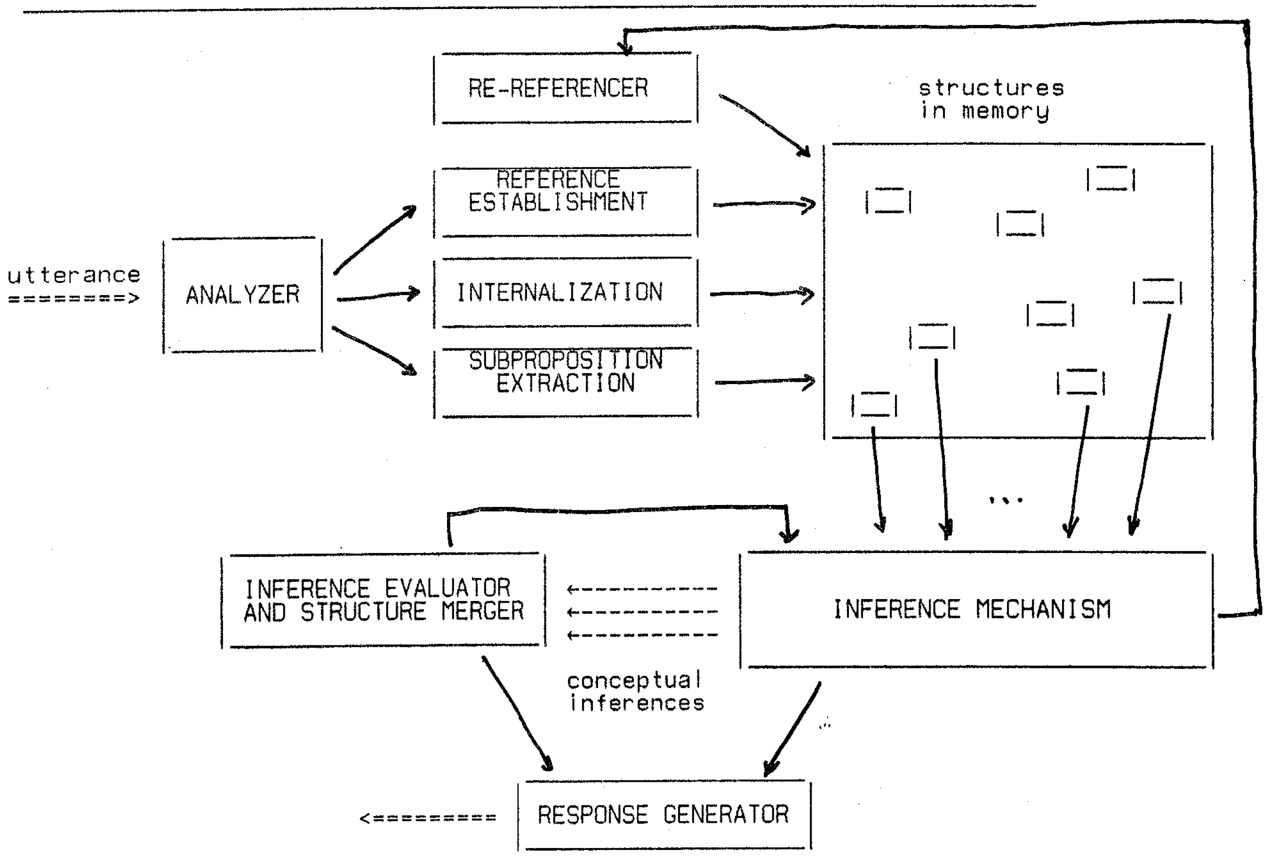



**Figure 4-18. From utterance to conceptual inferencing to response.**

# CHAPTER 5

## CONCEPTUAL INFERENCING: A SUBCONSCIOUS STRATUM OF COGNITION

In this chapter and the next, I will propose a partial theory of higher-level cognition which some might view as fairly radical in two respects. First, it prescribes quantities and qualities of computation which are not attainable (all at once, in reasonable amounts of time) on today's serial computers. This is a refreshing thought. It would be deeply disappointing if something so complex as a human language user could be modeled by a PDP10 computer! The second respect is that it may be counter-intuitive at first glance. The purpose of this and the following chapters is to convince you that your intuition is fooling you.

I will propose that this theory is a beginning step toward understanding one of the least-well understood aspects of human intelligence as it concerns natural language comprehension. This aspect is the seeming ability of a language user/comprehender to pursue only the most relevant paths of reasoning -- to "home in" on the important aspects of what language conveys in particular contexts -- while "excluding" other paths which are less relevant because of context. There are two main alternative schools of thought concerning this and related problems (in the context of language understanding), and I will briefly describe them and contrast them with this theory.

## 5.1 THE SPONTANEOUS SUBSTRATUM

The proposed theory is this: that, in order to use and understand language, the brain of a human language user does a *tremendous* amount (by today's standards of machine computation) of "hidden" computation in what I will term *inference space*. Furthermore, it does this in reaction to *every* (language) stimulus to which it is attending and which conveys any interpretable meaning content.

This reactive computation has several characteristics:

1. It is spontaneous and automatic.
2. It is subconscious for the most part. It is not normally subject to direct introspection or conscious control.
3. It is thought of as being performed by parallel, associative "firmware" in the brain.

4. It has little goal direction until certain criteria are met. Instead, its only "goal" is to increase the richness of interconnecting relations among information which has been communicated by language.

We can view natural languge utterances to which the memory is attending as giving rise to *points* in this inference space. Just as buckshot peppers a country stopsign, the subpropositions extracted from an utterance for inferencing pepper this inference space. Were these points merely to "lie where they fell", we would have no more than a passive data receptacle: the points would remain discrete, separate and unconnected from the rest of the inference space. But there are many other points in the space which exist from previous experience, and these are highly interconnected. To "understand" is to establish relations between the new and the old. Giving a system motivation to understand corresponds in this scheme to building in some sort of mechanism for spontaneously seeking out interesting relationships -- interesting points of contact in the inference space.

We may conceptualize this mechanism as one of "expanding spheres" in inference space. That is, rather than try to establish specific relations from the new buckshot points to previously existing points in the space, I am more interested in allowing the new points to blossom out in all directions in hopes of establishing *many* points of contact with other previously existing points in the space, which are simply information-bearing structures in the conceptual memory. Thus, the spheres expand simultaneously about the new points communicated by an utterance, and their horizons eventually contact horizons of other new points (the internal relationships of the utterance are being pieced together), and old points in the space (information in the utterance is making contact with existing knowledge). These points of contact constitute one source of "interesting events" in inference space, and I will have much more to say about them.

What is the interpretation of an expanding sphere in the inference space? The process is simply the automatic reaction to a new unit of information: "where does this fit in with what I already know; what interesting points of contact does it make with other information?" It is a reconstitution and elaboration of the content-rich situation alluded to by content-lean utterances.

### 5.1.1 CONCEPTUAL INFERENCE: THE EXPANSION FORCE

The force behind this expansion -- that process by which a sphere expands -- is *conceptual inference.* Since this expansion occurs at a very low level -- without conscious control -- there is no tangible goal-direction until an interesting point of contact is reached. But at that point, more conscious, goal-directed, computation can begin. The "dimensionality" of the inference space is finite, and we may think of each dimension as corresponding to a particular type, or class, of conceptual inference. The expanding spheres represent spontaneous exploratory inferences in *every* dimension of the space.

This spontaneous mechanism is not a complete theory of language-related cognition; I am by no means proposing this as the only mechanism of understanding. Rather, the conjecture is that it constitutes a necessary and low-level stratum in the cognitive process, and it is this stratum which feeds other more goal-directed strata with potentially interesting tasks in particular contextual environments.

As we will see, the theory entails an apparent degree of "wastefulness" at this level of cognition, since it is basically a "bottom-up" exploratory process which homes in on interesting events, setting up more "top-down" processes when such events occur.

> **This theory addresses what can be called the "lower-upper level" of cognition: it is the low level underpinning of our ability to "think" via language.**

### 5.1.2 A BRIEF ILLUSTRATION

To illustrate in a more concrete way what I want to be able to do in the conceptual memory, I will give a sketch of the kinds of spontaneous reasoning which, I propose, occurs in reaction to a simple utterance:

John McCarthy went to Boston.

We would want the memory's stream of consciousness analysis of this utterance would go something like this: "He went to Boston, eh? That means he was in Boston, and he probably wanted to be there. Why would he want to be there? Probably to do something which requires

his personal presence, like talking to some other high-up. Oh yes, he probably went to talk to someone like Minsky at MIT... about grant money or new research proposals, or something like that. That's understandable. Of course, it could just be a vacation. How did he go? Probably by flying. That's ok, he has the means, and there's no air strike on. Wait a minute. I thought he was giving a talk here tomorrow. Either he'll be back then, or it's been called off, or something. Better find out....."

This is the sort of reasoning we ultimately want a fully attentive conceptual memory to be able to perform. Notice how many assumptions were made about what is normal and how, by making them, much other information was drawn into the analysis. All this information is the sphere in inference space which develops about the original utterance. Even though some of the assumptions made may have been incorrect, the sphere at least forms a framework within which many other valuable discoveries can be made.

This theory is not concerned with pinning down *one* explanation, or with pursuing just one line of reasoning. Instead, the idea is to elaborate each utterance in as many directions as possible with the hope that some of the elaborated information fits together with, or contradicts other elaborations made previously. For instance, because we predicted his trip was by air, we implicitly predicted also that a significant amount of money was involved. This provided no conflict with our knowledge of McCarthy. However, the utterance may have sounded peculiar about our utterly broke friend Bill who lives in the hills. "How did he get that kind of money?", "Why did he want to be in Boston.", etc. I hope to demonstrate that these points of contact with other knowledge are possible only when this kind of spontaneous expansion by conceptual inference occurs.

### 5.1.3 ABOUT THE PSYCHOLOGY OF IT ALL

Our task is to apply this notion of spontaneous expansion in inference space to the problem of understanding the meanings of language utterances in particular situations, or *contexts*. To an extent, then, we are modeling a human language user, or, more succinctly, a person! To understand how a person might use and understand language, we must ask questions about how he understands the world about him in general, what motivates him to act, what he knows, and so forth. This need to model people is realized in fairly overt ways by some of the types of conceptual inferences I will propose as primary dimensions of the inference space.

Some of the central classes of conceptual inference in the current theory implement a "naive psychology". That is, they are based upon a "layman's" view of cause, effect, motivation and intentionality. Although such a basis is defensable on purely philosophical grounds, I am not modeling the "deep psychology" of a person, and hence have no quarrel with those who would criticize certain of the conceptual inferences on this basis. Instead, I am interested in the efficacy with which a certain class of conceptual inference can account for a human language user's ability to understand utterance X in situation Y. By asking enough questions about enough X's and Y's, I have arrived at a fairly compact set of conceptual inferences which seem to lie at the center of much language understanding. That they form a "naive psychology" is only of incidental significance to the theory, whose main goal is to explain language comprehension.

We must first be able to account for the simpler activities of language understanding before we tackle those which require a deeper analysis. We must first develop an understanding of how to deal with the "rule" before we can approach "the exception". That is, there must first be some critical mass of knowledge about simple cases before we can expect to grasp the subtler issues of language comprehension. The hope is that the critical mass established by this embryonic theory can eventually be embedded within a larger, more comprehensive one without massive dismemberment. That is, although a naive psychology may indeed be ultimately inadequate, the hope is that it can be *extended* rather than discarded as new issues arise.

### 5.1.4 THE FLYWHEEL EFFECT

One very natural application of a theory of conceptual memory of the sort I am proposing is to the comprehension of simple stories. We may view a story as a sequence of utterances (sentences) such that each utterance bears at least one relation to some other utterance of the story. Usually, the connections between any one utterance and others in the story are quite rich. Normally, the entire reason for the existence of a line in a well-written story is that it relates to, and explains, other ideas in the story. Each line serves to enrich the connectivity of the information content of the story.

Because it is the purpose of a story to create and preserve this richness of connectivity among its constituent information, we might view it as possessing a certain momentum at each point. That is, each thought in it tends both to explain some things, to raise questions about

others. There is a logical continuity, which the author helps the reader to focus upon by what he chooses to include and how he includes it, and this is an assumption the reader makes before trying to comprehend a story. I will call this logical continuity and momentum the *flywheel effect.*

The essence of story comprehension rests upon the reader's ability to stay in synchronization with this effect. This involves such things as understanding actors' motivations, recognizing cause/effect interrelationships, forming many "mini-hypotheses" (quite often only subconsciously) about what might happen next in a particular situation, then verifying them as the story unfolds. In other words, the comprehender expects every idea to fit in, and the implied task of fitting everything together is a universal goal of story comprehension.

My conjecture is:

> **The fundamentals of understanding a story are rooted in the spontaneous expansion in a "multi-dimensional" inference space of each new thought as it arrives.**

I pose this thought here simply to provide a similar, but more application-oriented, perspective on the nature of this phenomenon of spontaneous inferencing being proposed. How we can get a computer program to be able to stay in synch with this logical flywheel, will evolve as the various kinds of conceptual inference are presented.

### 5.1.5 TWO OTHER APPROACHES TO UNDERSTANDING, BRIEFLY

I have mentioned the existence of two significantly different approaches to understanding. These are the "inference-on-demand" approach and the "demon" approach, both of which acknowledge the importance of some sort of inference capability. The crucial differences concern *when* and *for what purposes* inferences should be made.

The main precept of the inference-on-demand approach is that inferences are very costly, and should be made to satisfy only the very specific, intermediate goals of larger, goal-directed processes which know what is interesting to do in all sorts of contexts. That is, an inference is not something which arises spontaneously as a person comprehends an utterance, but rather is something to be called upon to answer a specific question for which the answer is not immediately attainable from the "data base". This is characteristic of a question-answerer, or theorem-prover, or "planner" (in the MICROPLANNER sense [S14]).

The main inadequacy of this approach in so ill-defined a task as "understanding" is that there can in general be no de facto higher level goals until it has been discovered what comprise the *potentially interesting* aspects of a particular situation. That is, at this low level of higher cognition, there is no real *source* of demand for inference. Hence, although "inference on demand" systems are quite relevant to tasks such as sentence analysis and generation -- and to very specific aspects of understanding in a conceptual memory, once interesting tasks are uncovered -- they are not sufficient for the main purposes of a conceptual memory: to enrich all sorts of interconnections among information.

Charniak's "demon" approach [C1] is another distinct theory of language comprehension, and it lies slightly closer to the one I am proposing than a pure inference on demand scheme. Basically, a demon is a process which can be activated by certain combinations of situations in the same sense that a conceptual inference is triggered by combinations of conceptual information. The idea is to spawn demons at each point in, say, a story. The demons will "lie in wait" until they detect that they are applicable to some later event or situation, at which time they become active, releasing their potential to influence the interpretation of the pattern which has activated them. In this way, a continuity is maintained between information which spawns demons and information whose interpretation is later affected by previously spawned demons. The demons come out of suspended animation when patterns with which they are equipped to deal are detected.

The notion of a demon is a good one, and is probably necessary to good understanding systems. But demons are often guilty of "playing their cards too close to their faces." That is, since a demon's potency is stored only as a *potential* for influencing the later interpretations of conceptual information, the information it bears is not readily available to other language-related processes which could make their own idiosyncratic use of it to discover relations which were not the original intent of the demon, but which nevertheless depend in important ways upon the information it contains.

Herein lies the fundamental difference between a demon and a conceptual inference. A demon contains only a *potential* for exerting an influence. Because of this, it is not of much use in drawing out -- in making explicit -- the *implicit surrounding context* of an utterance. From the standpoint of the conceptual analyzer alone, this is a very important function of a memory.

Rather than mask all this implicit information in the form of demons which mete out their services when they become "applicable", I have taken the approach that it is useful to draw it out as much probablistic information as possible at each point -- to lay all the cards on the table for everyone who might be able to use them to see. This makes all the more likely the discovery of interesting underlying relationships (which could not have been anticipated beforehand).

There is another more pragmatic argument in favor of drawing things out explicitly, as opposed to bottling them up as potentials in the form of demons. It is this: the process of spawning a demon is essentially the same as generating a conceptual inference: in general, the same quantities of testing will be required to decide when a certain demon may be applicable -- when it might be relevant to spawn. As long as the applicability tests are essentially the same, why not go ahead with the probabilistic inference at that point, thereby making explicit its potential effects (interactions) with subsequent inputs? Aside from the obviously rapid consumption of computer storage space, this requires negligible additional effort. Doing this has the same desirable net result as a demon-based scheme, and has the potential for making far richer connections among the information which is processed this way.

## 5.2 WHAT IS A CONCEPTUAL INFERENCE?

**The heart of computer understanding of language is the expansion of conceptual structures in inference space, by the mechanism of conceptual inferences. What is a conceptual inference?**

When a language user hears

| | |
|---|---|
| | Mary kissed John in front of Sue. |
| and concludes: | Sue became extremely jealous. |
| | John sold his car. |
| and responds: | I didn't know John owned a car. |
| | Bill took an aspirin. |
| and asks: | What's wrong with him? |
| | Mary wants a book. |
| or<br>and asks: | A book about what? |

what has gone on inside his head? Why should hearing one thing elicit a response about something else. What mechanisms are responsible for this?

As I have proposed, this process is underlied by an inference reflex which spontaneously accesses beliefs and knowledge about the world in reaction to each unit of information in each conceptual input attended to. It is the purpose of this section to define this notion *conceptual inference* and show how conceptual inferences organized in a conceptual inference network can behave in this manner. To begin, it will be useful to contrast the notion of a conceptual inference with more traditional notions of inference and logical deduction, with an emphasis on their quite different roles.

### 5.2.1 "CONCEPTUAL INFERENCES" VS. "LOGICAL DEDUCTIONS"

In its broadest sense, a conceptual inference is simply a new piece of information which is generated from other pieces of information, which may or may not prove to be true in the world which it models, and which is "believed" by the inferencer, not in a black and white sense, but rather to a "fuzzy" degree (say, a real number between 0 and 1, rather than TRUE-FALSE). Since the intent of inference-making is to "fill out" a situation which is alluded to by an utterance (or story line) in hopes of filling in missing information and tying pieces of information together to determine such things as feasibility, causality, and intentions of actors at that point, many of the conceptual inferences may turn out to be useless. That is, the process of generating conceptual inferences is inherently a computationally wasteful process, because its intent is to *discover* what is interesting in a particular context.

A conceptual inference can be distinguished from the traditional notion of a logical deduction in a formal system in the following respects:

(1) Inferences are a "reflex response" in a conceptual memory. That is, *the main definition of "processing conceptual input" is the generation of conceptual inferences from it.* This means that there is always a deep-rooted motivation to generate new information from old. In a more formal theorem-prover or question-answerer, deductions are performed only upon demand from some external process. Someone (something) else has already decided what is and what is not interesting or useful to do. Normally, the

uses to which formal deductions are put are highly directed in the sense that a well-defined goal exists, and a path from some starting conditions (axioms) via transformations (theorems) on these conditions to this goal is desired. In this application extreme care must be taken not to "wander off" this path too far. For this reason, a recurring issue in formal deductive systems concerns the problems of search space restricting heuristics. Conceptual inferences on the other hand have very little direction. They are generally made "to see what they can see". The "goal" of inferencing is rather amorphous: make an inference, then test to see whether it looks similar to, is identical to, or contradicts some other piece of information in the system. When one of these situations occurs, the memory can take special action in the form of discontinuing a line of inferencing, asking a question, revising old information, creating new causal relationships, or perhaps invoking some sort of higher level, goal-directed belief pattern which will begin imposing a special interpretation upon what it subsequently perceives. The problems of severely narrowing the search space in hopes of establishing a path to a goal exist, but are not nearly so acute as in a goal-directed theorem prover: there is neither a "path" or a "goal" until one of the situations described occurs.

(2) An inference is not necessarily a logically valid deduction, and will quite often lead to apparent contradictions. This is in fact one facet of what it means to discover what is interesting about a particular utterance in a particular situation. But this means that the new information represented by the inference might not bear any formal logical relationship to those pieces of information from which it is generated. In order to understand language, we must model that horrendously illogical cognitive entity, the human language user -- both the processes he uses, and the substance of what those processes yield.

(3) It makes little sense to talk about believing an inference in an all or none sense. Rather, we must talk about the *degree* to which a conceptually inferred information (or *any* information, for that matter) is likely to be true -- a measure of how strongly the inferring mechanism beleves it. It is imperative that the memory retain and propagate measures of the *degree* to which a piece of information is likely to be true. The

memory in which conceptual inferences occur must be designed with the idea that *no* information it contains is inviolably true, but rather that "everything is just a guess, and some guesses turn out to be better than others".

### 5.2.2 THE CONCEPTUAL INFERENCE EVALUATION PROCESS: A PREVIEW

Chapter 7 is devoted to the details of how the program generates and evaluates inferences, and how, mechanially, they link together to form new connections in inference space. However, as we cover the various classes of conceptual inference, it will be useful to have a vague notion of what happens to each inference *after* it arises.

When a new inference is generated, one of three conditions can apply:

(1) the new inference can match something else in MEMORY. When this happens, the new information is said to *confirm* the old. This is one of the most fundamental events in the understanding of more than one utterance (ie. a story), or in the understanding of relationships within one complex utterance. It gives rise to a *merge* event, which is one form of contact point in inference space.

(2) the new inference *contradicts* (is incompatible with some old information. This means either that something is conceptually peculiar about the utterance or that the memory has made an incorrect decision about some referent or has generated a probabilistic inference which turns out to be unlikely. The ability to detect contradictions is another important aspect of understanding, and contradictions are another form of point contact in inference space.

(3) the new inference can neither be determined to contradict nor confirm old knowledge. In this case, the new information is simply remembered, and is said to *augment* existing knowledge. However, this new information can have profound effects on other aspects of understanding (in particular, the identification of referents, and the determination of time relationships).

## 5.3 THE MAINSTREAM CONCEPTUAL INFERENCES

**What are the main dimensions of inference space. In other words, what general classes of inference are there, based on their utility to language comprehension? How does each contribute to understanding language utterances?**

Although all inferences have many characteristics in common, their *utility* in the flow of processing which expands structures in inference space is generally quite distinctive. Because of this, it is helpful both theoretically and programmatically to distinguish inferences by type. This classification can help clarify the usefulness of a particular inference and how each might be said to contribute to the overall goal of understanding.

The main framework of the theory consists of the following 16 classes of conceptual inferences:

1. **specification inferences:** what are the missing conceptual components in an incomplete graph likely to be?
2. **causative inferences:** what were the likely causes of an action or state?
3. **resultative inferences:** what are the likely results (effects on the world) of an action or state?
4. **motivational inferences:** why did (or would) an actor want to perform an action? What were his intentions?
5. **enablement inferences:** what states of the world must be (must have been) true in order for some action to occur?
6. **function inferences:** why do people desire to possess objects?
7. **enablement-prediction inferences:** if a person wants a particular state of the world to exist, is it because of some predictable action that state would enable?
8. **missing enablement inferences:** if a person cannot perform some action he desires, can it be explained by some missing prerequsite state of the world?
9. **intervention inferences:** if an action in the world is causing (or will cause) undesired results, what might an actor do to prevent or curtail the action?
10. **action-prediction inferences:** knowing a person's needs and desires, what actions is he likely perform to attain those desires?
11. **knowledge-propagation inferences:** knowing that a person knows certain things, what other things can he also be predicted to know?
12. **normative inferences:** relative to a knowledge of what is normal in the world, determine how strongly a piece of information should be believed in the absence of specific knowledge.
13. **state-duration inferences:** approximately how long can some state or protracted action be predicted to last?

14. **feature inferences:** knowing some features of an entity, and the situations in which that entity occurs, what additional things can be predicted about that entity?

15. **situation inferences:** what other information surrounding some familiar situation can be imagined (inferred)?

16. **utterance-intent inferences:** what can be inferred from the *way* in which something was said? Why did the speaker say it?

I have based most of the discussions of these various inference classes, and of the processes which implement them in the computer program, on very simple examples. The reasons for this are twofold: (1) it helps to abstract and isolate certain processes which might otherwise be obscured in more complex examples, and (2) the examples will, for the most part, be easily representable in the representational formalism and memory structures which have been described, and hence will illustrate the modest -- but actual-- capabilities of the computer program. However, having read about each inference class in the context of the simple examples, the reader is urged to attempt to apply that class to instances of "real world" language about him in order to develop a feeling for the potential powers and/or weaknesses of each type, and the processing which implements it. A bit more will be said about the relative scope of these inference classes at the end of chapter 6.

### 5.3.1 AN IMPORTANT CAVEAT

The computer program which implements most aspects of this theory exists and runs. However, from the discussions of the kinds of things it does, one should not be misled into believing that a truly vast system yet exists. Where some issue is discussed as though the implemented memory can currently handle thousands of cases, more often than not, it will in fact only cope with a handful of examples. But I am confident that this is a failure of *data*, not of *process*. As time passes, and the theory evolves, so will the data. It is too early at this point to spend too much time encoding tomes of specific knowledge about the real world. We are still fumbling with the more basic processes of language understanding.

One final comment: I reemphasize that I will be discussing *classes* of inference: (1) how they are *useful* for understanding, (2) *when* they are applicable, (3) how they are *achieved* in the computer formalism and program. In a sense, then, rather than talk about *specific* inferences, I will be examining the when, where, and why of doing things certain ways. By doing this, cubby-

holes of processing will be established to which I can point and say "Yes, you're right, this particular case hasn't been discussed, but here is where it fits in the formalism and processing sequence." The rule of the game, therefore, is not to say "You can't do that", because what I describe *can be* and *has been* done, to varying degrees of success. Rather, you may say "That isn't quite right", or "You've oversimplified a very deep philosophical problem", or "This won't account for X", or, "Yes that's nice, but you'll never get it all to run at once on a PDP10 computer in reasonable times". If you play the game this way, you'll quite often be correct, and we can all laugh together!

Let's now look at the inference classes.

## 5.4 SPECIFICATION INFERENCES: PREDICTING AND FILLING IN MISSING CONCEPTUAL INFORMATION

**sample:** John picked up a rock.
He hit Bill.

**sample:** Bill was driving home from work.
He hit John.

**sample:** John and Bill were alone on a desert island.
Bill was tapped on the shoulder.
It was probably John who tapped him.

**sample:** John bought a cake mix.
It was likely a grocer with whom John traded money for the cake mix.

**sample:** Where was John Tuesday evening?
I don't know for sure. Probably at home.

**sample:** Mary accidentally dropped a sledgehammer on Bill's toe.
She apologized.

Language tends to be as economical a means of communication as possible. And it is so deeply ensconsed in people's knowledge of what is normal in the world that it rarely is used to communicate the obvious. Instead, it serves to relate new combinations of information to others who have not directly experienced them. One consequence of this phenomenon is that the conceptual structures of utterances are quite often incomplete. That is, where the underlying conceptual representation of an utterance would predict the existence of an ACT case or state argument, there was no actual reference to such information in the utterance. The speaker of an utterance simply assumes that the hearer is capable of "filling in the details" as part of his comprehension.

As we saw in chapter 2, there is a well defined set of conceptual cases for primitive actions, and equally well defined arguments for state relations. Furthermore, all action cases and state arguments are conceptually obligatory. Unlike syntactic cases, whose presence or absence is often optional and of little consequence, a conceptualization is simply incomplete without them. Without all the conceptual slots filled, the hearer of an utterance simply cannot fully imagine the entire situation to which the utterance alludes. This is intrinsic to the notion of "conceptual case". This section will demonstrate the importance of giving the memory this capability to make good contextual guesses about missing and unspecified information, and will describe how this capability has been implemented.

### 5.4.1 WHY DO IT?

We might well ask "If the hearer is capable of filling in the details in the first place, why should he bother to do it?" That is, what good comes from completing a meaning graph with "internally-generated" information which the hearer supplies *himself*? Can it possibly lead anywhere? The answer is an emphatic "Yes", for two reasons. First, how is the hearer to know whether or not he *can* in fact complete the meaning graph without trying! In cases where he cannot, a human language user frequently asks a question of the speaker. The commonness of question-asking based on missing information is testimonial that this process is a vital part of understanding.

The second reason is less superficial: by applying his knowledge of normality to the task of filling in missing information, the hearer generates *specific instances* of that normative knowledge. These specifics can then interact with other knowledge in entirely different ways from instance to instance: the prediction of missing information can be the beginning of important lines of inference. Let us call the process which attempts to specify missing or incomplete information in a meaning graph *specification inference.*

### 5.4.2 DETECTION AND MARKING IN THE CONCEPTUAL ANALYZER

To illustrate this process of specification consider the utterance "John hit Bill." In particular, imagine how the analyzer deals with it: John is recognized as the actor of a hitting action in which Bill is some sort of effected entity, possibly the conceptual object of the hitting action.

But the action of hitting is not conceptually primitive. Rather it consists of a PROPELling of an object, X, toward some goal, resulting in the physical contact of X and the goal. The conceptual template in the analyzer's dictionary which defines "hit" therefore predicts that the sentential object, "Bill", is not really a conceptual object, but rather that he is the affected entity (the directional goal) of the propelling action in the hit template. The left graph in Fig. 5-1 depicts the state of the analysis after the analyzer has located and partially filled in this "hit" template.

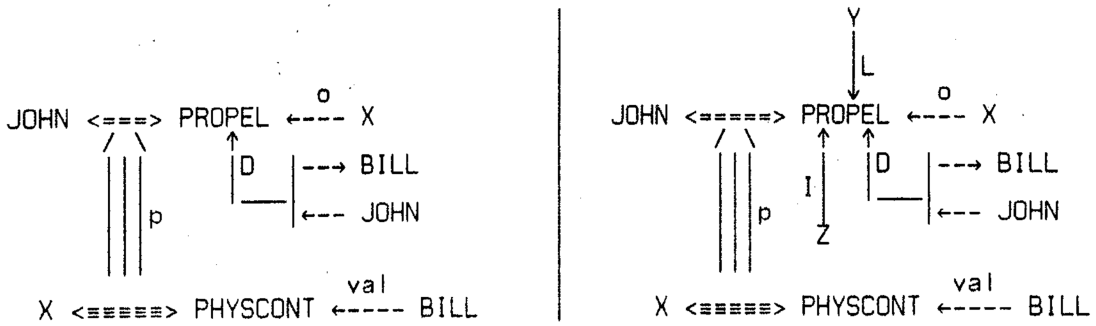



**Figure 5-1. Missing information in the utterance "John hit Bill".**

The underlying ACT is PROPEL, which requires, as do all the primitive ACTs, certain conceptual cases obligatorily, and only those. For PROPEL, those cases are ACTOR, OBJECT, DIRECTIVE (TO and FROM) in addition to the ubiquitous TIME, LOCATION, and INSTRUMENT cases, which are requirements of all acts. But notice that the analyzer, by using its linguistic ability, has been able to supply *only* the ACTOR, D-TO, D-FROM and TIME (and this, only partially specified as some point before "now"). The three remaining cases, OBJECT, INST and LOCATION remain unspecified. In the terminology of the analyzer, this means that the requests which arose during the analysis to locate and attach these missing cases to the meaning graph are still pending at the end of the analysis (see [R2]).

The analyzer therefore detects these and creates "dummy" cases and case fillers, and marks these missing entities by placing an (UNSPECIFIED _) in their descriptive sets. These UNSPECIFIED markers will thus become part of the occurrence set of each unspecified entity.

In general, the descriptive set will consist of more than the UNSPECIFIED marker, since the analyzer is usually able to glean at least a few conceptual features about the unspecified entity (for instance, the sex of a person from linguistic pronominal clues, and so forth.)

### 5.4.3 THE SPECIFICATION PROCESS

The meaning graph which the memory receives for "John hit Bill" is shown in the right of Fig. 5-1. During its processing in inference space, we want the memory to detect the X,Y and Z in Fig. 5-1 as unspecified, then consult its language-free world knowledge to make a "best guess" at these cases in whatever the current context of this utterance happens to be.

The meaning graph is first internalized in the memory's data structures, and part of this internalization consists of isolating all the subpropositions which have been communicated by the utterance. For this graph, the subpropositions are those shown in Fig. 5-2. In Fig. 5-2 I have broken with the standard "pound-sign" notation for internal tokens and concepts in order to make things more readable.

```
P1: (PROPEL JOHN X JOHN BILL)
P2: (LOC P1 Y)
P3: (INST P1 Z)
P4: (CAUSE P1 P5)
P5: (PHYSCONT X BILL)
P6: (TIME P4 T)

T represents a time atom for which the relation (BEFORE T Tnow) exists,
Tnow being the time atom which represents the moment of utterance.
```

**Figure 5-2. Subpropositions in "John hit Bill".**

#### 5.4.3.1 DETECTING MISSING INFORMATION IN THE INFERENCE MONITOR

Having been isolated, the memory structures P1-P6 will form the starting inference queue: the buckshot points in inference space. This means that each of P1-P6 will eventually come under the scrutiny of the inference monitor which will apply suitable inference molecules to them. This is the process by which inferences are generated to expand the points in the space into spheres: each structure in the starting inference queue will give rise to numerous inferences. These are appended to the end of the queue for later expansion, and will, in turn, give rise to other structures, and so on.

As the inference monitor picks up the next structure, S, from this ever-expanding inference queue for inferencing, the monitor *first* scans the S's bond, looking for entities in it which are marked as UNSPECIFIED. That is, just before applying an inference molecule to S, the monitor

checks for any unspecified information in S. Since unspecified information has been tagged by the analyzer as UNSPECIFIED, this scan consists of searching for an (UNSPECIFIED Xi) on the occurrence set of each Xi in the bond, including the conceptual predicate which may itself be unspecified (a dummy DO). If no entity in the bond is found to be unspecified, the monitor proceeds to locate and apply the appropriate inference molecule to the structure. However, if one or more unspecified entities are found in the structure's bond, the inference monitor interrupts and applies a *specifier molecule* to S.

In our hitting example, this type of interruption will occur for the structures representing P2, P3 and whichever of (P1 P5) is examined and successfully specified first by the inference monitor. This detection process is schematically illustrated in Fig. 5-3.

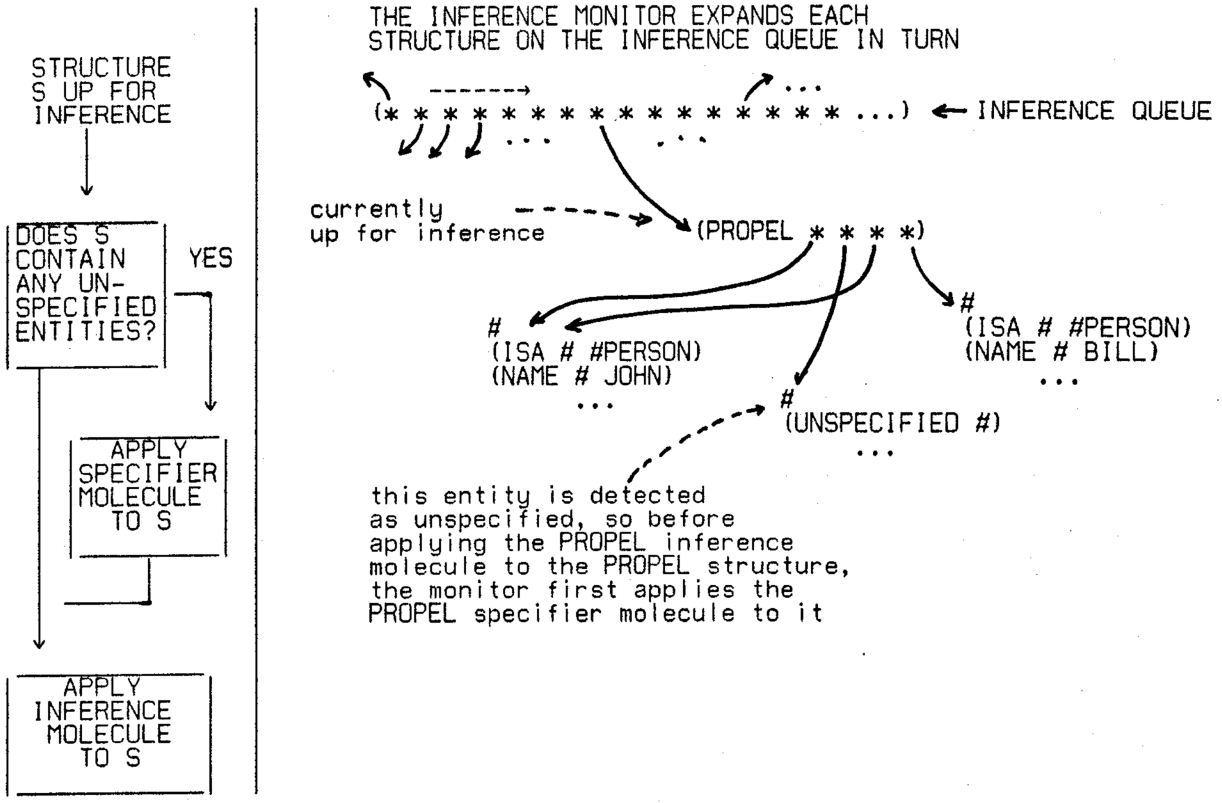



**Figure 5-3. The process of detecting missing information.**

### 5.4.4 SPECIFIER MOLECULES

Specifier molecules are executable LISP program modules which are organized by conceptual predicates. When applied to a structure, S, whose predicate is P, the P specifier molecule is capable of predicting the most likely candidate (that is, supply a pointer to some other existing structure in the memory) for a missing unit of information, in S, or, more modestly, at least helping toward this goal by predicting *more conceptual features* of the missing entity. The fidelity with which a specifier molecule does this task *in a context-sensitive manner* is simply a function of how sensitive it is to the dimensions which could affect its prediction. That is, the kinds of testing specifier molecules do in the process of specification must in general be very specific, relying on the whatever "local" heuristics are effective at accomplishing the specification, paying attention to its context. To illustrate just how local the heuristics must be from molecule to molecule, consider the following specification tasks:

| | |
|---|---|
| 1. John drooled as he viewed the banana. He ate. | FILL IN THE BANANA AS THE CONCEPTUAL OBJECT OF EATING |
| 2. Pete and Bill were alone on a desert island. Someone tapped Bill on the shoulder. | FILL IN PETE AS THE CONCEPTUAL ACTOR OF "MOVE" WHICH UNDERLIES "TAP" |
| 3. Mary picked up the rock. She hit John. | PREDICT THAT IT WAS THE ROCK WHICH WAS THE OBJECT OF MARY'S PROPELLING ACT |
| 4. John was driving his car. He hit Mary. | PREDICT THE CAR AS THE OBJECT OF THE PROPEL |
| 5. John bought a hammer. | "BUY" IS UNDERLIED BY A DUAL ATRANS ACT. WHO IS THE OTHER ACTOR? |
| 6. John was asleep. | WHAT IS THE LOCATION OF THIS COMMON STATE LIKELY TO BE IN THE ABSENCE OF OTHER EXPLICIT INFORMATION? |
| 7. Mary went to work. | WHAT IS THE TIME OF THIS COMMON ACTION LIKELY TO BE? |
| 8. John went to Paris. | PREDICT THE LIKELY INSTRUMENTALITY "FLY" |

The heuristics for these are all slightly different and peculiar to the individual situations. This is not to suggest that there are not common heuristics which are shared by many specifier molecules. Indeed, there are probably many such heuristics which remain to be discovered by examining enough specific cases. One case in point is the following heuristic whose general utility has become apparent:

Find an entity, X, which satisfies conceptual features {Y1,...,Yn},
and which has a recent RECENCY or TOUCHED tag.

For instance, in the example "He ate" the INGEST specifier molecule will try to locate something which is INGESTable (which ISA #FOOD), and which has recently been referenced explicitly or implicitly.

Since the specifier molecules are programs, they can easily reference such common heuristics via function calls to the processes which implement the common heuristics. But the knowledge required to perform any *particular* specification task is, more often than not, quite peculiar both to the conceptual predicate and to the features of the entities it relates. Thus, rather than discussing *instances* of specification inferences, I am more concerned with where they fit in the overall information flow within the memory, what they do, how they do it, and what they are good for.

#### 5.4.4.1 APPLYING SPECIFIER MOLECULES TO MEMORY STRUCTURES

The mechanism by which the inference monitor locates and applies a specifier molecule is uncomplicated. Having detected as UNSPECIFIED some entity in the bond of the structure to which an inference molecule is about to be applied, the monitor interrupts. It retrieves the bond and creates a parallel vector, V, whose contents denote which elements of the bond are unspecified: a NIL is placed in positions of V whose counterpart entity in the bond is unspecified, and entities which are not unspecified represent themselves in V. Fig. 5-4 illustrates the V which is created for our hitting example.

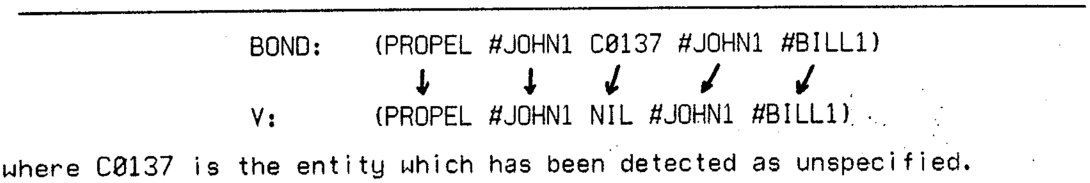



**Figure 5-4. The specification request vector.**

The monitor then locates the PROPEL specifier molecule attached as the property SPROG of PROPEL's property list. This property stores a LISP PROGram whose calling arguments are the following:

*UN* a list containing (1) the superatom, S, which represents the structure which contains one or more UNSPECIFIED entities in its bond, and (2) time information: the TIME, TS and TF of the structure if they exist, and the rough time frame these (PAST, PRESENT, FUTURE) these time aspects represent:
UN: ( S TIME TS TF FRAME )

*V* the parallel vector with NILs indicating which entities of the bond require specification

*AC OB DF DT* the actual entities in the bond, bound individually as ACtor, OBject, DFrom and DTo. Which, and how many, of these there are of course are specific to the particular conceptual predicate.

The monitor sets up these arguments, then applies the molecule to them. Since the molecule has complete information (it has access to the structure's surrounding environment and approximate time via UN, and the structure's bond is conveniently accessible through AC, OB, DT, DF), it can apply arbitrarily detailed heuristics to the specification task.

#### 5.4.4.2 INSIDE THE SPECIFIER MOLECULE

Within the molecule are *specifier atoms*. Each atom is prepared to specify one unspecified entity in the bond, in a context-sensitive way. Each atom tests a particular "slot" in V for NIL to determine whether the slot it is prepared to specify requires specification. If its slot is not NIL, the atom does nothing. Otherwise, the atom applies its heuristics in an attempt to specify its slot. These heuristics will typically be sensitive to the structure's surrounding context, to the nature of the other entities in the bond and to partial features already known about the unspecified entity (for example, it is already known that C0137 in Fig. 5-4 must be a physical object). I will describe the heuristics used to specify the three missing cases in this PROPEL example shortly. Fig. 5-7 shows a very small specifier molecule used by the program.

A specifier atom which is successful does two things:

1. it creates or locates the concept which specifies the unspecified entity
2. it replaces the NIL in V with a LISP dotted-pair which consists of (a) a pointer to this specifying entity, (b) and a list of REASONS which indicates *why* this entity was chosen.

The finished product of the specifier molecule is a new version of V, hopefully with fewer NILs. This V is returned to the monitor, which rescans it to detect any dotted pairs representing successful specifications. Fig. 5-5 shows what this V looks like when some specification atom has been successful.

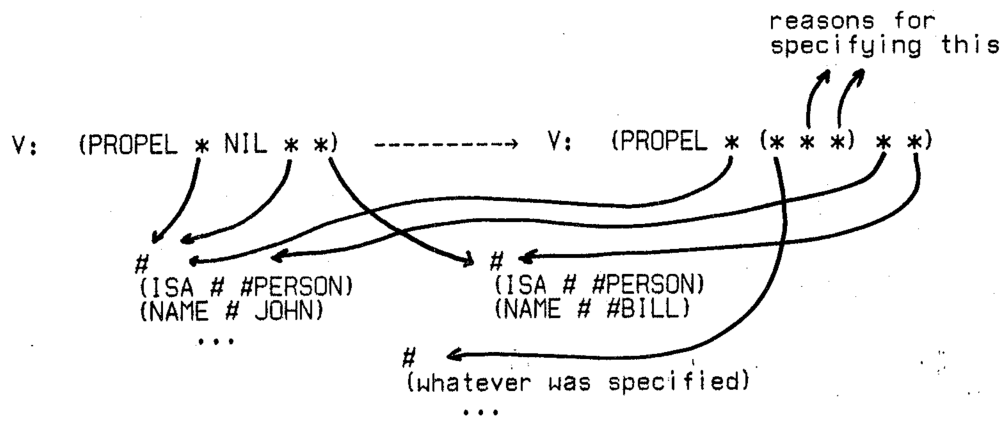



**Figure 5-5.**

### 5.4.5 MERGING THE NEW AND OLD ENTITIES

For each unspecified entity which was successfully specified, there will exist two objects in memory representing the same thing (the old, unspecified one and the new, specified one). These two entities must be *merged into one.* To accomplish this, the monitor calls the merge process, IDENTIFY_MERGE, which is described in sections 7.6 and 8.1.2. The important results of this merge are (a) that *all* references to the previously unspecified object are replaced by references to the result of the merge process, and (b) that any information collected about the unspecified entity *up to that point* will be preserved and attached as features of the newly-specified entity. To illustrate by a very simple example why this kind of conservation of existing features is important, consider the following sequence: "John picked up Pete's putty. He handed the warm round red mass to Mary." Under most circumstances, we would want these two entities which have been referenced by different descriptive sets to be identified as one and the same by the PTRANS specifier molecule. To identify is to merge the two tokens together, and do so with no loss of information. We would want the result of the IDENTIFY_MERGE process to have all the features of the two previously discrete tokens: that the entity (1) is a lump of putty, (2) is warm, (3) red, and (4) round, (5) is owned by Pete, and (6) was handed to Mary by John. The merge process is capable of doing this.

Notice that since *all* references to the old (unspecified) entity are *replaced* by references to the new one, in our hit example where the unspecified object appears in both P1 and P5 of Fig. 5-2, the successful specification of whichever of (P1 P5) is examined by the inference monitor first will obviate the need to perform specification on the other. That is, it will simply not be seen by the inference monitor again as an unspecified entity.

#### 5.4.5.1 THE PREDICATE "IDENTIFIES"

One by-product of IDENTIFY_MERGE is the creation of a memory structure (IDENTIFIES X Y), where X is the specifying object, Y is the previously-unspecified object. The REASONS returned by the specifier atom which specified Y as X constitute the REASONS list for this IDENTIFIES structure. Thus for instance, if we were to inquire of the memory "Why do you think the object John used to hit Bill was a rock?", it could respond "Because John was holding a rock at the time." This IDENTIFIES association and the attachment of REASONS to it are illustrated in Fig. 5-6.

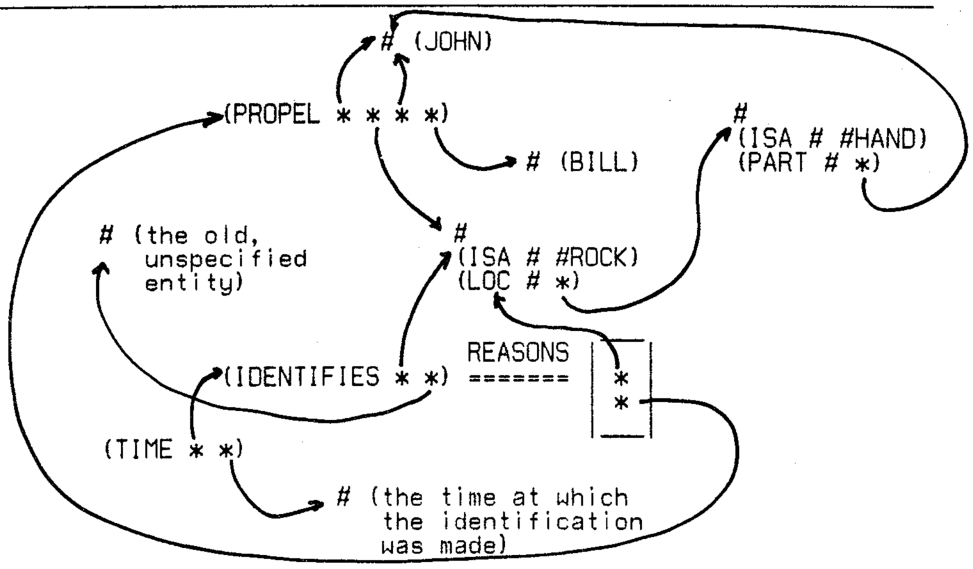



**Figure 5-6. The IDENTIFIES structure which stores the REASONS for the identification.**

After the merge, the monitor performs a small bookkeeping chore. The memory maintains a list, !MISSINGINFO, of entities which have been detected as unspecified and whose specification is pending. !MISSINGINFO is one source of things to react to after the inference processes cease: it

can be used to generate prompting questions about missing or incompletely specified entities which have been collected during the inferencing. To maintain this list, at each specification attempt, the inference monitor detects (during the scan of the specification vector, V, returned from the specifier molecule) which entities were successfully specified. Those which were are removed from !MISSINGINFO if they were on it, and those which the specifier molecule failed to specify are added to it if not already there.

After the merge, the specification process is complete. Hopefully, more features of the missing entities are now known. However, failures to specify will not preclude the application of an inference molecule to the structure; the inference will simply proceed, making the best use of whatever partial features *are* available. As a matter of fact, there is a potentially very important inference-specification interaction. The process of inferencing has the potential for uncovering new information about the unspecified entities, even based on only partial features of the objects. Because of this, the results of inferencing from structure S could be of use to the specifier molecule on a second-pass. That is, even though the specification failed on the first attempt, the process of inferencing may turn up new information which would allow the specification to succeed on a second or subsequent attempt. Since there are other reasons for subjecting all structures on the inference queue to more than one pass through the inference monitor, section 7.2 is devoted to a description of how this occurs in the program.

### 5.4.6 SPECIFIER MOLECULE EXAMPLE

We can now trace through this sequence as it specifies the missing object of the underlying PROPEL in the hit example. Fig. 5-7 shows a very simple specifier molecule with just the atom for specifying the object slot of PROPELs. There, X1, X2, X3 are temporary local variables, SP is a simple service function which replaces a NIL by the specified result, C is a low-level retrieval function which locates a concept or token from a descriptive set (or creates one if none can be found), and F1 is another retrieval function which locates a unit of information.

The specifier atom shown in Fig. 5-7 is, of course, not an ultimately realistic one since it is not sensitive to a realistic quantity of contextual information. However, for the sake of illustration, I have made it sensitive to one important dimension: if the actor of the PROPEL has something in his hand at the time of the propelling, it is reasonable to infer that this was the

object he propelled. Otherwise, the atom will infer that it was simply the actor's hand which he propelled, since this is the "default" object for a person's PROPELling.

```
(SPROG *PROPEL* (UN V AC OB DF DT) (X1 X2 X3) (
(COND ( (NULL (CADR V))
        (COND ( (AND (SETQ X1 (C (@ISA @_ @#HAND)
                                 (@PART @_ AC)))
                     (SETQ X2 (F1 (@*LOC* @_ X1)))
                     (SETQ X3 !GLOBALFIND))
               (SP V 2 X2   (LIST X3))
              )
              ( T
               (SP V 2 X1 NIL)
              )
        )
      )
      .
      . (other specifier atoms go here)
      .
)
(RETURN V)
))
```

This is a simplified specifier molecule containing just an object specifier atom. (NULL (CADR V)) is test for lack of object specificati

If unspecified, the atom locates the hand of the actor, assigning it to X1. It then checks to see if anything is located in X1. If something is found, it is bound to X2, and the LOC structure which expresses this information is bound to X3. If nothing is located in the actor's hand, his hand itself (X1) is inferred. The (LIST X3) in the first SP call is the list of REASONS (just one here) justifying the specification of the object the actor was holding as the object of the PROPEL.

**Figure 5-7. A very simple specifier molecule.**

### 5.4.7 SPECIFICATION-REFERENCE INTERACTION

In its most general form, the specification of an entity can involve the full powers of the reference-finding mechanism. For instance, consider the following dialog excerpt:

Bill: John bought some milk a few minutes ago.

Pete: That's funny, I was at the grocery a few minutes ago and I didn't see him.

Here, "bought" is underlied by a dual ATRANS (exchange of money for goods) in which one of the parties is unspecified. Pete, the hearer, is able to make the predictive specification that it was "a grocery store", using the knowledge that the object of one of the ATRANS's was a food. However, he clearly went on to determine *which* grocery store John probably went to; that is, he tentatively determined the *referent* of the specifying entity, not just its class concept, #GROCERYSTORE. This also occurred in our PROPEL example, but it occurred implicitly there, since there is little referential ambiguity in locating the token which is someone's hand.

In general, then, the specification process must not only make explicit the *concept* involved (grocery store, hand, etc.), it must also predict *which token* of the concept is likely to be the one involved. I have included a computer example at the end of this section which illustrates the beginnings of this capability: there, an ATRANS atom will predict that a grocery store is involved as the unspecified entity, X, in a dual ATRANS action. However, *which* grocery store will not be ascertainable. The specifier atom therefore furthers the specification of X in the structure by specifying X as *some* grocery, Y: (ISA Y #GROCERYSTORE); but the specifier atom leaves Y marked as unspecified when it fails to determine *which* grocery store.

It should be clear that the process of specification is not an all-or-none endeavor. For instance, although the exact referent implied by a missing case may not be inferrable in the current context, it may nevertheless be possible to infer enough *features* of it to allow full comprehension of the utterance. The problem of knowing when enough features about a new token have been collected to call it "specified" can be an elusive one, and is of course ultimately dependent upon what needs to be known about the entity for some particular purpose. I have taken the shortcut approach in implementing the memory that information which requires specification should remain unspecified until the specification process results in the identification of an *existing* token or concept in the memory.

### 5.4.8 OTHER EXAMPLES: TYPICAL SOURCES OF MISSING SPECIFICATION

To simplify the discussion, I have ignored the two other missing entities in the example utterance "John hit Bill": its INSTrumentality and its LOCation. Of course, these subpropositions (P2 and P3 in Fig. 5-2) also will come up for inference, be detected to contain unspecified entities, and similarly undergo specification by the INST and LOC specifier molecules, respectively. The heuristics used in the LOC molecule are things like: "The location of an action can be determined from the locations of the objects involved in the action", or "Some specific actions and states have very specfic normal locations." An example of the former is "John was watching the elephants.", where, knowing at least that John was in the Bay Area, we might infer that this action occurred at whatever the location of Bay Area elephants happens to be, very likely the San Francisco Zoo. Examples of the latter are: "Mary played tennis.", where the location of the actions of tennis playing is nominally a tennis court, and "Bill was asleep.", where

the normal location of such a state is at home in bed. Clearly, heuristics of the latter sort should be applied only after the more *specific* tests of the former sort fail, since, for example, we might just have been told that Bill was on the subway. The "default" specification of missing information, therefore, must rely heavily upon assumptions about what is normal in the world, and in the memory I have chosen to embody these assumptions in specifier molecules.

We will see in section 6.8 how the process of specification relates to a very important class of inference concerned with the maintenance of time relations.

### 5.4.9 A SUMMARY OF SPECIFICATION INFERENCES' UTILITY AND OPERATION

The importance of filling in missing and unspecified information as one goal of understanding utterances should be evident. We can summarize the potential contributions of a specification inference to the process of understanding an utterance by these five points:

1. **It can touch (draw out) implicitly referenced concepts and tokens**

   and these can clarify future references which might otherwise be ambiguous, or unsolvable

   (a) John picked up the apple and the knife. He ate. *It* tasted terrible.

   (b) Bill wanted to buy a catcher's mit.
   *The store* was closed.

2. **It can generate questions for more information**

   (a) John bought a new hat.
   Which store did he go to?

   (b) Bill was reading a book.
   What was the book about?

3. **It can begin new and important lines of conceptual inference**

   (a) John and Pete were alone on a desert island.
   John said that if anyone ever dropped a coconut on his head, he'd kill him.
   Next day, someone dropped a coconut on John's head.

   (b) John was reading the inscription on the Lunar plaque left by Apollo 11.
   (Instrumentality is specified as ATTEND through EYE. This leads to the inference that John is near what he is reading, namely that he is on the moon!)

4. **It can lead to the discovery of apparent contradictions**

   (a) John was bound and gagged.
   He hit Mary. (Here, the instrumentality supplied as John's MOVEing his hand, and this will lead to an apparent contradiction with the conceptual content of the first line)

5. **It implements one aspect of the flywheel effect,**

the logical momentum
through which the information communicated by several utterances can be knit together. That is, each specification inference potentially leads to new points of contact in inference space.

(a) John picked up a rock.
He hit the door.

(b) Mary was standing on the corner.
Pete came over to say hi. (that is, he said hi to *Mary*)

(c) Mary dropped the sledgehammer on Rita's foot.
She apologized.

## SPECIFICATION INFERENCE COMPUTER EXAMPLE 1

In this example, we will see how the context in which a specification inference occurs can affect the substance of the specification. Normally, the *hand of the actor* is supplied as the missing object case in the conceptual template which underlies "hit". However, when the hitter has some other object in his hand just before the time of hitting, the PROPEL specifier molecule predicts that that object is more likely than his hand. At the end, the results of the IDENTIFY_MERGE process are shown.

```
JOHN PICKED UP A ROCK

((*GRASP* (#JOHN1) (C0017))
(TIME _ (C0019)))

C0022

STARTING INFERENCE QUEUE:
((X 1.0 C0022))
```

To illustrate how context can affect the inferring of unspecified or missing information, we use the following example: "John picked up a rock. He hit Mary." Here, MEMORY will infer that it was the rock, rather than just John's fist, which came into contact with Mary. In the absence of the first line of this example, MEMORY infers that John simply used his hand. The second example will be an example where default world knowledge is used to specify missing information. In this example all other subpropositions have been suppressed.

```
      ......

ABOUT TO APPLY @GRASP1 TO C0022
   C0022: (*GRASP* #JOHN1 C0017)
INFERRING: (*LOC* C0017 C0024)
   ALSO GENERATING: (TIME C0027 C0019)

      ......
```

MEMORY spontaneously generates inferences.

One inference from the first line is that a rock begins being in John's hand.

```
JOHN HIT MARY

((CAUSE ((*PROPEL* (#JOHN1) (C0035)
(#JOHN1) (#MARY1)) (TIME _ (C0038)))
((*PHYSCONT* (C0035) (#MARY1))
(TIME _ (C0038)))))
```

Now MEMORY encounters the second thought.

That is, John propelled some physical object (C0035) from himself to Mary, and this caused C0035 to be in physical contact with Mary.

```
C0043
```

C0043 is the structure representing this second input.

---

```
(*BREAK* . HELLO)
```

---

```
C0035: NIL

ASET:
   C0041: (*PHYSCONT* # #MARY1)
   C0039: (*PROPEL* #JOHN1 # #JOHN1
                #MARY1)
   C0037: (UNSPECIFIED #)
   C0036: (ISA # C0033)
RECENCY: 8783
```

MEMORY is about to begin inferencing. We interrupt the program briefly to examine the token which represents the (unspecified) object which John propelled toward Mary. The lack of specification was denoted by the analyzer by the modification SPEC (*U*).

C0033 is the abstract concept for a physical object. This is the only feature the analyzer could infer about C0035 using its limited linguistic knowledge of "hit".

---

```
STARTING INFERENCE QUEUE:
((X 1.0 C0043))

        ......

UNSPECIFIED OBJECT(S) DETECTED
IN C0039: (*PROPEL* #JOHN1 C0035
                #JOHN1 #MARY1)
SPECIFYING...
PURGING: (UNSPECIFIED C0035)
PURGING: (ISA C0035 C0033)
MERGING:
        C0017: C0017
        C0035: C0035
```

Control is given back to the program. Inferences are begun for this input.

Eventually, a proposition containing this unspecified object becomes the focus of the inferencer. At that point, the unspecified of specification is detected by the inference monitor. It calls the *PROPEL* specifier molecule, indicating that C0035 is to be specified if possible. The specifier molecule infers that the object was probably C0017, the rock, because it was in John's hand at the time. Having specified C0035, MEMORY merges C0035 into C0017 (the rock), thus coalescing all knowledge about the object into C0017.

---

```
(*BREAK* . HELLO)
```

We again interrupt MEMORY to examine the C0017 and C0035 just before the merge.

---

```
C0017: NIL

ASET:
   C0027: (*LOC* # C0024)
   C0022: (*GRASP* #JOHN1 #)
   C0018: (ISA # #ROCK)
RECENCY: 6383
```

C0017 is the rock which John was holding. C0024 is John's hand. C0027 was an inference which arose from the first line.

---

```
C0035: NIL

ASET:
   C0041: (*PHYSCONT* # #MARY1)
   C0039: (*PROPEL* #JOHN1 # #JOHN1
                #MARY1)
RECENCY: 9650
```

C0035 is the unspecified object John brought into physical contact with Mary. Notice that the (UNSPECIFIED #) has been removed before merging. Notice also that C0035's ISA relation with #PHYSOBJ has been purged, since C0017 is already known to be a rock, which ISA #PHYSOBJ.

---

```
*PROCEED
SPECIFIED RESULT:
   (*PROPEL* #JOHN1 C0017 #JOHN1
                #MARY1)
```

Control is returned to the program. The specified object now appears in all structures which referenced its unspecified token, since the merge process replaces internal pointers. Having been specified, this proposition

```
      ......
```

will lead to other inferences via the normal inference molecule for *PROPEL*.

---

```
C0017: NIL

ASET:
   C0048: (*FORCECONT* # #MARY1)
   C0047: (IDENTIFIES # C0035)
   C0039: (*PROPEL* #JOHN1 # #JOHN1
               #MARY1)
   C0041: (*PHYSCONT* # #MARY1)
   C0027: (*LOC* # C0024)
   C0022: (*GRASP* #JOHN1 #)
   C0018: (ISA # #ROCK)
RECENCY: 9650
```

At the end of inferencing, we reexamine C0017, the rock. Notice that the merger has left a record of MEMORY's decision to specify the unspecified physobj as this rock. This information is preserved in C0047 which records this "identity relation" between C0017 and C0035.

---

```
C0035: NIL

ASET:
   C0047: (IDENTIFIES C0017 #)
SAVEDASET:
   (*PHYSCONT* # #MARY1)
   (*PROPEL* #JOHN1 # #JOHN1 #MARY1)
RECENCY: NIL
```

This is C0035 after the merge. Notice its only occurrence set (ASET) member is this identity relation with C0017. All other members of its ASET were de-activated (unlinked from the rest of MEMORY), and saved under the property SAVEDASET.

---

```
C0047: (IDENTIFIES C0017 C0035)

RECENCY: 9650
TRUTH: T, STRENGTH: 0.95
REASONS:
   C0039: (*PROPEL* #JOHN1 C0017 #JOHN1
               #MARY1)
   C0027: (*LOC* C0017 C0024)
ISEEN: NIL
```

This is the identity relation which the specification process created. Notice the preservation of MEMORY's reasoning: C0039 and C0027. In English: "The object C0035 must be the rock C0017 because John propelled it, and he was holding C0017 at the time." They are not visable, but there are of course time predications on both C0027 and C0039.

# SPECIFIER MOLECULE COMPUTER EXAMPLE 2

In this example, the missing second actor in the sentence "John bought some milk" is specified, using default knowledge in this case, as a grocery store (personified). Notice how the IDENTIFY_MERGE changes all references to the newly specified entity, and how "Milk is a food" is supplied as a reason for deciding upon "grocery store."

---

```
JOHN BOUGHT SOME MILK

((DUALCAUSE ((*ATRANS* (#JOHN1)
(C0028) (#JOHN1) (C0030)) (TIME
  (C0033))) ((*ATRANS* (C0030)
(C0036) (C0030) (#JOHN1))
(TIME _ (C0033)))))

C0042

STARTING INFERENCE QUEUE:
((X 1.0 C0042))
```

This example demonstrates how MEMORY's default knowledge of normality is used to specify missing information. Here, John's buying milk is represented as a double causal: John gives someone some money and that person in turn gives John some milk. MEMORY's job is to predict who the missing person is.

Again, other subpropositions have been suppressed for this example.

---

```
        ......

UNSPECIFIED OBJECT(S) DETECTED IN
C0040: (*ATRANS* C0030 C0036 C0030
                #JOHN1)
SPECIFYING...
PURGING: (UNSPECIFIED C0030)
PURGING: (ISA C0030 #PERSON)
MERGING:
        C0054: C0054
        C0030: C0030

(*BREAK* . HELLO)
```

Inferences are generated. Eventually, that someone (C0030) ATRANSed John some milk becomes the focus of the inferencer. At that point, C0030's lack of specificity is detected, and the *ATRANS* specifier molecule is called to fill in the actor (and donor) in C0040. The specifier molecule sees that the object off the ATRANS is some food, so, in the absence of context, predicts that C0030 is some grocery store. MEMORY is quite content to personify such things as stores, although this is admittedly sloppy. C0054 is the (newly-created) token representing the grocery store. It is about to be merged with C0030.

We interrupt MEMORY to see C0030 and C0054 just before the merge.

---

```
C0030: NIL

ASET:
   C0040: (*ATRANS* # C0036 # #JOHN1)
   C0038: (*ATRANS* #JOHN1 C0028
                #JOHN1 #)
RECENCY: 7216
```

---

```
C0054: NIL

ASET:
   C0056: (UNSPECIFIED #)
   C0055: (ISA # #GROCERYSTORE)
RECENCY: NIL
```

Notice that, even though MEMORY has specified C0030 as some grocery store, WHICH grocery store it is is still unknown.

---

```
*PROCEED
(*BREAK* . HELLO)
```

MEMORY proceeds with the merge. We again interrupt it to see the merged result, C0054.

Here is the merge result.

---

```
C0054: NIL

ASET:
   C0057: (IDENTIFIES # C0030)
   C0038: (*ATRANS* #JOHN1 C0028
               #JOHN1 #)
   C0040: (*ATRANS* # C0036 # #JOHN1)
   C0056: (UNSPECIFIED #)
   C0055: (ISA # #GROCERYSTORE)
RECENCY: 7216
```

---

```
C0030: NIL

ASET:
   C0057: (IDENTIFIES C0054 #)
SAVEDASET:
   (*ATRANS* # C0036 # #JOHN1)
   (*ATRANS* #JOHN1 C0028 #JOHN1 #)
RECENCY: NIL
```

Here is the previously unspecified ATRANSer of milk to John. It has been unlinked from the rest of MEMORY, identified, and had its occurrence set saved.

---

```
C0057: (IDENTIFIES C0054 C0030)

RECENCY: 25650
TRUTH: T, STRENGTH: 1.0
REASONS:
   C0040: (*ATRANS* C0054 C0036 C0054
               #JOHN1)
   C0037: (ISA C0036 #MILK)
   I0189: (ISA #MILK #FOOD)
ISEEN: NIL
```

Here is the identity relation between the grocery store, C0054, and C0030. Notice the reasons MEMORY has recorded to justify this identity: that the ATRANS event occurred, and that its object was #MILK, which is #FOOD.

---

```
*PROCEED

SPECIFIED RESULT:
(*ATRANS* C0054 C0036 C0054 #JOHN1)
```

MEMORY proceeds with inferencing, using this newly-specified object.

```
        ......

UNSPECIFIED OBJECT(S) DETECTED IN
C0038: (*ATRANS* #JOHN1 C0028 #JOHN1
               C0054)
SPECIFYING...
NO RESULTS
```

Somewhat later, the other ATRANS action reaches the inference monitor. This time it is detected that WHICH grocery store it was is still unknown. However, since there is no new information, no further specification results.

---

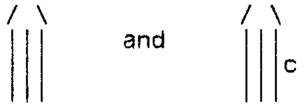
```
APPLYING INF MOLECULE *ATRANS* TO
C0038: (*ATRANS* #JOHN1 C0028 #JOHN1
               C0054)

        ......
```

Inferencing proceeds

## 5.5 CAUSALITY

Causality is perhaps the single most important notion to a conceptual memory, because it not only pervades language, but it is one very clear domain in which it is necessary to apply a detailed model of the world in order to comprehend. In chapter 2, we saw how causality and conditional causality are represented in the Conceptual Dependency framework. But, as we will see, there are many central tasks in the memory which are based upon conceptual causals, and these tasks are not immediately apparent from the issues of *representing* causality. Rather, they have to do with *explaining* causal relationships in terms of other world knowledge.

Before describing the two inference classes most closely related to the notions of causality and conditional causality, it will be useful to examine the possible kinds of information that can be related in a meaningful way by the causal relations




Two descriptive schemes are relevant to this purpose:

1. a "syntax" of structurally allowable causal forms *at the level of conceptual representation of an utterance*, and
2. a "syntax" of what can meaningfully be connected by causal relations, *relative to a model of causality* in the world

The set defined by (2) will be a subset of that defined by (1).

Why bother with the form of causals at all? The answer is an important one, because it concerns a crucial task of conceptual processing: the filling-in of an implied sequence of causal relationships where only one has been stated. Human language users do this when decoding the meaning of each utterance they perceive. Likewise, from the standpoint of generating language, people rarely make explicit the blow-by-blow details of the causality aspects of what they communicate, since they can safely assume the hearer will be able to fill in the missing pieces. When he cannot justify the communicated causal in terms of smaller cause-effect units in his model, the hearer stops and asks "how is that?" On the other hand, when he *can* explain the intervening causal steps, making them explicit will draw out and touch many other underlying concepts.

**It is important to explain language causality in terms of model causality. To do this, it iss necessary to distinguish between causality as people use it in language, and causality which actually occurs in the world.**

### 5.5.1 CAUSALITY COMMUNICATED BY LANGUAGE AND CAUSAL CHAIN EXPANSION

The memory receives a wide variety of causal relationships from the analyzer which the analyzer detects explicitly or infers from the conceptual content of linguistic structures in sentences it hears. However, the analyzer is hearing *people's* versions of causality, and is thus compelled to produce a conceptual analysis using these versions. Unfortunately, the only guarantee on these causal relationships is that they obey the syntactic relationships permissible for causals *in the representational formalism.* That is these permissible forms occur conceptually in what people say, and thus must be analyzed by the conceptual analyzer. These are enumerated, with examples of each, in Fig. 5-8.

(a) STATE <≡ STATE (b) STATE <≡ ACTION (c) STATE <≡ CAUSAL

(d) ACTION <≡ ACTION (e) ACTION <≡ STATE (f) ACTION <≡ CAUSAL

(g) CAUSAL <≡ CAUSAL (h) CAUSAL <≡ STATE (i) CAUSAL <≡ ACTION

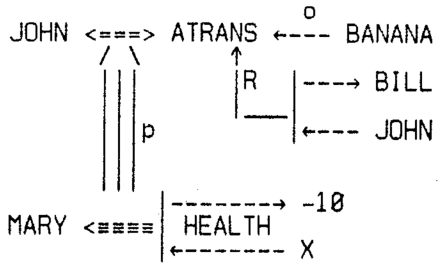
(a) "John wants to go because he is depressed."
(b) "John went because he was happy."
(c) "John kicked Bill because he was mad."
(d) "Mary cried because Bill ate the cookie."
(e) "Mary was hurt because John hit her."
(f) "John threw the ball because Bill told him to."
(g) "Mary kissed John because he hit Bill."
(h) "John was aggravated because Bill and Mary swapped toys."
(i) "Mary cried because John killed the plant."

**Figure 5-8. Representable causal forms.**

There is no guarantee, however, that the conceptual information conveyed by the stated causal makes any immediate sense, relative to the model's ability to explain causality in the world which it models. To emphasize the potential disparity between what can easily be *represented* by the conceptual analyzer, and what can easily be *explained* in terms of smaller cause-effect units in the model, consider the sentence

John killed Mary by giving Bill a banana.

which is analyzed as follows:




Outside of very peculiar contexts, a human language user would certainly be hard-pressed to make sense of this. Although the conceptualization is syntactically correct according to the representational formalism, it makes no direct sense because the causal relation is being used to stand for an *entire sequence* of unstated causal relations. To fill in this sequence of missing causals using world causality knowledge is a very important aspect of understanding. I will call it *causal chain expansion.*

A less nonsensical example of causal chain expansion is illustrated by the utterance "Mary's tears flowed because she knew her lover John had drunk some poison", whose underlying meaning is represented by the graph shown in Fig. 5-9.

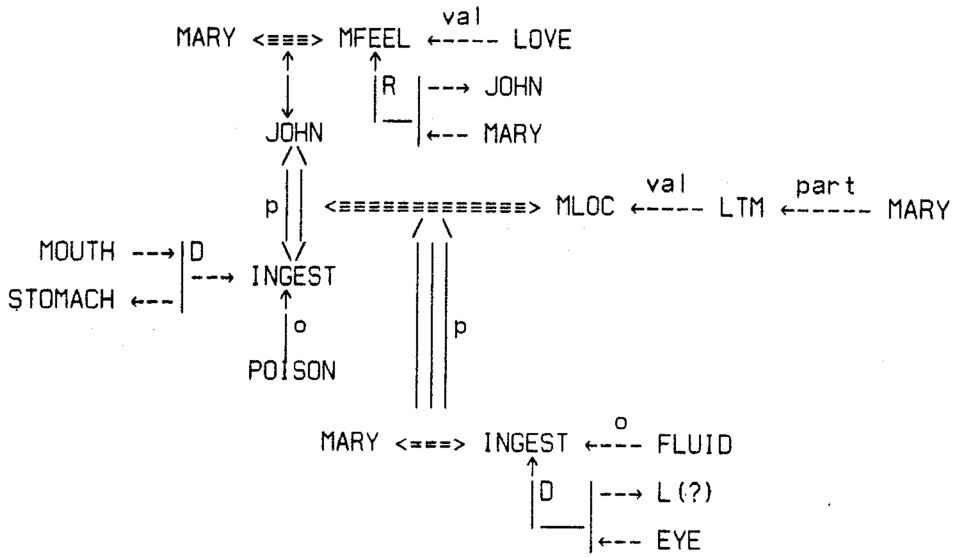



**Figure 5-9.**

The goal is to ingrain in the memory (a) the awareness to recognize that every causal must be reconciled with the memory's knowledge of causality, and (b) the ability to explain each causal, and recognize when it has and has not been explained. In this example, we would like an expansion similar to that shown in Fig. 5-10 to be achieved. In the banana example, we would like the memory to respond "How did that happen?"

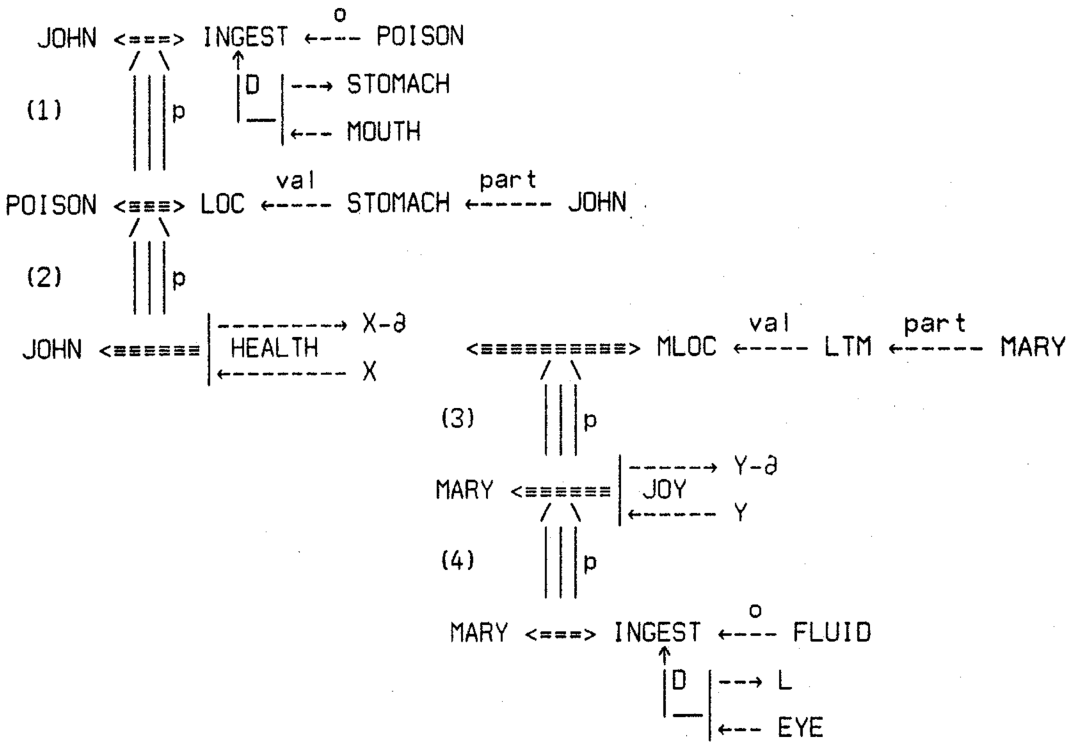



**Figure 5-10.**

### 5.5.1.1 "SHOULD", "OUGHT TO", ETC. AND CAUSAL CHAIN EXPANSION

One very common language source of underlying causality which requires expansion into underlying causal chains involves the notions "should", "ought to", "better", "have to", and related concepts. One of two conceptual forms nearly always underlies these notions, and the central link in both is a causal. Consider the sentence "I think I should give Bill the bike" (Fig. 5-11).

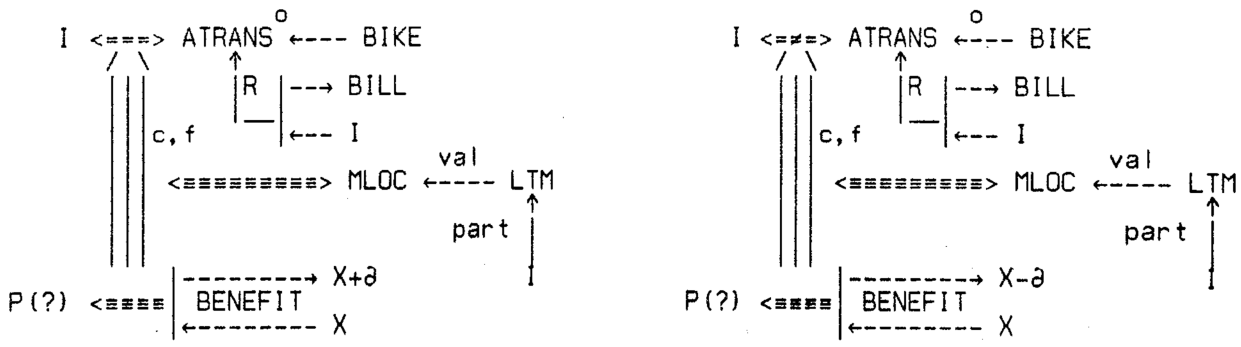


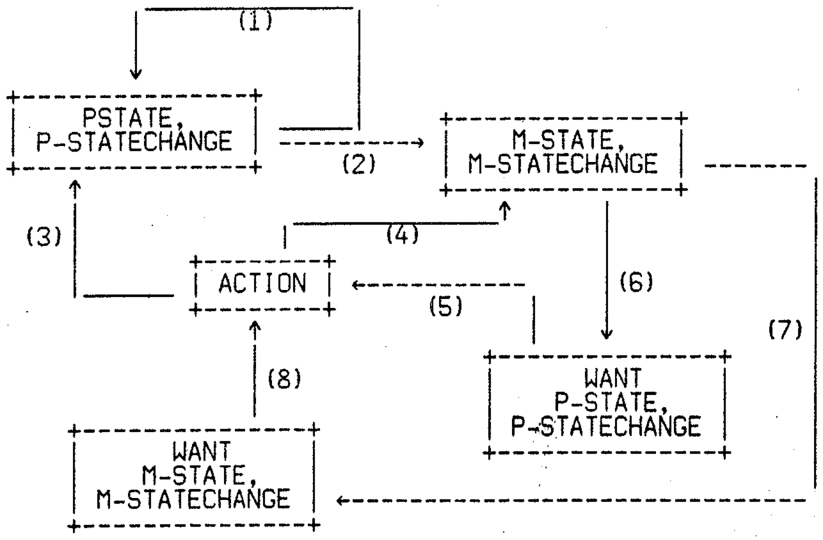

**Figure 5-11.**

The central issue of understanding these forms is to fill in one of the two paths which explains (a) why an action can lead to someone's benefit, or (b) why an inaction can lead to someone's detriment. Notice that a successful expansion of the causal involved in one of these two underlying meanings of "should" can help select *which* underlying meaning is most appropriate, based on a knowledge of causality in the world. It can also provide information which will allow a specifier molecule to fill in the missing recipient of the benefit, P (or the P who averts some sort of loss).

These observations about causal chain expansion lead to the principle:

> **Every incoming causal must be suspected of conveying an entire unstated causal chain.**

Section 5.5.4 describes how causal chain expansion occurs in the program. I will conclude this section with a "causal transition diagram" outlining the memory's "naive psychology" of cause and effect explanations in the world. Fig. 5-12 shows the types of causal transitions I would like the memory to adhere to in expanding causal chains relative to its model of the world.




EXAMPLES:

(1) "The sunshine melted the ice."
(2) "Mary is sad because John is dead."
(3) "John's hitting Mary hurt her."
(4) "Pete knows John is here because Bill told him so."
(5) "John hit Bill because he wanted him to be hurt."
(6) "Mary wanted Bill to die because she was angry at him."
(7) "Knowing that Bill hit John angered Mary."
(8) "Mary went to the party because she was depressed."

**Figure 5-12. Causality in the memory.**

### 5.5.2 RESULTATIVE AND CAUSATIVE INFERENCES

**sample:** John hit Mary with a rock.
Mary was hurt.
John was probably mad at Mary.
Mary may have become mad at John.

**sample:** Mary gave John a car.
Mary doesn't have the car anymore.
John has the car.

**sample:** John told Mary he saw Bill yesterday.
Mary knows that John saw Bill yesterday.

**sample:** Mary was supposed to help John Tuesday.
She didn't do it.
She felt guilty.

If it can be said of any one class of conceptual inference, the workhorses of understanding

by inferencing are those inferences which predict and explain cause and effect relations (a) within a single utterance and (b) among many utterances, or sentences in a story. Let us call those which predict the *cause* of some structure in the memory *causative inferences*, those which predict the *effects* (results) of some memory structure, *resultative inferences*. Since they are, roughly speaking, "inverses" of each other, they will be described together in this section.

The conclusion of the previous section is that people spend a major portion of "thought time" trying to explain, justify or predict the causes and effects of everything they perceive, from both linguistic and sensory stimuli. Watching the magician, we become quite disturbed when we cannot explain cause and effect. To know what causes states of the world to come about, what causes people to act, and what influence specific actions exert on the world lies at the heart of our ability to comprehend and use language. Because of this, resultative and causative inferences constitute two very strong "dimensions" in the spontaneous inference space.

#### 5.5.2.1 RESULTATIVE INFERENCES

The problem of explaining cause and effect is the following: given a state or action which has occurred in the world, what CAUSEd it, and what did it in turn CAUSE in the particular context in which it existed or occurred? In general, there will be many factors which, considered together, explain the cause of something, or predict the effects it will have. Some cause-effect relations are quite simple, involving only one factor, whereas others are quite complex and involve large numbers of factors. For example, an extremely simple resultative inference which invariably arises with very high likelihood from a TRANS action is that the TRANSed entity begins existing at the location to which the TRANS occurred. Thus, if Mary gives Bill the book, Bill begins having the book and Mary ceases having the book. These two resultative inferences rely on just one antecedant: the book was ATRANSed (Fig. 5-13).

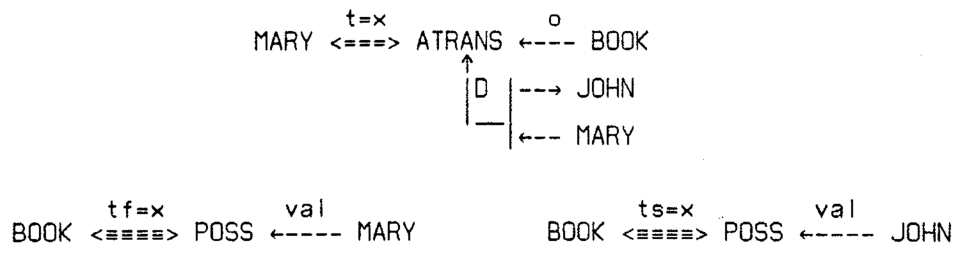



**Figure 5-13. Two very simple resultative inferences.**

However, to illustrate a more complex resultative inference, consider the reaction to "Baby Billy threw his top at the cat" in the context of Baby Billy's mother having seen him do it, and having previously told him not to do it. Among other simpler ones (the cat gets hurt, for example), we would like the memory to recognize the likelihood of Billy's mother becoming angry because of his action, and hence becoming angry at him.

#### 5.5.2.2 CAUSATIVE/RESULTATIVE INFERENCES, "CAUSE" AND "REASONS"

This example typifies most complex resultative inferences: although the inference is *triggered* by just one other unit of information (Billy's kicking), the triggering at that point is only possible because all the other requisite conditions for the resultative inference already existed at the time the triggering information was perceived. These more complex resultative inference are frequently called "belief patterns". The relation between the triggering information and the other contributing factors is shown in Fig. 5-14.

By convention, when a resultative inference for a complex pattern such as this is triggered (detected and generated by an inference molecule), the information structure which triggered it is said to have CAUSEd the structure which is the product of the resultative inference. In the example above, this means that Billy's mother's anger was directly CAUSEd by his kicking. But in addition, to preserve the surrounding circumstances (antecedants) whose existence permitted the triggering, those circumstances are recorded as the REASONS for R's existence. Thus, if we ask the memory "What is likely to have happened?", it has enough information to make the response: "Billy's mother probably became angry at Billy because he kicked the cat, she knew he did it, and she had told him not to do it."

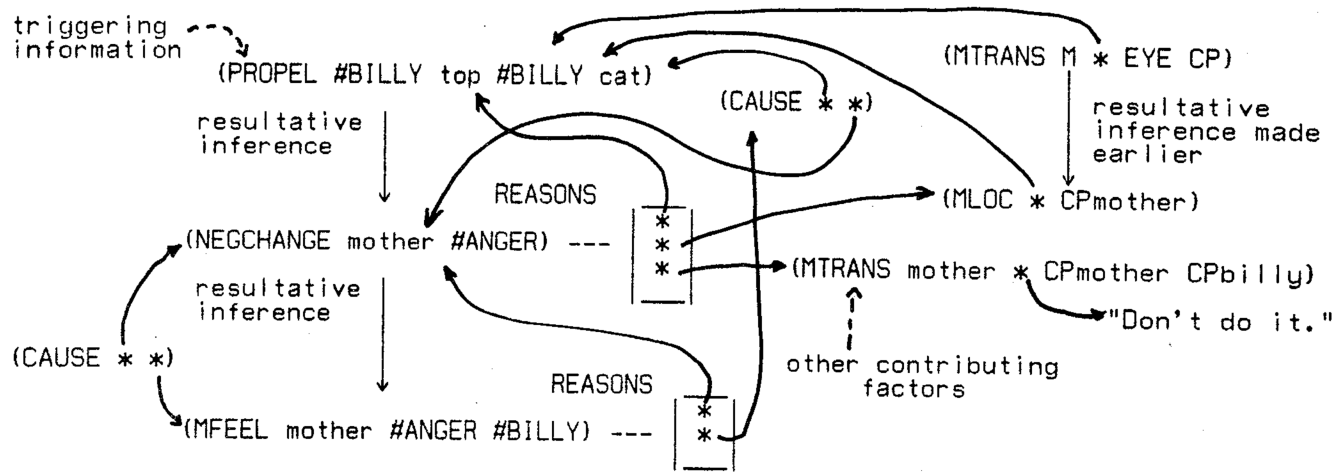



**Figure 5-14. The relations among causative/resultative inferences, REASONS and CAUSE.**

The resultative inference class is very broad, very useful, and vital to other kinds of conceptual inferences. Since we will see many examples of resultative inferences throughout other sections, I will not undertake more examples here.

#### 5.5.2.3 CAUSATIVE INFERENCES

Causative inferences are in general less easy to predict than resultative inferences. This is in part because language tends to emphasize how the world moves forward, and this is the domain of resultative inferences. It is also partly because many actions and states in the world are caused by people, and discovering their intentions is not always easy. Because of this, section 5.6 which deals with motivation and intentionality will account for a large class of causative inferences.

However, there are many causative inferences which can be made rather easily, and which can contribute to understanding in important ways. To make them is to draw out new information and touch new concepts; hence they should be spontaneously generated in as much proliferation as is possible.

Consider the examples:

1. Mary has the diamond ring.
2. The plastic doll melted rapidly.
3. John went to the store.
4. Mary was mad at John.

These are cases where it is possible and quite useful to make causative inferences. From (1) it highly likely that the cause of Mary's having the ring is that someone ATRANSed it to her. If the memory makes this inference, it will draw into the picture the question of *who* ATRANSed it to her; to discover this might be important to the larger understanding of this utterance, and it would be missed entirely if the causative ATRANS inference is not made. Similarly for (2): what is likely to be causing the doll to melt? One likely explanation is that it is near something very hot. But this possibly means that someone PTRANSed it there (another causative inference). To draw all this probabilistic information out increases the chances of relating (2) to other information, say in a story. And if it does not, this is an important cue that to understand (2) might require some special processing by some higher level heuristics. That is, it can help to discover what might be a potentially interesting task to which to devote some goal- directed processing.

(3) above is an example of causality which can be explained in terms of an actor's probable intentions. In (4) a very likely and useful causative inference is that John did something which caused some sort of NEGCHANGE (directly or indirectly) to Mary. To infer this as a causative inference is to draw out this fact in which an UNSPECIFIED action is predicted to have occurred. This will eventually be detected by the DO specifier molecule which will attempt to specify this missing action. If, for example, utterance (4) occurs in the environment

John had painted the kitchen cabinets black.
Mary was mad at him.

the specifier molecule could tentatively infer that it was this action which had angered Mary. On the other hand, if the specification is not possible by the heuristics in the DO inference molecule, the memory at least has the basis at that point for asking the question "What did he do?". Without the causative inference, this would not be drawn out.

#### 5.5.2.4 MAKE THEM ALL!

In general, it will be possible to make more than one causative and more than one resultative inference from a structure. When this is the case, *they all should be made.* Recall that it is the goal of inferencing to establish as many points of contact in inference space as possible. To do this, there must be considerable breadth. Otherwise, things which are seemingly unimportant in most contexts might be squelched in some contexts in which they were of extreme importance. Since it is the goal of conceptual inferencing to make these discoveries, the memory cannot safely suppress things at this low cognitive level.

### 5.5.3 LANGUAGE-COMMUNICATED CANCAUSE RELATIONS

There is always at best only a fine distinction between what is process and what is data. In the memory, I have chosen to encode as much inferential knowledge about the world as possible in the form of executable LISP procedures which I have called inference molecules. These processes which generate inferences can be made arbitrarily sensitive to context simply by having them perform enough tests for the presence or absense of other information in the memory which could affect the nature of the inferences they generate.

But how is the memory to encode highly specific patterns of inference which come and go with the passage of time? Specifically, how can very specific, often transitory, CANCAUSE information which has been communicated by language exert an influence on the generally program-based control structure I am proposing? For instance, if Mary tells John that to possess a catcher's mitt would make her happy (Fig. 5-15), how can this knowledge augment the *less transitory* inferences the memory can already make about acts of POSSessing in general (that is, those which are already encoded as *process* in inference molecules)? Clearly, if there existed effective algorithms for mapping data patterns into programs which could test for those patterns, we could manifest the *entire* inference capabilty of the memory in inference molecules. New (language-communicated) inferences could be mapped from their data form into chunks of code in the appropriate inference molecules (POSS, POSCHANGE in this example). There, they would exert their influence in the same way as all other "original" inferences.

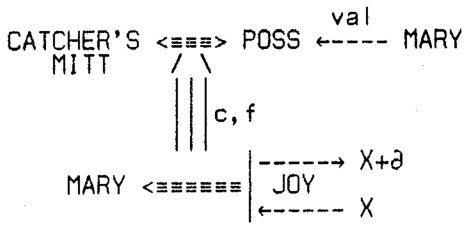



**Figure 5-15. Having a catcher's mitt would make Mary happy.**

But there are two problems with this. First, we don't yet know enough about procedures which transform *descriptions* of processes (a fairly simple inference in this case) into *procedures* which implement those processes (an inference atom in this case). Second, many of the specific inference patterns communicated by language are extremely fleeting, and it is not clear that they *should* be framed in the same relatively static procedural knowledge of the world as more universally applicable inferences. The example of Fig. 5-15 is a case in point: as soon as Mary gets a catcher's mitt, this inference is no longer of much utility, and even if she doesn't get one, the validity of the pattern may fade rapidly with time. For these reasons, it is desirable that the memory have the ability to use data-based CANCAUSE patterns to augment the basic inference capability in causative and resultative inference molecules.

In order to make the process of generating a causative or resultative inference sensitive to CANCAUSE data patterns, the inference monitor must, in addition to applying the appropriate inference molecule to each structure S from which it is to generate inferences, also *perform a search* for information of the forms

```
(CANCAUSE S X)  (to discover resultative inferences)
(CANCAUSE Y S)  (to discover causative inferences)
```

If the first form can be found, then the resultative inference, X, can be generated; if the second form can be found, the causative inference, Y, can be generated. Of course, there may be several applicable CANCAUSE structures; if so all should be applied.

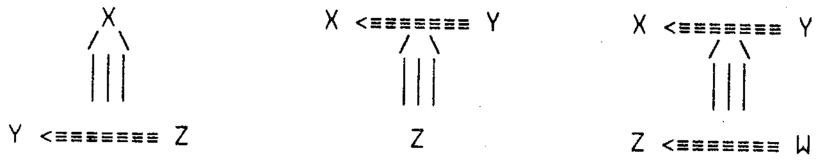
There are currently no heuristics for selectively "deactivating" a language-communicated

CANCAUSE structure after having used it one or more times. That is, if Mary does receive a catcher's mitt, are we justified in tagging the CANCAUSE structure of Fig. 5-15 as "USED", noting that it probably won't be useful again? Indeed, having used a CANCAUSE structure such as this once, it is perhaps more to the point to deactivate it and *generate another* CANCAUSE structure which would indicate that, should Mary receive *another* mitt, she will probably say "Thanks, but I already have one." There are many problems with knowing what to do with this kind of CANCAUSE information after it has been used one or several times. I have not pursued them.

### 5.5.4 IMPLEMENTING CAUSAL CHAIN EXPANSION

We have enough now to describe the process of causal chain expansion. Language-communicated causals, (CAUSE C1 C2) or (CANCAUSE C1 C2), are detected in the input during inferencing by the CAUSE inference molecule. For real-world events (ie. there is some concrete time aspect associated with the causal relation itself, or, equivalently, with C1 and C2), the CAUSE inference molecule places C1 and C2 on the inference queue, then calls the service function RECORD_CAUSAL, which places the pair (C1.C2) on a global list, !CAUSALS.

Conceptual causal configurations of the form




(that is, something causing a causal, a causal causing something, or a causal causing another causal) are detected by RECORD_CAUSAL as special cases. This is a heuristic which is made necessary by the language use of causals, as cases c,f,g,h,i of Fig. 5-8 illustrated. Conceptual forms like this will arise for which the expanded causal explanation might have the respective forms:

X <=== ... <=== Y <=== ... <=== Z

for the first two forms, and

X <=== ... <=== Y <=== ... <=== Z <=== ... <=== W

for the third form. That is, in addition to the existence of a path from Y-Z in the first form, X-Y in the second, and X-Y and Z-W in the third, there might also exist longer paths from X-Z in the first and second forms, and from X-W in the third form. Hence, to understand these three forms, these longer paths must also be explained.

Having recorded these language-communicated causals and placed C1 and C2 on the inference queue, inference spheres will begin to expand around C1 and C2 (in parallel, and along with many other structures on the inference queue). And in particular, this expansion will include causative and resultative inferences from C1 and C2. If an explicable causal path exists between language-communicated causals, then some causative inference path on C2's sphere will eventually intersect with some resultative inference path on C1's sphere.

Recall that as each new inference (of *any* theoretical type) is generated it is evaluated for confirmation, contradiction or augmentation. At some point some inference lying on a resultative chain from C1 will *confirm* (match) some inference lying on a causative chain from C2. Since this intersection is detected by the inference evaluator, which is part of the inference monitor, this function must always be aware of pending "causals" on !CAUSALS. This means that for each confirmation which arises as the result of a causative or resultative inference, causal chains in both directions away from this confirmation point must be scanned in order to detect whether one in the causative direction matches some left member on !CAUSALS and one in the resultative direction matches the corresponding right member.

When this occurs, the structures which, when matched, established the point of contact between C1' resultative line and C2's causative line are *merged* into one structure, S, thus completing a causal chain between C1 and C2. In addition, (C1.C2) is removed from !CAUSALS, the association:

(S C1 C2)

is placed on another list, !EXPANDED_CAUSALS, which simply maintains a record of successful causal chain expansions. At the end of all inferencing, the list !CAUSALS provides an important source of MEMORY-generated questions for those causal chains which could not be explained.

It should be clear that there is little goal direction to this process. Since the theory I am

proposing predicts that a human language user automatically performs these large expansions in inference space, the process of causal expansion is scarcely more than an important byproduct of the expansion. But is a very important one, and failures to explain causals at this "subconscious" level provide motivation to higher level processes which might attempt special heuristic analysis to explain causal chains.

### 5.5.5 ANOTHER TASK RELATED TO LANGUAGE USE OF CAUSALITY

There is another aspect of the language use of conceptual causals which has not been addressed in the current implementation of the theory, but which deserves mention. It is this: conceptual causals are frequently used not to convey causality between the two events they *appear* to relate, but rather to convey *the cause of the speaker's belief* that an event occurred. For example, "John must have come because his car is here" will be analyzed conceptually as shown in the left of Fig. 5-16. However, one potential meaning, which we would like the memory to be able to discover, is shown in the right of Fig. 5-16: "The reason the speaker believes John came is because John's car is here."

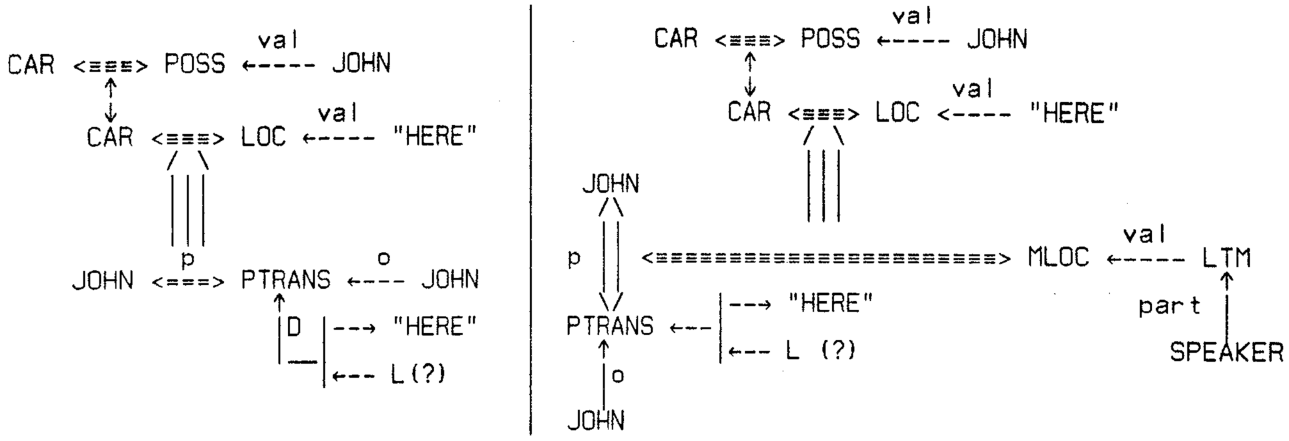



**Figure 5-16.**

### 5.5.6 PRESERVING CAUSAL CONNECTIVITY

**sample:** John kicked the dog.
The dog bit John.

**sample:** John's hitting Mary pleased Bill.

**sample:** John rubbed Mary's sore back.
She kissed him.

**The memory must do more than simply generate resultative and causative inferences. It must also make explicit the underlying causal relations themselves.**

There are many inferences which are triggered by some state of the world, but which require in addition to the existence of the state, information about what *caused* that state to exist. There are other inferences which rely on explicit information about what caused what in order to predict actors' intentions. These are two of several reasons why the memory needs to preserve causality relations as explicit structures.

Consider the second sample sentence above: "John's hitting Mary pleased Bill.", whose underlying conceptual representation is shown in Fig. 5-17. Among other things, we would like one of the memory's potential responses in suitable contexts: to be "Why doesn't Bill like Mary?". That is, it will be insightful to discover how we can get from the original utterance to this question, regardless of whether such a response is actually ever generated. I will show here the processing which underlies a response of this sort.

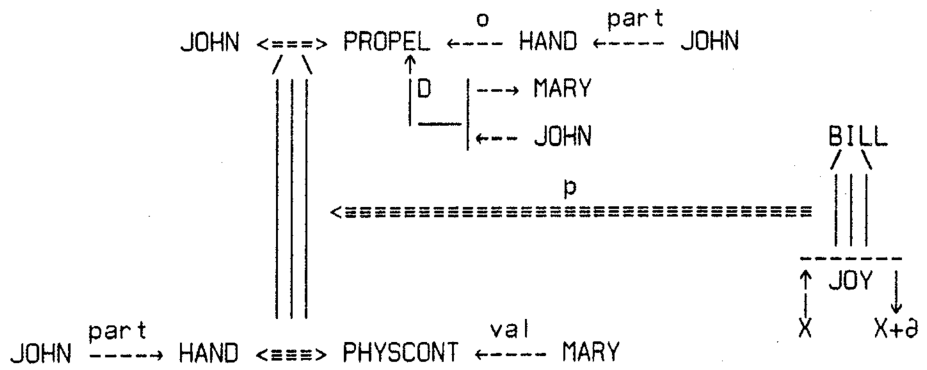



**Figure 5-17.**

In this utterance we must ask the question "Why was Bill pleased?" The chances are that it was neither John's act of PROPELling nor the PHYSCONT, nor even the entire causal relationship (the fact that his propelling caused a physical contact). Rather, what actually caused Bill's pleasure was likely to have been some other *inferred result* of this conceptualization, namely that Mary became hurt:

(NEGCHANGE #MARY #PSTATE)

That is, although the NEGCHANGE is only an *inferred* result of this utterance, it's importance to explaining Bill's pleasure is foremost.

The memory must therefore realize that, when an event is stated to have caused a statechange of some person on some scale, it is quite possible that not the event itself, but rather some other inferrable result of the event was in reality the cause of the statechange. In order to do this, the memory must keep track of possible causals of this nature: it would not be acceptable to forget that (NEGCHANGE #MARY #PSTATE) (having arisen from the input) might in fact be the cause of Bill's pleasure. Were this to happen, the belief pattern (causative inference):

**X undergoes a NEGCHANGE on some scale**
**CAUSEs**
**Y to undergo a POSCHANGE on the JOY scale**
**implies that**
**Y has a negative relationship with X**

would never be accessed. That is, Bill's POSCHANGE has added significance when it has been CAUSEd by Mary's NEGCHANGE. Without remembering its cause, an important inference about Bill and Mary's relationship would be altogether missed. This is illustrated in Fig. 5-18.

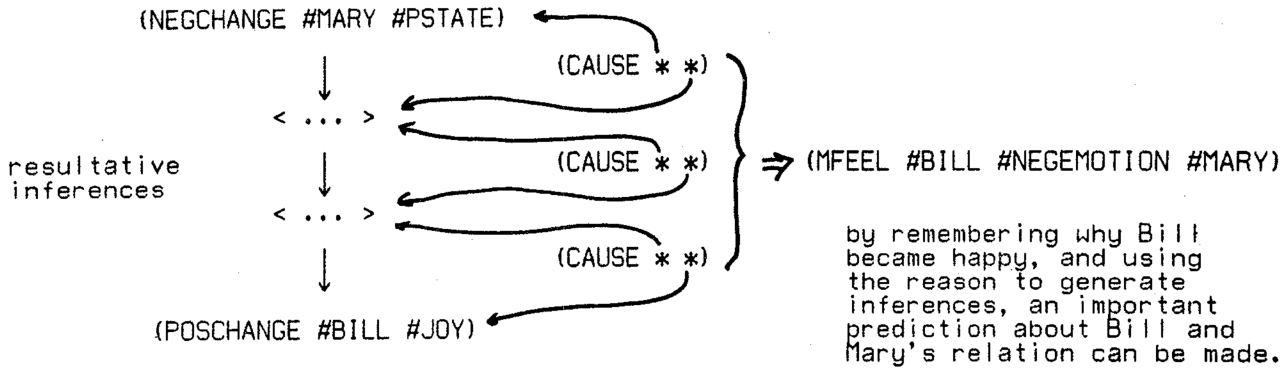



**Figure 5-18. Inferences which are based on causal relations.**

This is very general principle:

> **The significance of ANY information can be highly dependent upon HOW that information came into existence -- its surrounding causal environment.**

Two other examples are: "The hammer had come to rest on the vase. John had flung it across the room." and "Mary gave John a tool he had been wanting." In the first example, the first sentence communicates a PHYSCONT relation. However, the result of such a relation is hard to assess in the absence of information about how it came about. When we discover in the next line that it was a fairly vigorous PROPEL, the

(CAUSE (PROPEL ...) (PHYSCONT ...))

information makes possible the prediction that the vase may have been broken as the result of a PHYSCONT in this causal context. From the utterance "Mary gave John the tool he had been wanting" a probable resultative inference is that John experiences a POSCHANGE in JOY. If the memory were not to remember that it was Mary's action which caused this POSCHANGE, it would not be possible to apply the crucial pattern: "people usually MFEEL a positive emotion toward people who cause them POSCHANGES", thus inferring that John started feelin a positive emotion toward Mary.

I will label this carrying-along of causal information with every causative or resultative inference with the ominous title: *causal connectivity preservation.* Adherence to this principle means that

> **Anytime a causative or resultative inference is generated, its causal relation to the information from which it was generated must be generated and stored as well.**

Furthermore, the REASONS associated with the explicit CAUSE structure thus generated are those other information structures in the memory which were used by the inference molecule to generate the resultative or causative inference. Fig. 5-19 illustrates this: because Mary gave John something he wanted, he experiences a poschange, and because Mary was responsible for this poschange, he MFEELs a positive emotions toward her.

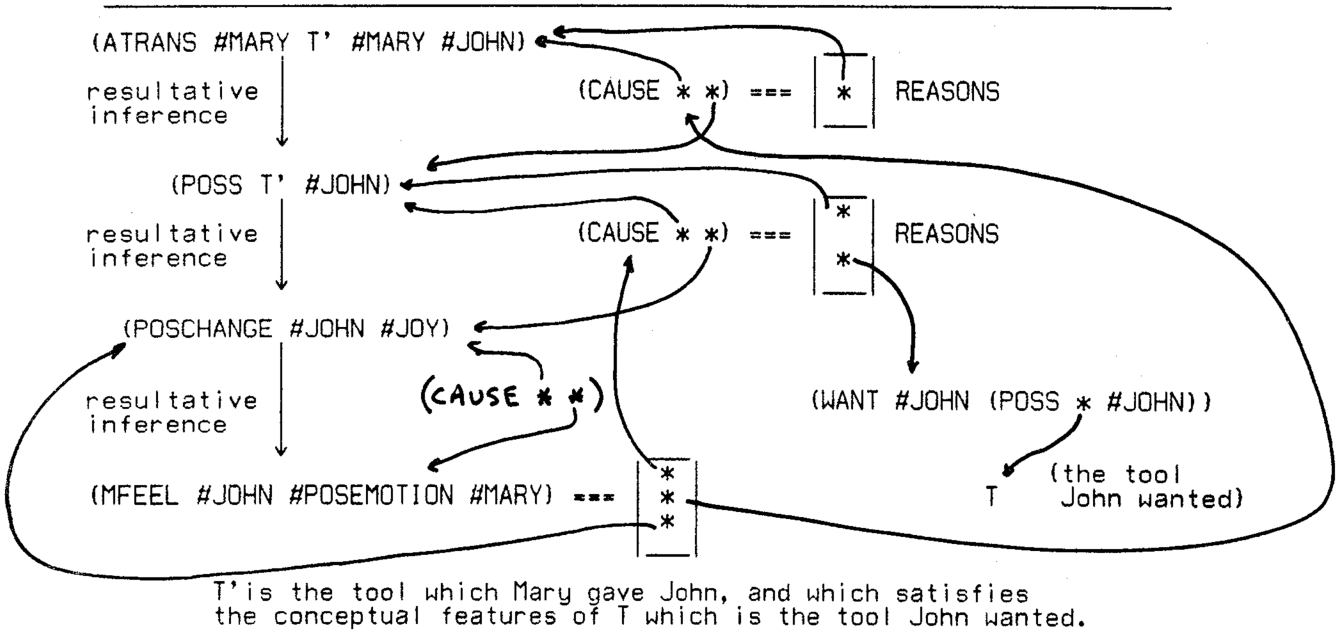



**Figure 5-19. How REASONS, CAUSE, resultative and causative inferences are related.**

## CAUSAL CHAIN EXPANSION COMPUTER EXAMPLE

In this example, we assume Mary doesn't like Bill very much. In this context, the memory receives the conceptual graph underlying "Mary kissed John because he hit Bill" (Fig. 5-20), in which there are three language causals which the memory must explain in terms of its knowledge of causality. I have suppressed the two less important causal expansions, and have focused on the main one: how could John's hitting action have caused Mary to kiss John? The particular path discovered in this example is six memory structures long, involving five intervening causals. Fig. 5-21. shows the path, as it would be decribed in English, and we will have a look at the internal structures at the end of the computer example.

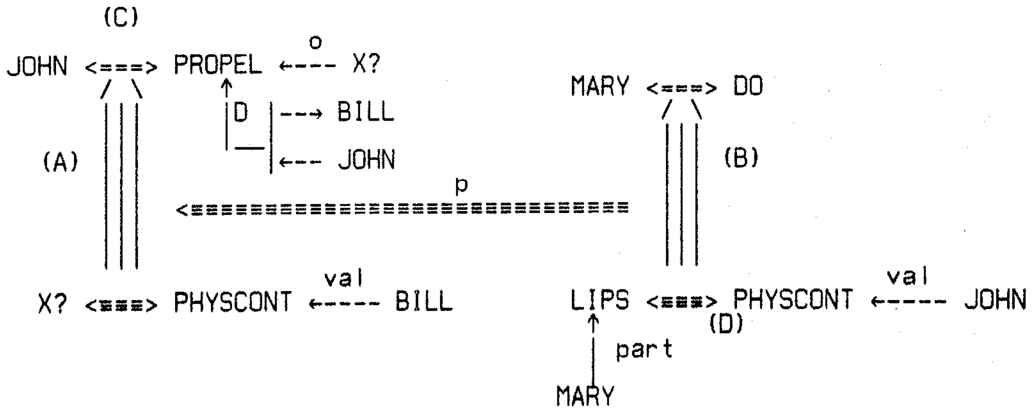



**Figure 5-20. Mary kissed John becaues he hit Bill.**

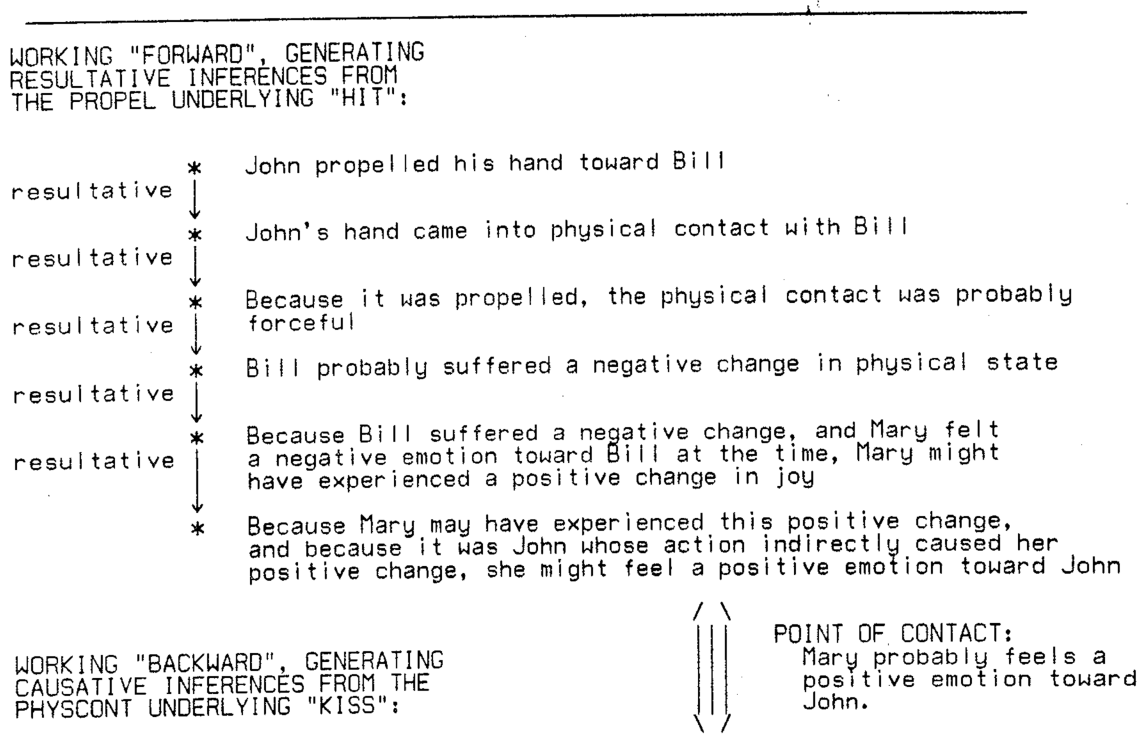



**Figure 5-21. One explanation of why Mary's kissing was related to John's hitting.**

---

```
(MARY KISSED JOHN BECAUSE HE HIT BILL)

(((CON ((CON ((ACTOR (JOHN) <=>
(*PROPEL*) OBJECT (*PHYSOBJ* SPEC
(*U*)) FROM (JOHN) TO (BILL)) TIME
(TIM01)) <= ((ACTOR (*PHYSOBJ* SPEC
(*U*)) <=> (*PHYSCONT* VAL (BILL)))
TIME (TIM01)))) <= ((CON ((ACTOR
(MARY) <=> (*DO*)) TIME (TIM02) SPEC
(*U*)) <= ((ACTOR (*LIPS* PART (MARY))
<=> (*PHYSCONT* VAL (JOHN))) TIME
(TIM02))))) TIME (TIM01))

(TIM00 ((VAL T-0)))
(TIM01 ((BEFORE TIM02 X)))
(TIM02 ((BEFORE TIM00 X)))
```

This is the input sentence. Its underlying conceptual graph is shown next.

```
((CAUSE ((CAUSE ((*PROPEL* (#JOHN1)
(C0013) (#JOHN1) (#BILL1)) (TIME _
(C0016))) ((*PHYSCONT* (C0013) (#BILL1))
(TIME _ (C0016))))) ((CAUSE ((*DO*
(#MARY1) C0010) (UNSPECIFIED _) (TIME _
(C0017))) ((*PHYSCONT* (C0021) (#JOHN1))
(TIME _ (C0017)))))) (TIME _ (C0016)))
```

This is the partially integrated memory structure, after references have been established. No reference ambiguity is assumed to exist for this example.

```
C0035
```

C0035 is the resulting memory structure for this utterance.

---

```
STARTING INFERENCE QUEUE:
        ((X 1.0 C0035))

        ......
```

We suppress all but this structure on the starting inference queue.

(We will be seeing about one fourth of the original trace output for this example)

```
ABOUT TO APPLY @CAUSE1 TO C0035
C0035: (CAUSE (CAUSE (*PROPEL* #JOHN1
   C0013 #JOHN1 #BILL1) (*PHYSCONT*
   C0013 #BILL1)) (CAUSE (*DO* #MARY1
   C0010) (*PHYSCONT* C0021
   #JOHN1)))
INFERRING: C0028

ABOUT TO APPLY @CAUSE2 TO C0035
C0035: (CAUSE (CAUSE (*PROPEL* #JOHN1
C0013 #JOHN1 #BILL1) (*PHYSCONT*
C0013 #BILL1)) (CAUSE (*DO* #MARY1
C0010) (*PHYSCONT* C0021 #JOHN1)))
INFERRING: C0034
```

Here, the CAUSE inference molecule is injecting the two subconceptualizations, A and B in Fig. 5-20, into the inference stream.

---

```
RECORDING CAUSAL RELATION:
        (C0024 . C0032)
```

The causal structure of this conceptualization indicated that a path should be found relating structure C to structure D in Fig. 5-20. This is noted. C0024 corresponds to C, C0032 to D.

---

```
        ......

ABOUT TO APPLY @PHYSCONT1 TO C0032
C0032: (*PHYSCONT* C0021 #JOHN1)
INFERRING: (*MFEEL* #MARY1 #POSEMOTION
                #JOHN1)
   ALSO GENERATING: (TIME C0039 C0017)

        ......

ABOUT TO APPLY @PROPEL1 TO C0024
C0024: (*PROPEL* #JOHN1 C0048 #JOHN1
                #BILL1)
INFERRING: (*FORCECONT* C0048 #BILL1)
   ALSO GENERATING: (TS C0052 C0016)

        ......

ABOUT TO APPLY @FORCECONT2 TO C0052
C0052: (*FORCECONT* C0048 #BILL1)
INFERRING: (NEGCHANGE #BILL1 #PSTATE)
   ALSO GENERATING: (TIME C0055 C0016)

        ......
```

Here, the causative inference that Mary's kissing was probably caused by her feeling a positive emotion toward John is made.

Because the PHYSCONT of John's hand and Bill was caused by a PROPEL, MEMORY here makes the inference that it was a forceful contact.

Since one of the objects involved in the FORCECONT was a person, MEMORY predicts a small NEGCHANGE on his part. The degree of the NEGCHANGE is dependent upon the type of object which cane into contact with him.

```
ABOUT TO APPLY @NEGCHANGE2 TO C0055
C0055: (NEGCHANGE #BILL1 #PSTATE)
INFERRING: (POSCHANGE #MARY1 #JOY)
   ALSO GENERATING: (TIME C0061 C0016)

        ......

ABOUT TO APPLY @POSCHANGE1 TO C0061
C0061: (POSCHANGE #MARY1 #JOY)
INFERRING: (*MFEEL* #MARY1 #POSEMOTION
                #JOHN1)
   ALSO GENERATING: (TS C0068 C0016)
```

Here, because Mary was feeling a negative emotion toward Bill at the time, when Bill underwent a small NEGCHANGE, the prediction can be made that Mary may have experienced a degree of joy.

Looking back the causal path which lead to Mary's likely change in joy, the POSCHANGE inference molecule discovers that it was an action on John's part which was most directly responsible for her joy. The inference that Mary might have started feeling a positive emotion toward John is made.

---

```
CAUSAL EXPANSION ACHIEVED:
        (C0024 . C0032)
CONTACT POINTS ARE: (C0068 C0039)

MERGING:
C0068: (*MFEEL* #MARY1 #POSEMOTION
                #JOHN1)
C0039: (*MFEEL* #MARY1 #POSEMOTION
                #JOHN1)

        ......
```

As this last inference is made, the inference evaluator notices that the same information exists elsewhere in the memory. This is a point of contact in inference space. It is furthermore noticed that the two MFEEL structures join a causal path between two structures which have been related causally by language. The two MFEEL structures are merged into one, and this event is noted as a causal chain expansion. To the left, C0068 and C0039 are the contact points, C0024 and C0032 are the two structures which have now been causally related.

---

```
*!EXPANDED_CAUSALS

((C0024 . C0032))

*(CAUSAL_PATH @C0024 @C0032)

(C0024 C0052 C0055 C0061 C0068 C0032)
```

Inference proceeds, and finally stops. At that point, we took a look at the structures lying along this explained causal path. C0024 is the original PROPEL structure, C0032 is the PHYSCONT-lips structure. The service function CAUSAL_PATH will track down the causal linkage for us. The causal chain consists of the six structures to the left.

---

```
-----------------------------------
C0024: (*PROPEL* #JOHN1 C0048 #JOHN1
                #BILL1)
ASET:
   C0054: (CAUSE # C0052)
   C0028: (CAUSE # C0026)
   C0025: (TIME # C0016)
RECENCY: 9900
TRUTH: T, STRENGTH: 1.0
REASONS:
   C0028: (CAUSE C0024 C0026)
OFFSPRING:
   C0070: (CAUSE C0061 C0068)
   C0068: (*MFEEL* #MARY1 #POSEMOTION
                #JOHN1)
   C0065: (CAUSE C0055 C0063)
   C0063: (*MFEEL* #BILL1 #NEGEMOTION
                #JOHN1)
   C0054: (CAUSE C0024 C0052)
   C0053: (TS C0052 C0016)
   C0052: (*FORCECONT* C0048 #BILL1)
ISEEN: (@PROPEL1)
```

This is the original PROPEL. During the process, but not shown, C0048 was detected as unspecified, and filled in as John's hand. Notice on the REASONS and OFFSPRING sets the results of other inferencing which was not discussed above.

```
-----------------------------------
C0052: (*FORCECONT* C0048 #BILL1)
ASET:
   C0077: (WANT #JOHN1 #)
   C0057: (CAUSE # C0055)
   C0054: (CAUSE C0024 #)
   C0053: (TS # C0016)
RECENCY: 18416
TRUTH: T, STRENGTH: 0.89999999
REASONS:
   C0024: (*PROPEL* #JOHN1 C0048 #JOHN1
           #BILL1)
OFFSPRING:
   C0078: (TS C0077 C0016)
   C0077: (WANT #JOHN1 C0052)
   C0057: (CAUSE C0052 C0055)
   C0056: (TIME C0055 C0016)
   C0055: (NEGCHANGE #BILL1 #PSTATE)
ISEEN: (⊗FORCECONT2)
```

Here is the FORCECONT which was inferred from the PROPEL.

```
-----------------------------------
C0055: (NEGCHANGE #BILL1 #PSTATE)
ASET:
   C0079: (WANT #JOHN1 #)
   C0067: (CAUSE # C0059)
   C0066: (CAUSE # C0061)
   C0065: (CAUSE # C0063)
   C0057: (CAUSE C0052 #)
   C0056: (TIME # C0016)
RECENCY: 19833
TRUTH: T, STRENGTH: 0.85500000
REASONS:
   C0052: (*FORCECONT* C0048 #BILL1)
   I0008: (ISA #BILL1 #PERSON)
OFFSPRING:
   C0080: (TIME C0079 C0016)
   C0079: (WANT #JOHN1 C0055)
   C0067: (CAUSE C0055 C0059)
   C0066: (CAUSE C0055 C0061)
   C0065: (CAUSE C0055 C0063)
   C0064: (TS C0063 C0016)
   C0063: (*MFEEL* #BILL1 #NEGEMOTION
           #JOHN1)
   C0062: (TIME C0061 C0016)
   C0061: (POSCHANGE #MARY1 #JOY)
   C0060: (TS C0059 C0016)
   C0059: (WANT #BILL1 C0058)
ISEEN: (⊗NEGCHANGE3 ⊗NEGCHANGE2
        ⊗NEGCHANGE1)
```

This is Bill's likely (small) change in PSTATE which resulted from the FORCECONT.

```
-----------------------------------
C0061: (POSCHANGE #MARY1 #JOY)
ASET:
   C0070: (CAUSE # C0068)
   C0066: (CAUSE C0055 #)
   C0062: (TIME # C0016)
RECENCY: 24616
TRUTH: T, STRENGTH: NIL
REASONS:
   C0055: (NEGCHANGE #BILL1 #PSTATE)
   I0137: (MFEEL #MARY1 #NEGEMOTION
           #BILL1)
OFFSPRING:
```

This is the important inference that Bill's NEGCHANGE may have cause a small degree of happiness in Mary. Notice that one of the REASONS was assumed to be the case beforehand (I0137). In section 7.6 we will see other aspects of this same example which illustrate how structure merging occurs. But there, that Mary may feel a negative emotion toward Bill will be established as an inference from a preceding input instead of being assumed as a starting condition.

```
   C0070: (CAUSE C0061 C0068)
   C0069: (TS C0068 C0016)
   C0068: (*MFEEL* #MARY1 #POSEMOTION
                  #JOHN1)
         ISEEN: (⊗POSCHANGE1)
```

---

```
C0068: (*MFEEL* #MARY1 #POSEMOTION
                #JOHN1)
ASET:
   C0085: (WANT #JOHN1 #)
   C0040: (TIME # C0017)
   C0044: (*MLOC* # C0041)
   C0047: (CAUSE # C0032)
   C0070: (CAUSE C0061 #)
   C0069: (TS # C0016)
RECENCY: 27366
TRUTH: T, STRENGTH: 0.95000000
REASONS:
   C0061: (POSCHANGE #MARY1 #JOY)
   C0024: (*PROPEL* #JOHN1 C0048 #JOHN1
                #BILL1)
   C0044: (*MLOC* C0068 C0041)
OFFSPRING:
   C0087: (TS C0085 C0016)
   C0086: (TIME C0085 C0017)
   C0085: (WANT #JOHN1 C0068)
ISEEN: NIL
```

Here, Mary is feeling a positive emotion toward John, whose action indirectly caused her joy. This structure is the point of contact, and is the structure which resulted from the merge. Notice that its STRENGTH has assumed the higher STRENGTH of the two structures which were merged.

---

```
C0032: (*PHYSCONT* C0021 #JOHN1)
ASET:
   C0088: (WANT #JOHN1 #)
   C0071: (WANT #MARY1 #)
   C0047: (CAUSE C0068 #)
   C0046: (CAUSE # C0044)
   C0034: (CAUSE C0029 #)
   C0033: (TIME # C0017)
RECENCY: 12016
TRUTH: T, STRENGTH: 1.0
REASONS:
   C0034: (CAUSE C0029 C0032)
OFFSPRING:
   C0089: (TIME C0088 C0017)
   C0088: (WANT #JOHN1 C0032)
   C0072: (TIME C0071 C0017)
   C0071: (WANT #MARY1 C0032)
   C0047: (CAUSE C0068 C0032)
   C0046: (CAUSE C0032 C0044)
   C0045: (TS C0044 C0017)
   C0044: (*MLOC* C0068 C0041)
   C0040: (TIME C0068 C0017)
ISEEN: (⊗PHYSCONT2 ⊗PHYSCONT1)
```

This is the original PHYSCONT-lips structure which lead, via a causative inference to the prediction that Mary may have felt a positive emotion toward John.

This WANT is a prediction that one reason Mary may have kissed John is so that he would know she felt a positive emotion toward him.
This MLOC represents the inference that John probably now knows that Mary MFEELS a positive emotion toward him. We will account for these types of inference in upcoming sections.

## 5.6 MOTIVATIONAL INFERENCES: ACTIONS AND INTENTION

**sample:** John hit Mary.
John probably wanted Mary to be hurt.

**sample:** John told Mary that Bill wants a book.
John may want Mary to give Bill a book.

**sample:** John set out to the grocery.
John probably wanted to be at the grocery.

**sample:** Mary stabbed herself with a knife.
Mary probably wanted to die.

**sample:** Bill went to the store.
Bill probably wanted to be at the location of the store.

**sample:** Mary pointed out to Bill that he hadn't done his chores.
Mary may have wanted Bill to feel guilty.

**sample:** Rita liked Bill.
Mary kissed Bill in front of Rita.
Mary may have wanted Rita to become jealous.

When dealing with conceptual information which involves people and their actions in the world, it is of considerable importance to be able to to separate what *actually* happens by way of actions from what is *intended* to happen by an actor who has performed some action. That is, the *intentions* of actors, and what motivates them to those intentions are very important. The notion of a *motivational inference* deals with this distinction between the actual and the intentional levels of events in the world. Motivational inferences hence always relate the internal states and actions of *people*. In this section I will describe the idea behind a motivational inference.

To illustrate, let us return to our battle-fatigued example, "John hit Mary." Again, the underlying conceptual content of this utterance is shown in Fig. 5-22.

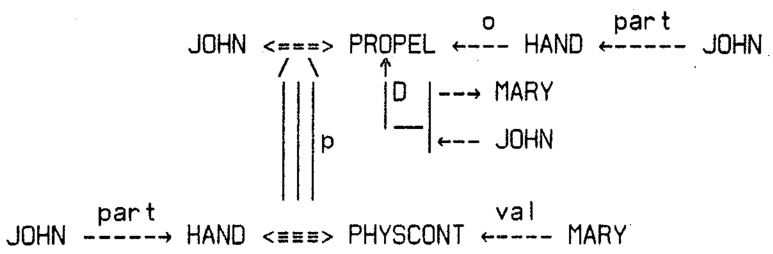



**Figure 5-22.**

There are lines of inference which arise directly from the *mechanics* of the event which this structure conveys. Inferences in these lines begin with the main causal predication, inferring that both events or states which the central causal structure of Fig. 5-22 relates actually occurred. Inferences organized under PROPEL and PHYSCONT (these two inference molecules) will thus be called into play, and they in turn will lead to various other inferences which deal with the explicit contents of the conceptualization as expressed. In this case, these inferences result in predictions about the physics of hitting, what happens to a person when a PROPEL causes the physical contact of some object with that person, what enabling states (next section) must have been in effect at the time, and so on. These are all extremely interesting conceptual inferences.

But there is another level at which every utterance can be simultaneously analyzed. This level concerns inferences about humans: in particular, their motivations and reasons for performing actions in the world. An analysis of motivations will eventually lead to other inferences relating to their social interactions. Hence, much of the interesting content of this conceptualization would be lost if only those inferences explicitly activated by the input -- those concerning the mechanics of the event -- were generated.

In this example, the missing link between the literal description of the situation and this other level of intentionality is the following simple fact: John probably *intended* to hit Mary. That is, John probably preconceived that his PROPELling would cause the PHYSCONT, and, in general, the probable consequences of the PHYSCONT as well. Thus, aside from the analysis of the mechanical facets of this utterance there is another entire realm which is entered only by making this crucial prediction. This example characterizes a very general principle: analyzing both sides of this duality between the actual and intentional is crucial at all levels of inference. The primary means of accomplishing it is to have the memory assume that every real-world action *might have been* volitional (in the absence of explicit information to the contrary). It is the purpose of this section to describe the mechanism by which motivations of actors can be inferred.

### 5.6.1 RESULTATIVE INFERENCES AND MOTIVATION

A first approximation to drawing out what an actor wanted an action to achieve can be had by asking "What actually happened as results of his action in the context in which he performed it?" That is, what conceptual *resultative* inferences could the memory make *from the mechanics* of

the hitting action? If these can be ascertained then they are good *candidates* for things John may have had in mind as results of his action.

> **That is, an actor's desire for one or more of the results of his action might have been what motivated him to perform the action.**

Resultative inferences are easily locatable in the memory, since preserving causal connectivity is one important byproduct of the causative/resultative inference process: in order to locate the results of actor AC's action, A, which is under analysis to generate motivational inferences, the motivational inference process, ASSERT_WANT, can simply gather the set of structures, Ri, which lie in the relation (CAUSE* A Ri) with A. The result of such a retrieval is a set {R1,...,Rn} which is the set of all structures were predicted by the memory to have been results of A, all structures which were in turn the results of the results, and so on.

### 5.6.2 MODELING THE ACTOR'S KNOWLEDGE

In a first-approximation, the memory could at that point simply generate the motivational inferences (WANT AC Ri) for each Ri in this set. That is, *many* probablistic predictions (Fig. 5-23) about what the actor, AC, wanted could be made. But this is obviously a fairly crude approximation: although it would encompass everything the memory could infer by way of results of action A, it is based on *the memory's* characterization of cause and effect in the world, and on the memory's specific world knowledge at the time, *not* on actor AC's knowledge. Since the actor's knowledge of the environment in which he performed his action is clearly more relevant than the memory's for the purposes of predicting his intentions, this difference must be taken into account. {R1,...,Rn} must be thought of only as *candidates* for what the actor may have had in mind.

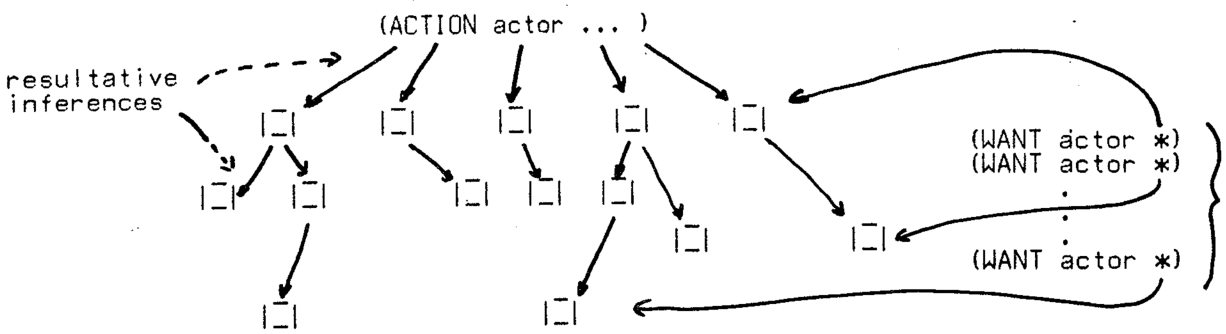



**Figure 5-23. A first approximation: candidates for motivational inferences.**

The memory must realize that its own knowledge is not necessarily the same as the actor's. Where is this modeling of the actor's knowledge at the time of his action injected into this process of making motivational inferences? Recall that each Ri in the result set MEMORY generates from action, A, has an associated set of REASONS. For resultative inference, Ri, this is a list X1,...,Xk of other structures in the memory which were factors in Ri's generation as a resultative inference. Therefore, to ask the question "was AC also able to make this resultative" inference is to ask "did AC have access to information X1,...,Xn at the time of his action?" If it can be predicted that he did, then it is also reasonable to infer that he may have been aware of the same Ri as memory.

Thus (Fig. 5-24), in order to predict (WANT AC Ri), the memory must ascertain that AC knew X1,...,Xk: that is, (MLOC Xj L), where L is AC's LTM or CP, and $1 \leq j \leq k$. If it can ascertain this, the motivational inference (WANT AC Ri) can be generated, and given the following REASONS: (a) the action occurred, (b) the action CAUSEd Ri (perhaps through several levels of resultative inference -- all the CAUSE structures are explicit structures generated by the resultative inference process), and (c) the MLOC structures which represent the actor's knowledge of X1,...,Xk at the time of his action.

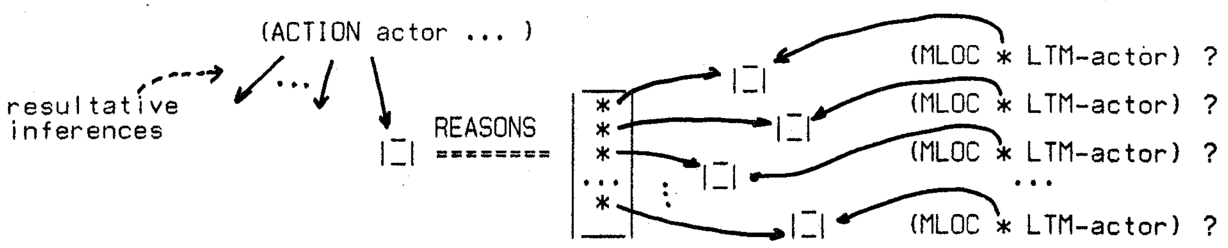



**Figure 5-24. Where predictions about the actor's knowledge fit.**

Since knowledge propagation inferences (section 6.6) require the same knowledge modeling ability for a slightly more general process, rather than discuss here how the capability to make such predictions about an actor's knowledge at some time has been implemented, I will wait until that section. Suffice it to say here that memory's knowledge of what people *can be expected* to know, as well as what the memory explicitly knows they know, plays an important role in this modeling.

It is quite common that the memory simply will not be able to predict whether or not certain information essential to the generation of a motivational inference was in fact available to the actor at the time of his action. When this happens, it is far more desirable to have the memory make the tentative assumption that the actor *might* have had access to the same information the memory used to generate a particular resultative inference, Ri, and proceed with the motivational inference, (WANT AC Ri), on this assumption. That is, it is safer to be wasteful than to be too frugal, and thereby miss important points of contact in the inference space. This of course is a guiding principle of all inferences, but it is especially important with respect to motivational inferences, because modeling another person's knowledge in any detail is difficult, and requires tremendous quantities of data in a practical program. This is the approach taken in the current implementation: when it cannot be decided one way or the other what the actor knew, give him the benefit of the doubt. As we will see, "peculiar" motivational inferences thus generated will be detected by the inference evaluator, and re-evaluated.

I can summarize the main idea behind a motivational inference as follows:

**In the absence of information to the contrary, people can be assumed to perform actions for the probable consequences of those actions. That is, if an actor performs an action with the knowledge of what that action will result in, R1,...,Rn, then it can be inferred that the action was motivated by the actor's desire for one or more of R1,...,Rn.**

It should be emphasized that I am not concerned with discovering the *actual* intentions of an actor at some mysterious higher level than perhaps even he himself could explain. Rather I want only good commonsense predictions about what he might have had in mind as the outcome of his action, because predictions about what he may have been up to can lead to more points of contact in inference space. Without such predictions in this hitting example, the memory would miss altogether the important inferences about human interactions, which begins from the inferred pattern:

(WANT #JOHN (NEGCHANGE #MARY #PSTATE))

since (NEGCHANGE #MARY #PSTATE) is an eventual resultative inference which arises from the mechanics of hitting. This inferred structure will subsequently lead to inferences about MFEELing anger (a causative inference), and so forth.

### 5.6.3 IMPLEMENTATION OF MOTIVATIONAL INFERENCES

Motivational inferences are implemented in the memory via a special procedure, ASSERT_WANT, which is called by the inference monitor *after* the expansion of inferences resulting from some *action*. This may be viewed as an interruption to the operation of the inference monitor, and is part of the larger process POSTSCAN, which is described further in 7.2.2.

During this interruption, the following things occur within the POSTSCANner, relative to motivational inferences.

1. each action structure on the current queue of inferences is examined, and its actor, AC, isolated
2. a memory search is performed, gathering all resultative inferences which have been generated

from this action, A. This is the set of other structures, R1,...,Rn in the memory which lie in a (CAUSE A Ri) relation to A.

3. For each Ri, in turn, Ri's REASONS, X1,...,Xk are retrieved. For each Xj in this reasons set, it is determined whether AC knew or could be expected to have known Xi. If not all Xj can be assumed to have been known by AC, the motivational inference for Ri is not generated.
4. Otherwise, ASSERT_WANT then infers (WANT AC Ri) for each Ri collected in step (2). In addition, it makes explicit the probable causal relation: the desire for A's result, Ri, could have been the cause for AC's performing A.
5. All the new motivational inferences generated by (1)-(4) are placed on the inference queue for subsequent further expansion.

The inference monitor then proceeds. This process is depicted schematically in Fig. 5-25.

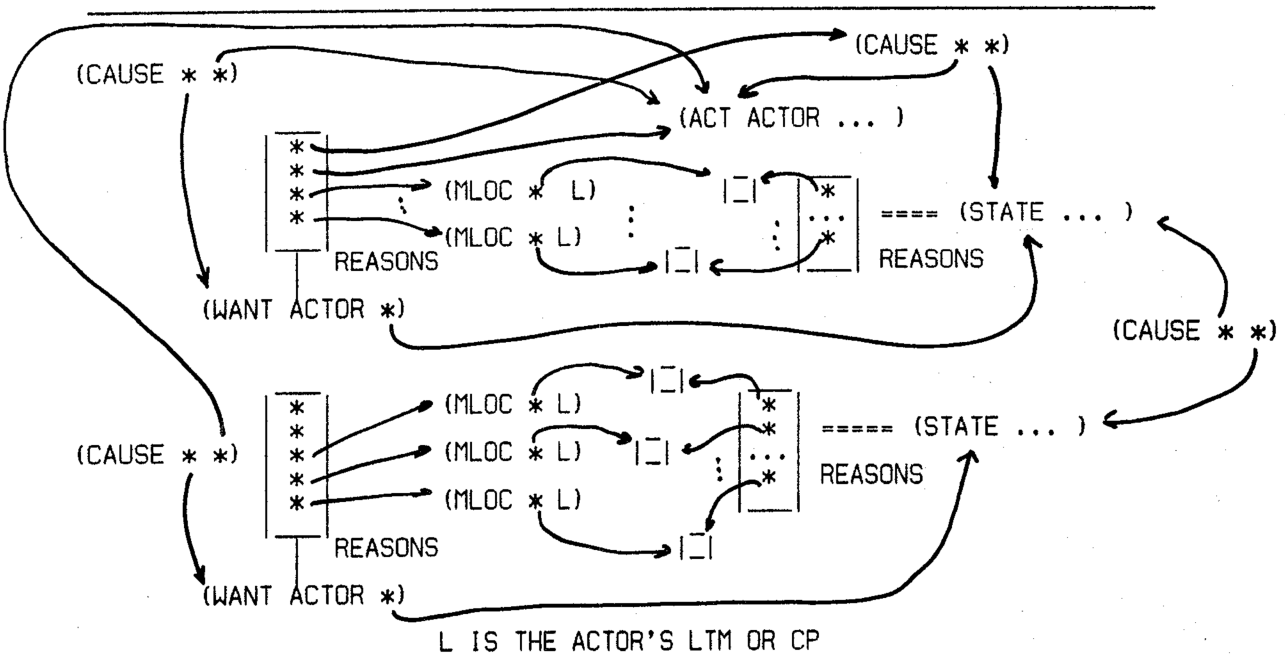



**Figure 5-25. The generation of motivational inferences.**

### 5.6.4 PROBLEMS WITH INTENTIONALITY: "PECULIAR" MOTIVATIONAL INFERENCES

The problem of assessing actors' intentionality is vast, and I claim only to have a

rudimentary ability to deal with intentions. Even with this ability to model an actor's knowledge at the time of his action, there are still difficult problems. The main one is the following: even though the actor may have been fully aware of the consequences of action A, in some particular environment, if those consequences led to NEGCHANGES for the actor himself, or, in some cases for others, it is often likely either (a) that the performing of the action itself was not volitional, or (b) that an incorrect model of the actor's knowledge was used. I do not pretend to have solved many of the deeper issues, but will indicate how one of them has been approached.

Consider the sentence "John ate a spoiled hamburger." Most adults know that eating spoiled meat leads (with a fairly high probability) to sickness. Because of this, the average hearer of this utterance would probably not infer that John WANTed to eat the meat, because people don't normally wish to induce sickness. But why is this, and how are we to have the memory realize such things?

The problem here is not with the intentionality of the *action* (John probably WANTed to perform the simple ACT of ingesting). Rather the problem has to do with a certain feature of the object of the INGEST and the consequences of the action which depended on that feature. The problem, therefore, is that, at the time the motivational inference process needs to know whether or not John knew the meat was spoiled, it could easily happen that the process which models this knowledge (normative inferences, section 6.7) simply cannot make a decision, based on the information conveyed by this sentence alone. The possibility that John knew this can not therefore be ruled out ("Mary wanted to end it all. She ate a spoiled hamburger."). The memory must proceed with the motivational inference that John may have WANTed the probable resultative structure (NEGCHANGE #JOHN #HEALTH).

It is at this point that the *inference evaluator* comes into the picture. As each new inference is generated, it is evaluated by the inference monitor which makes use of small programs called *normality molecules* (N-molecules, section 6.7.1). The function of these molecules is to assess the degree to which some unit of information agrees or disagrees with memory's other knowledge of the world. Hopefully, in a case such as this one, even though the inference

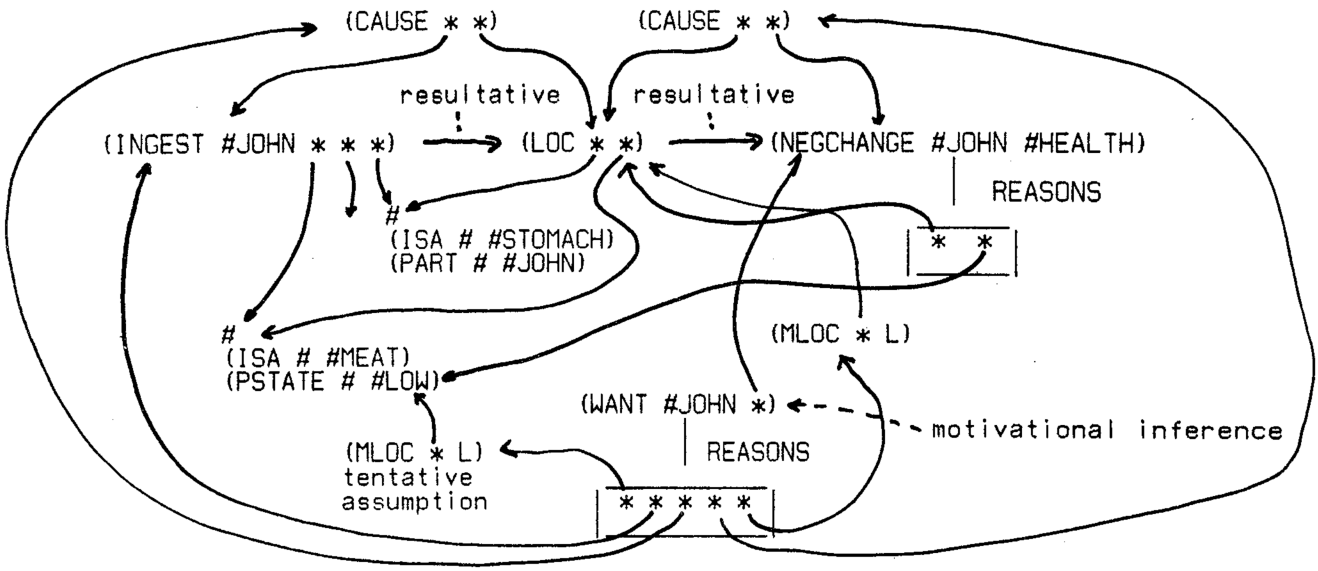
(WANT #JOHN (NEGCHANGE #JOHN #HEALTH))

is generated by the motivational inference process, it will be assessed very low by the WANT normality-molecule, because it will contradict memory's knowledge of normality in the world: people do not normally WANT NEGCHANGEs for themselves. (It is at this point -- in an N-molecule -- however, that the memory could make very specific tests about John before assessing this structure. That is, the WANT normality atom which assesses this structure could be sensitive to knowledge such as "John is a masochist", or whatever else might affect the assessment of this structure for normality.)

We need the services of the inference evaluator and N-molecules for the purposes of motivational inferences because of the following principle:

> **If a motivational inference is assessed by a normality molecule as being highly incompatible with memory's other knowledge of the world, its STRENGTH and the STRENGTHs of the tentative assumptions upon which it was based should be decreased. That is, it should be considered less likely because of the feedback from the evaluator.**

Since the assumptions upon which each motivational inference is based are stored as its REASONS, it is possible to retrieve them. Those which were "tentative" are the ones with low STRENGTHs to begin with (less than STRENGTH 0.50 in the current program). This heuristic prescribes that these assumptions, as well as the motivational inference they led to, should have their STRENGTHs severly decreased (to a quarter of their original value in the program). This has features of "backtracking" in that it will go back and alter the STRENGTH of tentative assumptions when those assumptions have led to "fuzzy inconsistencies" such as are detected by the normality molecules. This process is depicted in Fig. 5-26 for the spoiled hamburger example.




The pattern (WANT #JOHN (NEGCHANGE #JOHN #HEALTH)) is evaluated as incompatible with the memory's knowledge of what is normal. This causes the REASONS for the WANT structure above to be reexamined. All but the (MLOC (PSTATE M #LOW) L), where M is the meat he ate, have relatively high strengths. Sensing that this structure was a tentative assumption in the first place, its STRENGTH and the WANT structure's STRENGTH are both decreased.

L IS #JOHN'S LTM OR CP

**Figure 5-26. Retracing tentative assumptions.**

#### 5.6.4.1 DETECTING THE NON-INTENTIONALITY OF THE ORIGINAL ACTION ITSELF

What is the memory to do if *no tentative* assumptions can be located (that is, *all* the REASONS behind the motivational inference have very high STRENGTHs)? In this case, the performance of the *action itself* may have been accidental. That is, the actor might not have wished the action ever to be performed in the first place. "Bill dropped his camera" is a conceptualization in which this will occur. There are no low-STRENGTH assumptions on the REASONS list for the motivational inference (WANT #BILL (NEGCHANGE C #PSTATE)), where C is his camera: Bill can be assumed with near certainty to have known the consequences of dropping C, namely that this action could lead to the camera's demise. Because there are no tentative assumptions, as in the hamburger example, instead of reducing STRENGTHs, the action structure is marked with the conceptual modification (NONVOL A) as a prediction that the action was not

volitional. No more motivational inferences are generated from it or from any of its resulting structures, and those which have already been generated have their STRENGTHs reduced to a near zero value.

### 5.6.5 MOTIVATIONAL INFERENCES AND FUTURE ACTIONS

Although we have been examining the concept of a motivational inference from the point of view of predicting possible desires of actors who have performed actions, it should be clear that motivational inferences can also be used to understand why a person might desire a *future* action. That is to understand "John wants to give Mary a present", the memory can create a structure standing for this hypothetical future event, let resultative inferences arise from it, then predict that the reason John wants this action is because of the probable results it can achieve (perhaps to make Mary happy). On the other hand, *what John might do* to satisfy this desire to give Mary a present (for example, go to the store and buy it) concerns inferences which attempt to *predict* his future actions. These are distinct from, yet related to, motivational inferences which attempt to *explain* why people desire actions.

I will conclude with an example of how motivational inferences can be of use in understanding why someone might want to perform some future action. This example is shown in Fig. 5-27, and relates to the computer example in section 6.10: it shows how the utterance "John wanted an aspirin." can lead, through motivational inferences to the question "What happened to him?" Although Fig. 5-27 focuses on just one line of likely resultative inferences, bear in mind that there will generally be many others as well.

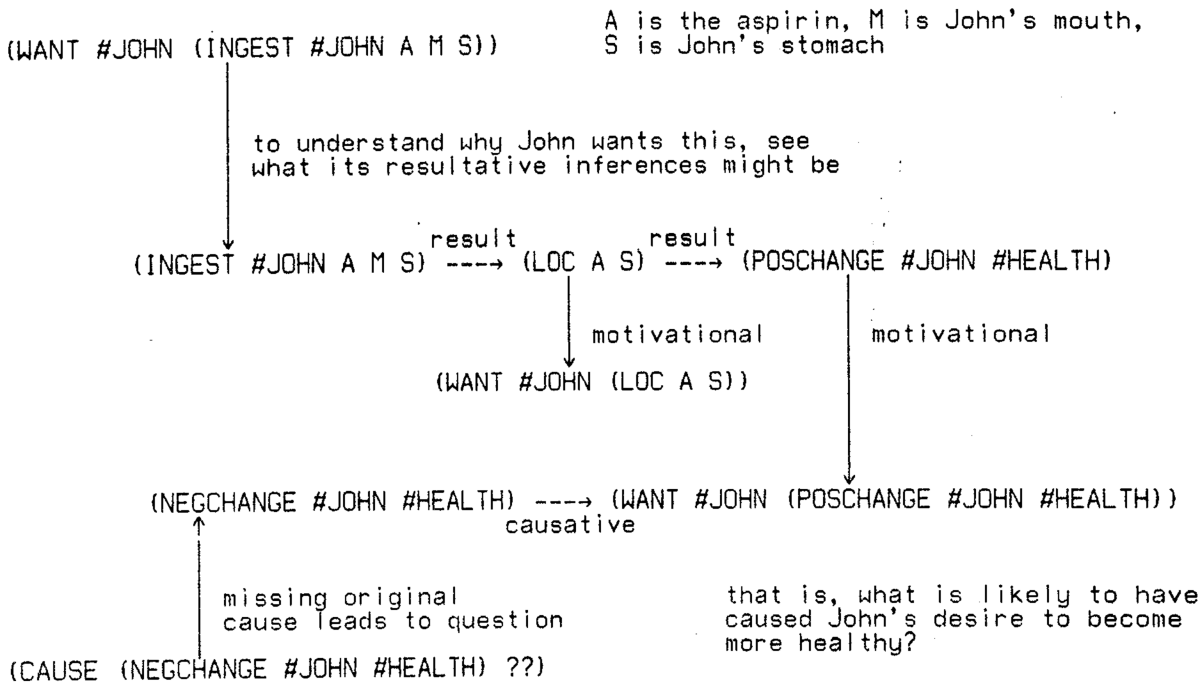



**Figure 5-27. How motivational inferences lead to interesting things.**

## MOTIVATIONAL INFERENCE COMPUTER EXAMPLE

This computer example illustrates the postscan mechanism as it concerns the generation of motivational inferences. The input sentence is "John hit Mary.", from which numerous inferences arise. Among others are the following resultative inferences: John's hand came into forceful contact with Mary, Mary suffered a negative change in physical state, and Mary began feeling a negative emotion toward John. During the postscan process, these inferences will be detected as resultative inferences from John's propelling action. Since they were predictable results of John's action, MEMORY will infer that John wanted them, and that wanting them motivated his action. One of the motivational inferences which is made (John wanted Mary to suffer some negative change) will lead to another inference: John probably wanted this because he felt a negative emotion toward Mary. This becomes a causative inference which begins the line of inferencing which will attempt to discover what caused him to feel this negative emotion.

```
JOHN HIT MARY

((CAUSE ((*PROPEL* (#JOHN1) (C0003)
(#JOHN1) (#MARY1)) (TIME _ (C0006)))
((*PHYSCONT* (C0003) (#MARY1))
(TIME _ (C0006)))))

C0011
```

To the left, the input sentence is being read and internalized in MEMORY structure C0011. In English this structure is: John propelled some physical object (C0003) from John to Mary, causing this physical object to come into physical contact with Mary. This happened at time C0006. During the course of inferencing, one task will be to specify this unspecified physical object. This is not shown, but it occurs behind the scene.

---

```
STARTING INFERENCE QUEUE:
((X 1.0 C0011))

        ......

ABOUT TO APPLY ⊗CAUSE2 TO
C0011: (CAUSE (*PROPEL* #JOHN1 C0003
        #JOHN1 #MARY1) (*PHYSCONT*
        C0003 #MARY1))

INFERRING: C0009

        ......

ABOUT TO APPLY ⊗PROPEL1 TO
C0007: (*PROPEL* #JOHN1 C0015 #JOHN1
        #MARY1)

INFERRING: (*FORCECONT* C0015 #MARY1)
   ALSO GENERATING: (TS C0019 C0006)

        ......

ABOUT TO APPLY ⊗FORCECONT2 TO
C0019: (*FORCECONT* C0015 #MARY1)

INFERRING: (NEGCHANGE #MARY1 #PSTATE)
   ALSO GENERATING: (TIME C0022 C0006)

        ......

ABOUT TO APPLY ⊗NEGCHANGE1 TO
C0022: (NEGCHANGE #MARY1 #PSTATE)

INFERRING: (WANT #MARY1 (C0025))
   ALSO GENERATING: (TS C0026 C0006)

        ......

ABOUT TO APPLY ⊗NEGCHANGE3 TO
C0022: (NEGCHANGE #MARY1 #PSTATE)

INFERRING: (*MFEEL* #MARY1 #NEGEMOTION
                #JOHN1)
   ALSO GENERATING: (TS C0028 C0006)

        ......
```

We suppress all subpropositions but the main one for this example.

I will show only the resultative inferences which are generated as results of this starting structure. To the left, C0009 (the second part of the CAUSE relation, that C0003 and #MARY1 were in PHYSCONT) is inferred.

Here, the probable result of propelling an object toward #MARY1 is that the object came into forceful contact with her. This inference is C0019. Notice that by this point in the inferencing, C0003 has been specified as C0015 (John's hand) which has replaced it.

One result of an object coming into forceful contact with a person is that the person suffers a negative change in his physical state. This inference is C0022.

A person who undergoes a negative change on some scale might begin wanting to undergo a positive change on that same scale. This inference could lead to predictions about the person's future actions which would tend to bring about this positive change.

Mary underwent a negative change. However, in addition, MEMORY detects that it was an action by John WHICH LEAD TO THIS NEGCHANGE. MEMORY thus infers that Mary begins feeling a negative emotion toward John.

---

```
ENTERING POSTSCANNER...

ACTION C0007: (*PROPEL* #JOHN1 C0015
```

Finally inference via the inference monitor ceases. The postscanner is called into action to scan the queue of inferences which exist

```
                #JOHN1 #MARY1)
DETECTED.

PERFORMING MOTIVATIONAL SCAN...
RESULT SET OF C0007 IS
(C0009 C0019 C0022 C0028 C0026)

        ......
```

so far. One function of the postscan process is to detect actions, locate those resultative inferences from those actions which have strengths high enough to be condidered "predictable", then infer that the actor of the action WANTed those results. To the left, the postscanner has located 5 such resultative inferences from John's propelling action. It generates 5 motivational inferences.

---

```
C0032: (WANT #JOHN1 C0009)
(WANT #JOHN1 (*PHYSCONT* C0015 #MARY1))
ASET:
   C0034: (CAUSE # C0007)
   C0033: (TIME # C0006)
RECENCY: 18666
TRUTH: T, STRENGTH: 1.0
REASONS:
   C0007: (*PROPEL* #JOHN1 C0015 #JOHN1
               #MARY1)
ISEEN: NIL
```

The inferences generated during the postscan are subjected to another pass through the inference monitor. I have interrupted MEMORY at that point to examine the motivational inferences and one other inference which resulted from one of the motivational inferences. To the left is the inference that John wanted his hand t be in PHYSCONT with Mary.

---

```
C0035: (WANT #JOHN1 C0019)
(WANT #JOHN1 (*FORCECONT* C0015 #MARY1))
ASET:
   C0037: (CAUSE # C0007)
   C0036: (TIME # C0006)
RECENCY: 18733
TRUTH: T, STRENGTH: 1.0
REASONS:
   C0007: (*PROPEL* #JOHN1 C0015 #JOHN1
               #MARY1)
ISEEN: NIL
```

John wanted his hand to be in forceful contact with Mary.

---

```
C0038: (WANT #JOHN1 C0022)
(WANT #JOHN1 (NEGCHANGE #MARY1 #PSTATE))
ASET:
   C0049: (CAUSE C0047 #)
   C0040: (CAUSE # C0007)
   C0039: (TIME # C0006)
RECENCY: 18783
TRUTH: T, STRENGTH: 1.0
REASONS:
   C0007: (*PROPEL* #JOHN1 C0015 #JOHN1
               #MARY1)
OFFSPRING:
   C0049: (CAUSE C0047 C0038)
   C0048: (TIME C0047 C0006)
   C0047: (*MFEEL* #JOHN1 #NEGEMOTION
               #MARY1)
ISEEN: (@WANT1)
```

John wanted Mary to suffer a negative change in physical state. Notice that this motivational inference has in turn lead to the causative inference (C0047) that John felt a negative emotion toward Mary, and that this feeling was the cause of his desire that she become hurt. C0047 is displayed at the end.

---

```
C0041: (WANT #JOHN1 C0028)
(WANT #JOHN1 (*MFEEL* #MARY1 #NEGEMOTION
               #JOHN1))
ASET:
   C0043: (CAUSE # C0007)
   C0042: (TIME # C0006)
```

This motivational inference predicts that John wanted Mary to feel a negative emotion toward him. The evaluation function demotes this inference because this is negative with respect to John and people normally do not want things which are negative with

```
RECENCY: 18850
TRUTH: T, STRENGTH: 1.0
REASONS:
   C0007: (*PROPEL* #JOHN1 C0015 #JOHN1
               #MARY1)
ISEEN: NIL
```

respect to themselves.

---

```
C0044: (WANT #JOHN1 C0026)
(WANT #JOHN1 (WANT #MARY1 (POSCHANGE
               #MARY1 #PSTATE)))
ASET:
   C0046: (CAUSE # C0007)
   C0045: (TIME # C0006)
RECENCY: 18933
TRUTH: T, STRENGTH: 1.0
REASONS:
   C0007: (*PROPEL* #JOHN1 C0015 #JOHN1
               #MARY1)
ISEEN: NIL
```

This is the inference that John wanted Mary to want to get better. Although unlikely, it could nevertheless be a valid inference in the correct context, and should not be suppressed at time of inference. Such a context might be: "Mary was hyterical. John slapped her."

---

```
C0047: (*MFEEL* #JOHN1 #NEGEMOTION
               #MARY1)
ASET:
   C0049: (CAUSE # C0038)
   C0048: (TIME # C0006)
RECENCY: 21100
TRUTH: T, STRENGTH: 1.0
REASONS:
   C0038: (WANT #JOHN1 C0022)
ISEEN: NIL
```

Here is the causative inference which arose from the motivational inference that John wanted Mary to suffer a negative change (C0038).

## 5.7 ENABLING INFERENCES

**sample:** Mary said that she killed herself.
That's impossible.

**sample:** John kissed Mary.
John and Mary are near each other.

**sample:** John told Mary that Pete was at the store.
John must have known that Pete was at the store.

**sample:** Mary gave John the book.
Mary had the book just before she gave it.

**sample:** Pete went to Europe.
Where did he get that kind of money?

### 5.7.1 INTRINSIC AND EXTRINSIC ENABLING STATES

Every action which occurs in the world must have a fairly well-defined -- and usually predictable -- surrounding environment: namely one in which the action is possible! The environment in which an action occurs interacts with the action in two important ways. First, it can influence the *result* of an action after it has been initiated or completed. That is, the specific *effects* an action can have on the world are determined by the condition of the world before and during the action. (This includes the action's time, location, etc.) Second, the state of the world can either predetermine the nature of the action before it is undertaken, or even preclude its occurrence altogether. The first form of interaction concerning the course of an action, or the effects it has on the the world after it has been initiated or completed, is the realm of *resultative* inferences, and these must be inherently sensitive to the dimensionality of influence the environment can exert upon an ongoing or completed action. These inferences pay attention to the conceptual features of the entities involved, and states of the world which might influence the course or end effects of the action in a particular situation.

Resultative inferences, however, do not account for the second form of interaction between an action and its environment: circumstances which must obtain *before* the action can be successfully *initiated.* To account for and deal with these pre-conditions -- those states of the world which must be in effect for an action to be performed -- we must distinguish a separate class of inference. I will call those inferences which attempt to make explicit the probable enabling states which surround an action *event enabling inferences*, or just "enabling inferences". Fig. 5-28 schematizes the idea of enabling conditions.

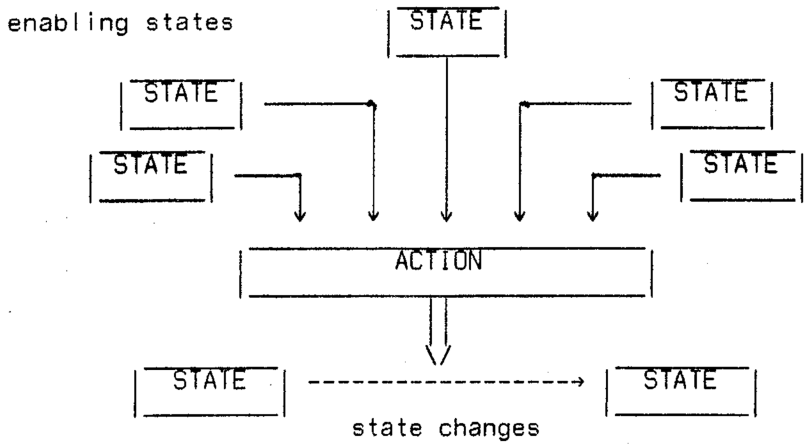



**Figure 5-28. Actions cause changes in states. States enable actions.**

An enabling inference therefore is a prediction about particular states (preconditions) of the world which must be (must have been) true in order for some action to be (have been) performed. We may further discriminate inferences in this class in a way which will be useful to the operation of the memory. This distinction concerns, in a sense, the degree to which the would-be actor can *influence* some precondition of an action. Those preconditions over which the would-be actor has some degree of potential control I will call "extrinsic". These are the actor-manipulable states of the world whose existence is requisite to the action.

In contrast, there are some preconditions over which the actor has no control, but rather which he *himself* must implicitly satisfy in order to perform the action. We can call these "intrinsic" preconditions.

Examples of very simple extrinsic preconditions are: "before a person P1 can give object X to P2, P1 must first have possession of X", or "in order for person P1 to kiss P2, P1 and P2 must have spatial proximity". Examples of intrinsic preconditions are "for a person P1 to perform an action, P1 must be alive", or "for a person P1 to play the piano, P1 must have healthy fingers". Matters of the actor's basic ability are not in general so well defined as these examples, since abilities can both develop and atrophy with time. However, at any given time, an actor may be thought to possess a basic set of abilities which we can view as *intrinsic* to him at that time (they are neither directly nor immediately alterable), but perhaps extrinsic over periods of time.

Why be concerned with enabling states of actions in the world? There are two reasons. The first lies at the heart of the theory of conceptual memory: part of what it means to "comprehend an action" is to be aware not only of how it occurred, what its causes were and what it caused, but *what must have been true in order for it to have occurred.* When a language user "comprehends" a language utterance which is underlied by a conceptual action, I will argue that he subconsciously expands the situation which must have surrounded the action when it occurred; he "imagines" the situation. This is the purpose of all inferences. If the memory can draw out many predictions about what must have surrounded the situation to which each utterance alludes, it stands a much better chance of discovering how one utterance relates to the next.

Enabling inferences are a particularly powerful source for this expansion about actions. As the examples we will examine shortly will bear out, it is the generation of enabling inferences which, perhaps more than inferences of any other single class, illustrates the need for a vast amount of computation in what I have called this unconscious substratum of cognition in which all conceptual inferencing occurs. By being aware of the preconditions for *every* action which we perceive indirectly through language, by exploring its implied intrinsic and extrinsic enabling conditions, we can discover very useful relations which would not otherwise be drawn out. In a sense then, enabling inferences must *put back* the richness surrounding an action which is lost in the process of communicating just the action by language: they "reconstitute the situation", elaborate it, and this can frequently lead to interesting discoveries -- intersections with other spheres in this inference space which might have been apparent if the situation had been experienced directly, but which are likely otherwise to be missed when experienced indirectly through language.

The second reason for concerning ourselves with enabling inferences is quite a bit more specific: many other processes in the memory which ultimately contribute to expansion in the inference space, particularly those concerned with making predictions (say, at each point in a story), rely heavily on an ability to predict preconditions for actions. A typical question posed by a predictive inference might be: "John may want to perform action A because of X, Y and Z. What might he do first in order to be able to perform A?" To answer such a question is to generate extrinsic enabling conditions for action, A, in A in the context in which it has been

predicted to be performed. The inference monitor has the ability to allow only inferences of a certain type to be generated from a structure. This is one application which requires this ability. Because of this, each type of conceptual inference is marked with a distinctive mnemonic by the inference molecule which generates it. Enabling inferences are are marked with the mnemonics EENB and IENB for extrinsic and intrinsic enablement, respectively.

### 5.7.2 ARGUMENTS FOR GENERATING MANY ENABLING INFERENCES SPONTANEOUSLY

It might be asked "Does a human language user really make very many enabling inferences on the average?" I argue emphatically "Yes": he makes a tremendous number, and many have the false appearance of being trivial! Insight into this is more readily gained by considering *contradictions* which a human language user will detect immediately, but which would not be readily detectable without this expansion of each action's underlying enabling conditions.

Consider the following five examples:

1. Mary said she killed herself.
2. John's dog wrote a concerto yesterday.
3. Billy's innertube had a hole in it. He inflated it and off he swam.
4. George, the N.Y. skid row bum, took a vacation to Europe.
5. Mary was in Seattle John was in Spokane. Mary bent over and kissed John.

It costs a human language user very little thought to uncover the absurdity of (1). Why is this? How do we so readily recognize that it is impossible, and impossible because it contains a contradiction? That is, suppose it were buried deeply in a story in which there were many facets and levels of understanding involved, and perhaps in which the reader even had some specific goals which were motivating and guiding his interpretation as he was reading the story. How does this leap out at him, almost disruptively? Although this utterance is not in itself very interesting, it points out an extremely interesting undercurrent of the understanding process: everything the reader reads *has to fit*, and he subconsciously verifies that it does. When it does, fine. But when it does not, this lower cognitive stratum where all sorts of conceptual inferences are being produced detects it and brings it to the attention of some higher level process. In this case, this happens because Mary performed an action at some time after she is alledged to be

dead. But it is a very direct intrinsic enabling inference that the actor of any sort of action be alive at the time of the action. The computer example at the end of this section illustrates this example.

The other examples (2-5 above) are equally absurd, or at least fishy for the same reason: at some level, an enabling inference has been drawn out and it clashes with information which has arisen from another line of inference initiated from some other aspect within the same utterance, or from some other source altogether. (4) causes concern because to take a vacation involves many ATRANS' of money from the vacationer to other people, and a precondition for ATRANSing an object is that the ATRANSer have possession of it first!

My operating assumption concerning enabling inferences is therefore:

> **Anytime a conceptual action is perceived in language, generate as many relevant intrinsic and extrinsic enabling inferences for the action as possible. They can lead to interesting discoveries, and they insure every time that the action adheres to the memory's assumptions about what is normal in the world.**

Section 8.1 contains another computer example illustrating how an enabling inference can be (as can all classes of inference) a vital link in the process of unambiguously establishing the identity of a *reference* to some person. In that example, if the enabling inference had not been spontaneously made, there would have been no way of distinguishing which of two people by the same name was being referred to. Furthermore, there would have been no good way for some "goal-directed" process to go back and discover this information. I take this to be one more source of confirmation for the hypothesis that human language users make copious quantities of subconscious enabling inferences in reaction to language stimuli.

# ENABLING INFERENCE COMPUTER EXAMPLE

In this example, we will see how an inferred intrinsic enabling state will play a key role in the discovery of a rather blatant contradiction: "Mary said she killed herself." Here, the enabling inference that Mary was alive at the time she said this is made from the main MTRANS action. But the substance of what she allegedly said leads by a resultative inference that she ceased to be alive *before* the time at which she spoke. Without *spontaneously* making both the intrinsic enabling and resultative inferences, the contradiction would be altogether missed by the memory, and presumably by a human language user!

---

```
(MARY SAID SHE KILLED HERSELF)

(((ACTOR (MARY1) <=> (*MTRANS*)
MOBJECT ((CON ((ACTOR (MARY1) <=>
(*DO*))) <= ((ACTOR (MARY1) <=>F
(*HEALTH* VAL (*ONE*)) <=>T (*HEALTH*
VAL (-10))) TIME (TIM01))) TIME
(TIM01)) FROM (*CP* PART (MARY1)) TO
(*CP* PART (*ONE*) SPEC (*U*)))
TIME (TIM02))

(TIM00 ((VAL T-0)))
(TIM01 ((BEFORE TIM02 X)))
(TIM02 ((BEFORE TIM00 X)))

((*MTRANS* (#MARY1) ((CAUSE ((*DO*
(#MARY1) (C0008))) ((STATECHANGE
(#MARY1) (#HEALTH) (C0010)
(#MINUSTEN)) (TIME _ (C0012))))
(TIME _ (C0012))) (C0017) (C0020))
(TIME _ (C0013)))

C0028
```

Here is the input utterance. This example assumes that the conceptual analyzer has identified "Mary", "she" and "herself" all to refer to the same person.

This is the partially integrated result in which the referencer has decided that "Mary" refers to #MARY1.

This is the resulting memory structure.

---

```
STARTING INFERENCE QUEUE:
((X 1.0 C0028))


ABOUT TO APPLY @MTRANS0 TO C0028
C0028: (*MTRANS* #MARY1
      (CAUSE (*DO* #MARY1 C0008)
             (STATECHANGE #MARY1 #HEALTH
               C0010 #MINUSTEN))
         C0017 C0020)
INFERRING: (TIME #MARY1 C0013)


ABOUT TO APPLY @MTRANS1 TO C0028
INFERRING: (*MLOC* C0026 C0031)
   ALSO GENERATING: (*TIME* C0034 C0013)

ABOUT TO APPLY @MTRANS3 TO C0028
INFERRING: (*MLOC* C0026 C0020)
   ALSO GENERATING: (TS C0036 C0013)
```

We suppress all but the main structure on the starting inference queue.

Here, the intrinsic enabling inference that Mary must have been alive at the time of her MTRANS action is being generated. This inference becomes structure C0030.

Other inferences about Mary believing what she said and whoever she said it to starting to know what she said are made by the MLOC inference molecule. These will lead to the memory's considering what she said.

```
        ......

ABOUT TO APPLY ⊗MLOC2 TO C0043
C0043: (*MLOC* (CAUSE (*DO* #MARY1
C0008) (STATECHANGE #MARY1
#HEALTH C0010 #MINUSTEN)) C0040)
INFERRING: C0026

        ......

ABOUT TO APPLY ⊗CAUSE1 TO C0026
C0026: (CAUSE (*DO* #MARY1 C0008)
(STATECHANGE #MARY1 #HEALTH C0010))
#MINUSTEN))
INFERRING: C0023

ABOUT TO APPLY ⊗CAUSE2 TO C0026
INFERRING: C0024

RECORDING CAUSAL RELATION:
        (C0023 . C0024)

        ......

ABOUT TO APPLY ⊗STATECHANGE1 TO C0024
C0024: (STATECHANGE #MARY1 #HEALTH
C0010 #MINUSTEN)
INFERRING: (TF #MARY1 C0012)
```

Here, MEMORY begins to consider the substance of what Mary said. The two resulting inferences are that she did something, and what she did caused a STATECHANGE in her health to -10.

The causal relation is recorded for causal chain expansion. This is not directly related to this example, but has been included for reference.

Here is the resultative inference from Mary's STATECHANGE that she ceased to exist at the time of the STATECHANGE.

---

```
CONTRADICTION DETECTED:
        C0046 CONTRADICTS C0030
        REASONS: (C0016)

C0046: (TF #MARY1 C0012)
C0030: (TIME #MARY1 C0013
C0016: (BEFORE C0012 C0013)

        ......
```

At this point, the inference evaluator detects a direct contradiction: Mary must have been existing after she ceased to exist (TIME and TF structures are contradictory). Notice that the reason the evaluator supplies for this contradiction is, C0016, the relation of the two times, C0012 and C0013. The contradiction is recorded, the two structures are removed from the inference queue, and inferencing proceeds.

# CHAPTER 6

## MORE CONCEPTUAL INFERENCES

This chapter is a logical continuation of the previous one, which was getting pretty long and, out of compassion for the reader, was artificially interrupted. This is just the "sixth-chapter-stretch"!

## 6.1 FUNCTION INFERENCES

**sample:** John wants the book.
John probably wants to read the book.

**sample:** Mary needs a hammer.
What is she building?

**sample:** John went to the grocery.
He probably wants to buy some food.

**sample:** Mary was furious at John.
She asked Bill for the baseball bat.
Mary might want to use the baseball bat to clobber John.

**sample:** A fly was annoying Bill.
He asked Pete to hand him the newspaper.
Bill probably wants to use the newspaper to swat the fly.

Everyone has at his disposal a wealth of information concerning the normal functions of physical objects, and this information is closely related to our algorithmic knowledge of the world. That is, given any common task, the average human language user will have a fairly thorough idea of what is necessary for the successful execution of that task. When the task is a physical one, the chances are high that some conceptual instruments will be involved in the algorithm which will accomplish the task. For example, if someone wants to open a bottle, he will perhaps want momentary control over a bottle opener; if someone wants to learn about computers, he is likely to want a book on the subject. In both cases, some object (bottle opener, book) is involved in some action lying on a path to the task solution.

A later section describes a class of inference which predicts a person's future actions based on his current wants. Inferences in this class have been termed *action prediction inference.* Action prediction inferences work forward from a person's WANT states to algorithms he is likely to engage to satisfy those wants. However, it is often desirable to go the other way: to be able

to infer the algorithms and goals themselves *from some small glimpse of the algorithm* in which he is engaged. For instance, given that someone wants a book, what might he be up to? What would having the book enable him to do? Predicting the use to which a person intends to put an object can lead to other important inferences concerning his goals. Predictions of this nature will be called *function inferences.* This section describes how the memory makes function inferences, and how they are useful to understanding.

### 6.1.1 TRIGGERING INFORMATION

What conceptual pattern should trigger a function inference? That is, how is the memory to detect when a person may be involved in some algorithm which requires an object? Clearly, the *intent to use the object* is one criterion for a function inference. That is, if a book falls off the shelf into John's lap, John *has* the book, but probably has no intentions of using the book in its normal manner. This suggests that function inferences should only be triggered when someone is known to *want* an object. But it is conceptually impossible to WANT just an object itself, as in (WANT JOHN BOOK). Conceptually, WANT only takes an entire conceptualization: some *action* or *state involving* the object is WANTed. To want an object commonly means to want to *have possession of* that object. Thus, John's wanting a book is represented conceptually as:

```
(WANT #JOHN1 (POSS C1 #JOHN1))
```

C1 being a token of a book.

Function inferences are therefore triggered by the pattern

```
(WANT P1 (POSS X P2))
```

where P1 may or may not be distinct from P2 ("John wants a book.", "John wants Mary to have a book.").

(It should be pointed out that the object in the WANT-POSS pattern which reaches the memory from the conceptual analyzer is, because of the representational formalism, guaranteed to be a real object. In other words, conceptualizations such as "John wants political power" have been mapped onto radically different conceptual structures *in the analyzer*, and hence are incapable of (incorrectly) triggering a function inference.)

### 6.1.2 NORMAL FUNCTION INFERENCES

What should the nature of the function inference which is triggered by patterns of this form in the proper causal environment be? Since it is the purpose of a function inference to relate a person's wanting possession of an object with some step in a probable algorithm, and since steps of an algorithm are actions by people,

> **the function inference should be in the form of an action by the person, involving the object he is believed to WANT.**

Furthermore, this one action (or several actions if more than one is applicable) should be *all* the function inference introduces. Indeed, consequences or implications of the inferred action will subsequently lead to the person's motivations for the action, the results of the action, what enabling states must have applied at the time, and so forth. But these are *other inferences and should not* be generated as part of the function inference because there are other inference classes designed for them later. Consider for example some object which is commonly thought of as food. What people usually do with food is ingest it for the purpose of nourishment. There are, however, clearly two parts to this: (1) *the action of ingesting* and (2) *the nourishment which results*. Since the memory needs for other purposes the general inference that when a person ingests food, he becomes nourished (John ate a steak => John became nourished), it would be redundant for a function inference to consist of more than the some mimimum action involving the object.

#### 6.1.2.1 THE NORMAL FUNCTIONS OF OBJECTS

Therefore, a function inference (triggered by the pattern (WANT P1 (POSS X P2)) ) produces a pattern (WANT P1 A), A being some action by P2 involving X. This means that the content of function inferences devolves on a knowledge of normal actions which represent the usual function to which physical objects are commonly put. Some examples are:

(1) The normal function of a book is that it be read.
(2) The normal function of money is that it be traded with someone in return for something else.
(3) The normal function of food is that it be ingested.
(4) The normal function of a car is that it be driven.
(5) The normal function of a telephone is that it be used to communicate.

and so on.

The examples in Fig. 6-1 illustrate how patterns which represent the normal function of objects and places are entered into the memory. The examples there are the LISP S-expressions which are read from the initialization file.

```
(NFCT #STORE
   ((DUALCAUSE ((*ATRANS* (∃ X (ISA _ #PERSON)) (∃ NIL (ISA _ #MONEY)) X #STORE))
               ((*ATRANS* #STORE (∃ NIL (UNSPECIFIED _)) #STORE X)))))

(NFCT #FOOD
   ((*INGEST* (∃ X (ISA _ #PERSON)) #FOOD (T NIL (ISA _ #MOUTH) (PART _ X))
                                          (T NIL (ISA _ #INSIDE) (PART _ X)))))
(NFCT #PRINTEDMATTER
   ((*MTRANS* (∃ X (ISA _ #PERSON)) (∂ NIL (ISA _ #CONCEPTS) (*MLOC* _ #PRINTEDMATTER))
              #PRINTEDMATTER (T NIL (ISA _ #CP) (PART _ X)))))

(NFCT #MONEY
   ((DUALCAUSE
     ((*ATRANS* (∃ X (ISA _ #PERSON)) #MONEY X (∃ Y (ISA _ #PERSON) (UNSPECIFIED _)
                                                                    )))
     ((*ATRANS* Y (∃ NIL (UNSPECIFIED _)) Y X)))))
```

**Figure 6-1. Four common NFCT patterns.**

#### 6.1.2.2 GENERATING A FUNCTION INFERENCE

Let us consider how the normal function of a book is stored and accessed for use in a function inference. For example, knowing

John wants a book.
(WANT #JOHN1 (POSS C1 #JOHN1))

(C1 being the token for some book), we want to know how the inference having content "John wants to transfer concepts in the book to his mind" (ie. John wants to read the book) arises.

The above pattern is intercepted by the WANT inference molecule which is called in the normal course of inferencing. This inference molecule calls a special procedure WHYWANT with the arguments #JOHN1 and C1. WHYWANT attempts to locate a normal function of C1 first by finding a memory structure of the form (NFCT C1 _). In this example, if C1 *itself* were known to have some unusual function not shared by other books (eg. it had been hollowed out by dope

smugglers to conceal their shipment), this function would be found and used. Of course, this will not in general happen for tokens of abstract concepts, because most tokens of concepts serve in the usual capacity as defined for the concept.

WHYWANT next scans up C1's ISA-set sequence searching for NFCTs until some abstract concept lying on this sequence is found with which a normal function is associated (Fig. 6-2). In this example, discovering (ISA C1 #BOOK) would cause the NFCT of #BOOK to be sought. In the current taxonomy of concepts in the memory, finding no NFCT of #BOOK, (ISA #BOOK #PRINTEDMATTER) would cause the NFCT of #PRINTEDMATTER to be sought and this time located. (In the case that no NFCT information is located in this manner, WHYWANT simply generates no function inference.) The located NFCT structure will be interpreted as a *pattern* to be instantiated, substituting C1 for occurrences of #PRINTEDMATTER, and #JOHN1 for the actor in the pattern. Other things will happen during the instantiation, but we must first understand the structure which is stored.

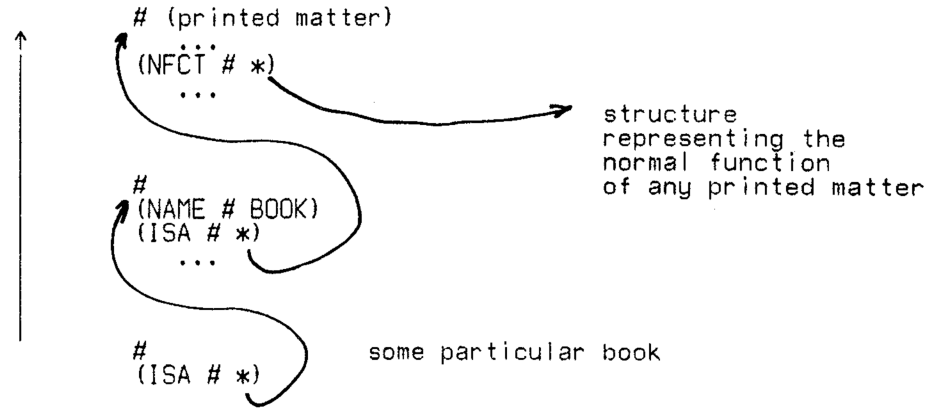



**Figure 6-2. Searching up ISA sets for an NFCT property.**

The normal function of #PRINTEDMATTER, is represented by the memory structure shown in Fig. 6-3. (There, A,B,C are for the purposes of the following discussion only). In English this is: "The normal function of something which is printed matter is for someone to transfer mental concepts located in the printed matter to his conscious processor (short term memory)." The same data structures used for all passive data are also used here. However, patterns such as these have some additional information associated with each substructure which indicates how

that substructure is to be interpreted during instantiation. This information also serves to denote what is template, requiring instantiation, and what is not template and should be taken literally. If this information were not available, the instantiator would wander off through the rest of memory thinking everything to be part of the template!

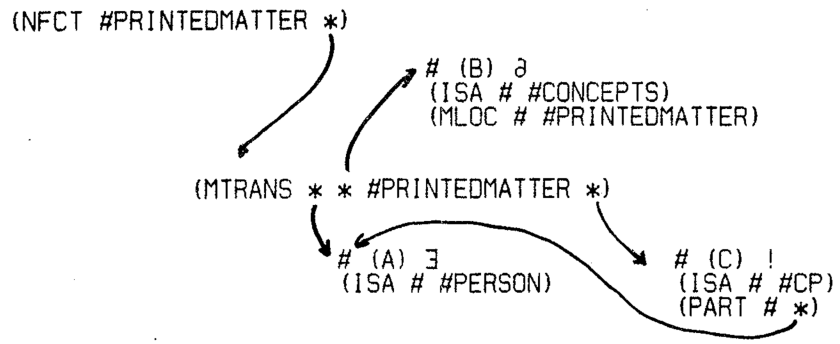



**Figure 6-3. The memory structure which stores the normal function of printed matter.**

Template information for some substructure signals one of three actions to be taken by the instantiator for that structure. Each of these three is illustrated in this example:

! -- *Locate a concept satisfying each element of the template component's occurrence set. If none is found, create one and use it.* The conscious processor of the person who reads printed matter is such an example. If the token representing John's CP cannot be found, it is only because MEMORY has never had occasion to reference it. Since it is perfectly normal to assume John has a CP, and that it is sufficiently specified by the two facts about it in Fig. 6-3, it should simply be created with no special attention paid to its creation.

∂ -- *Locate a concept satisfying each element of the template component's occurrence set. If none is found, create one and mark it as an unspecified concept.* The concepts which are located in the printed matter read by a person are an example of this. If the nature of the concepts contained in C1 is known, the token standing for them will be located and used. However, if the contents of C1 are not known, a token representing them will be created and marked as unspecified. Whenever the function inference which contains this unspecified concept is in

turn subjected to inference, the unspecified contents of the book will be detected. At that point some specifier molecule will either successfully guess what the book is about (unlikely), or the unspecified contents of the book will be recorded on the missing information list, !MISSINGINFO. This list is one of several sources of response after the memory finishes its reaction to an utterance.

∃ -- *Create a new concept which has the properties specified by the component's occurrence set.* This provides a way of forcing the creation of a new token, and generally corresponds to some existentially quantified variable which is unspecified at the time of the template instantiation. This form is not well illustrated in this example, because the actor of the MTRANS will be substituted as #JOHN1. The pattern for the normal function of money provides a better illustration. There, the person who receives the money in the trade is represented in this form, and is instantiated as some unspecified person whose identity may be filled in later by some specifier molecule.

Having located this NFCT, WHYWANT next determines where the actor in the template lies so that #JOHN1 may be substituted for it during the instantiation. This is a simple task since the normal function is assumed to be either a simple action or a causal form. In the case of a causal form, the actor is assumed to be the actor in the causing action. In Fig. 6-3, A is located as the actor.

Next, the pattern is instantiated, with the substitution list #JOHN1 for A, C1 for #PRINTEDMATTER, and following the three instantiation modes just described for the pattern's substructures. The instantiator takes care not to flow back from lower levels to higher levels in the graph during instantiation, since this would result in duplication of information and cycles. In this example, the result of instantiation is shown in Fig. 6-4 (assume the memory had no knowledge of *what* concepts the book, C1, contained).

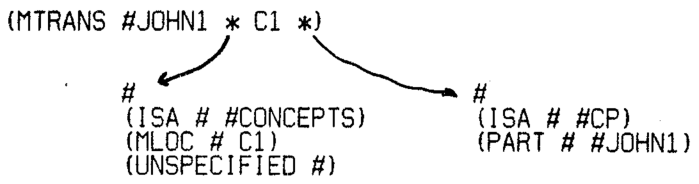



**Figure 6-4. The instantiated NFCT pattern about reading.**

(Notice that this is an inference process which can give rise to concepts and tokens whose specification is missing. This is essentially the same as a lack of specification at the linguistic level. Here, the origin of an UNSPECIFIED entity is internal to the memory: an inference implies the existence of an entity, but it cannot be specified. Such internally-generated unspecified entities are also noted on the list !MISSINGINFO and will subsequently pass through the specification process.)

WHYWANT then generates the inference: (WANT #JOHN1 X), where X is the above structure. In this example, it supplies as REASONS for the new inference four other memory structures: the original (WANT #JOHN1 (POSS #JOHN1 C1)), the two ISA relations which related C1 to #PRINTEDMATTER, and the NFCT structure used in the instantiation. Were the memory later to be asked "Why do you believe that John wants to read the book?", it could respond "Because John wants C1, C1 is a book, a book is printed matter, and the normal function of printed matter is that a person reads it."

In addition to the function inference itself, a CAUSE relation is generated to record that John's wanting to read C1 probably caused him to want to possess it. That is, the desire to use an object in its normal capacity causes a person to want to possess that object. Finally, any time information is copied from the original WANT to the new function inference. The result of the function inference for this example is shown in Fig. 6-5.

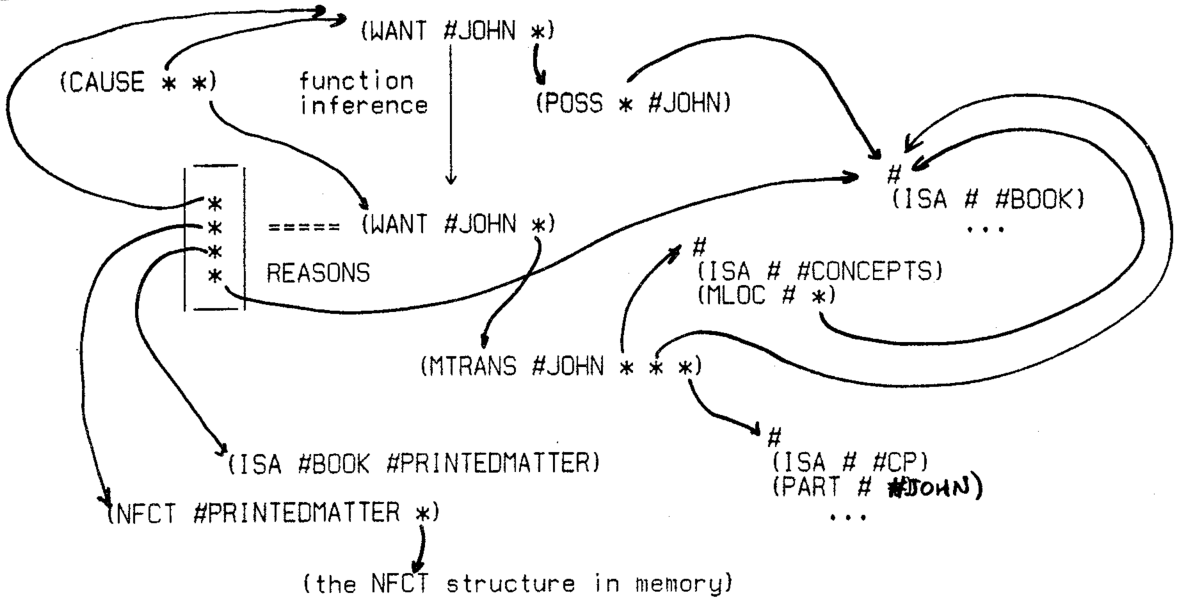



**Figure 6-5. The complete function inference and its surroundings.**

### 6.1.2.3 CONTRIBUTION TO UNDERSTANDING

What becomes of a function inference after it is generated? That is, why is it useful? In this example, further inferencing will occur from the function inference: knowing that John probably wants to MTRANS concepts in the book to his CP, MEMORY can apply motivational inferences which predict that he wants this action because of its probable consequences. One almost certain consequence is that he will begin to know the concepts he read. Hence, John must want to know about whatever topic the book discusses. But this in turn will lead to other predictions about what he might be up to, since knowing about some topic enables a person to engage in other algorithms involving that topic.

> **The general utility of a function inference is that, based on a knowledge of common functions of an object, it predicts some likely action which might be enabled by possession of that object. This will open up a new realm of motivational inferences.**

Actually, I have oversimplified the nature of the pattern which triggers function inferences. In fact, the memory reacts to three other patterns:

1. A person wanting an object to be at his location or at the location of his hand
2. A person wanting to OWN an object, rather than merely to possess it
3. A person *needing* an object, which is represented as P's not POSSessing X would cause (CANCAUSE) some negative change for X

In addition, there is a class of similar function inferences based on *locations* and the normal function of common locations. If a person wants to be at some location, he probably wants to perform an action which can be predicted from the normal function of the location (as in buying something from a store). (The computer examples which follow will illustrate this concerning John's wanting to be at a store.) As we will see in the next section, this notion of function inference is just one example of a more general class of inferences which involve the notion of action enablement in a way different from that of enabling inferences.

### 6.1.3 OVERRIDING NORMAL FUNCTION INFERENCES

Normal function inferences are clearly embedded deeply in our knowledge of normality in the world. It is equally clear that normality should provide no more than a backdrop which catches all types of reasoning which have terminated or failed because of a lack of *specific* world knowledge. We must therefore ask the question: by what mechanism does specific world knowledge override this knowledge of normality implied by the process of function inferencing? In particular, how does our *specific* knowledge of the individual who desires possession of an object, or of the object itself, influence the function inference process?

To illustrate, suppose we have a friend, John, who pours chocolate sundaes down womens' dresses. That is, that John does this is explicitly stored in a memory structure, accessible as a direct conceptual feature of both John and chocolate sundae. One day, we hear "John was with Mary yesterday. He asked Bill for a chocolate sundae." How is the inference that John wants to *eat* the chocolate sundae suppressed and the inference that he is likely to pour it down Mary's dress drawn out?

To answer this question, we must examine more closely what it means for a piece of world knowledge to be "specific". The specificity of a piece of knowledge, X, must be defined relative to some process which manipulates a class of knowledge of which X is an instance. What is

regarded as specific to one process might be regarded as general to another. For instance, before an appropriate inference can be made, the process which makes the inference must know what dimensions should influence it, and should be responsive to differences and deviations from the norm along those dimensions. This dimensionality can be quite narrow and well-defined, or it can be a very general, even ill-defined one. An example of specific dimensionality to which an inference process might be sensitive is the mass of object X, where, having discovered that object X was dropped on Pete's foot, the severity of the resulting state of affairs is up for inference. In contrast, an example of a very general, ill-defined dimensionality is the following: "is there some relation between object X and object Y which is not true for most other objects in the classes represented by X and Y?" The latter, more general type of dimensionality appears to characterize the tests which enable special-case knowledge to influence the substance of function inferences.

Specifically, there are two dimensions to which the function inference process is sensitive. The first solves problems of specific knowledge, the second makes function inferences sensitive to contextual instrumentality.

#### 6.1.3.1 OVERRIDING FUNCTION INFERENCES BY SPECIFIC KNOWLEDGE

The first dimension is this: before a function inference is generated, a test is made to determine whether there is any *special* relationship between the person who wants the object and the object. This question is framed as follows: is there a path (through conceptual propositions) between "John" and "chocolate sundae" other than those which include #JOHN's ISA set and #CHOCOLATESUNDAE's ISA set relations, (#PERSON and #FOOD, respectively)? If so, is this relation some structure which involves John as the actor (ie. an action or causal) and chocolate sundae as some sort of object, and is this proposition a timeless statement (that is, does it represent a fact or belief rather than one isolated event which may have occurred)? In the event that such a relation can be found, then it is a possibility that John may want the chocolate sundae for use in an action of this form.

By excluding conceptual paths between "John" and "chocolate sundae" *which pass through these concepts' supersets*, we automatically exclude "standard" relations between people and food, the most obvious of which is that people INGEST food. This guarantees that, should any other

paths be found, they will be specific to the two concepts, "John" and "chocolate sundae". This heuristic is illustrated abstractly in Fig. 6-6.

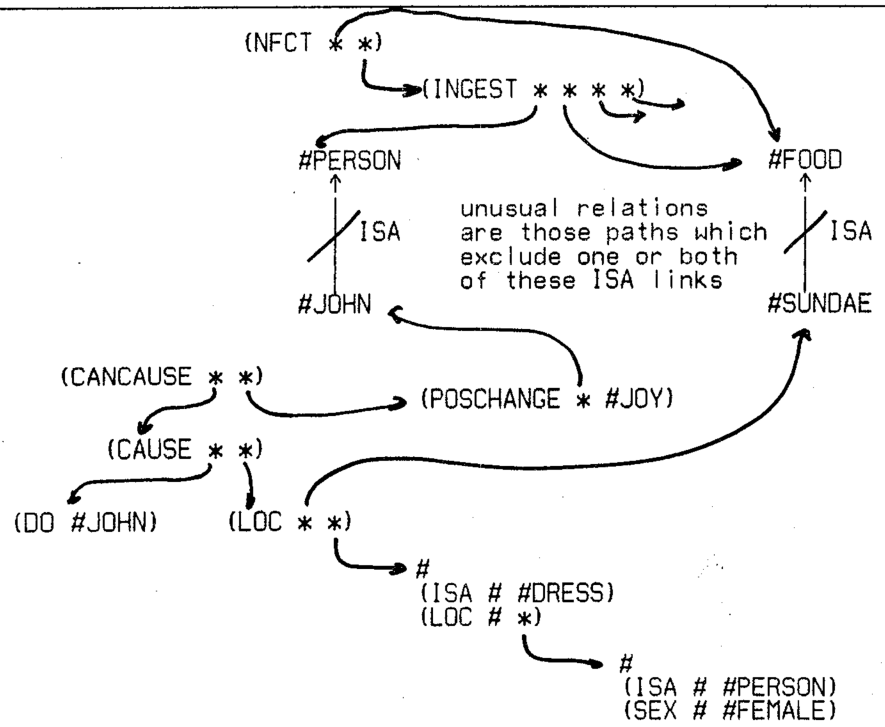



**Figure 6-6. Discovering unusual relationships between two tokens or concepts.**

### 6.1.3.2 THE INFLUENCE OF ACTION PREDICTIONS ON FUNCTION INFERENCES

The second "vague" dimension to which function inferences can be sensitive consists of tests aimed at discovering whether the person wanting the object is engaged (or has been *predicted* to engage in, via a predictive inference) some activity requiring an object having certain features. If the function inference process detects that the object the person is stated to want has the requisite features, it suppresses the normal function inference in deference to this contextual use of the object.

An example of this dimensionality is:

A fly was aggravating Bill.
Bill asked Pete to hand him the newspaper.

From the first line it is a predictive inference that a likely future action of Bill's is that he might be expected to propel some massive (from the fly's point of view) physical object toward the fly with the intention of killing the fly. At this point, Bill's wanting possession of a physical object (whose features are sufficiently specified to rule out things like the kitchen sink -- or a feather) can and should be associated with his current probable desires concerning worldly actions. This association is made by searching for structures of the form (WANT P A), where A is some action structure in which P is the actor and which involves an UNSPECIFIED object in some conceptual capacity. If such a structure can be found and if the requisite features of the unspecified object in the wanted action, A, are possessed by the object involved in the WANT-POSS triggering pattern, then A, with its unspecified object replaced by the object Bill wants to possess is a highly probable candidate for overriding the normal function inference. Here, this would mean that the inference "John wants to swat the fly with the newspaper" overrides "John wants to read the newspaper."

Both these "overriding" heuristics have a very attractive feature: since they involve searches through ever-changing conceptual structures, they are inherently sensitive to context. That is, *they allow the situation to override the norm.* Failures to override correspond to missing dimensions of testing before the normative inference is generated. No claim will be made here that the two dimensions of sensitivity to overriding situations just described are adequate, only that they are useful and can account for many interesting situations.

# FUNCTION INFERENCE COMPUTER EXAMPLE 1

This computer example will illustrate how the normal function of printed matter is accessed in order to relate the desire to possess a book with the desire to read it: one potential reason John may have given Mary the book is because he wanted her to read it.

---

```
JOHN GAVE MARY A BOOK

((ACTOR (JOHN1) <=> (*ATRANS*) TO
(MARY1) FROM (JOHN1) OBJECT (BOOK1
REF (*A*))) FOCUS ((ACTOR)) MODE
(NIL) TIME (TIM01))

(TIM00 ((VAL T-0)))
(TIM01 ((BEFORE TIM00 X)))


((*ATRANS* (#JOHN1) (C0020) (#JOHN1)
(#MARY1)) (TIME _ (C0022)))

C0025
```

This is the input sentence whose analyzed version, partially internalized version, and finished version (C0025) are shown.

---

```
STARTING INFERENCE QUEUE:
        ((X 1.0 C0025))


        ......

ABOUT TO APPLY @ATRANS2 TO C0025
C0025: (*ATRANS* #JOHN1 C0020
                 #JOHN1 #MARY1)

INFERRING: (*POSS* C0020 #MARY1)
   ALSO GENERATING: (TS C0029 C0022)

        ......
```

C0025, the structure representing this input is submitted to the inference mechanism. Other subpropositions have been suppressed for this example.

Numerous inferences are generated. Among them is the resultative inference that Mary begins possessing the book at time C0029, the time of John's *ATRANS* action.

---

```
APPLYING INF MOLECULE WANT TO C0041:
(WANT #JOHN1 (*POSS* C0020 #MARY1))

(ENTERING 1 WHYWANT)
(WHYWANT C0041 #JOHN1 #MARY1 C0020)


SEARCHING FOR NFCT OF C0020
SEARCHING FOR NFCT OF #BOOK
SEARCHING FOR NFCT OF #PRINTEDMATTER
NFCT FOUND: I0177


(*BREAK* . HELLO)


-----------------------------------
I0177: (NFCT #PRINTEDMATTER I0176)
(TEMPLATE T)
```

Other inferences arise. Among them is the motivational inference that John wanted Mary to possess the book. This inference is generated, subsequently to be detected by the function inference generator. Here, by turning on traces, we see the function inference mechanism in action.

C0020 represents the book which is in Mary's possession. The first step is to locate a NFCT for it or one of the abstract concepts Lying on its ISA superset. The NFCT for printed matter is located.

At this point, we will interrupt the program and take a look at this NFCT structure.

I0177 represents the knowledge that I0176 represents the normal function of printed matter.

```
RECENCY: NIL
TRUTH: NIL, STRENGTH: NIL
ISEEN: NIL

-----------------------------------
I0176: (*MTRANS* I0168 I0170
          #PRINTEDMATTER I0173)
(TEMPLATE T)

ASET:
   I0177: (NFCT #PRINTEDMATTER #)
RECENCY: NIL
TRUTH: NIL, STRENGTH: NIL
ISEEN: NIL
```

I0176, in English, says the following: a person *MTRANS*s concepts contained in something which is printed matter from that printed matter to his conscious processor. I0168 represents the person, I0170 the concepts, I0173 the person's conscious processor.

```
-----------------------------------
I0168: NIL
(TEMPLATE ∃)

ASET:
   I0176: (*MTRANS* # I0170
             #PRINTEDMATTER I0173)
   I0175: (PART I0173 #)
   I0169: (ISA # #PERSON)
RECENCY: NIL
```

Here is the actor in the structure. Mary will assume this role in the finished inference.

```
-----------------------------------
I0170: NIL
(TEMPLATE ∂)

ASET:
   I0176: (*MTRANS* I0168 #
             #PRINTEDMATTER I0173)
   I0172: (*MLOC* # #PRINTEDMATTER)
   I0171: (ISA # #CONCEPTS)
RECENCY: NIL
```

Here is the template representing the concepts contained in the printed matter. Notice that its template marker is ∂, indicating that if such a structure is not found for the book, C0020, the template should be instantiated as an unspecified token, ie. that the concepts in C0020 are unknown.

```
-----------------------------------
I0173: NIL
(TEMPLATE !)

ASET:
   I0176: (*MTRANS* I0168 I0170
             #PRINTEDMATTER #)
   I0175: (PART # I0168)
   I0174: (ISA # #CP)
RECENCY: NIL
```

Here is the template which represents the person's conscious processor. Notice that its template is of type !, indicating that if such a token is not found, it should be created with no attention being paid.

---

```
*PROCEED
SEARCHING FOR ACTOR IN I0176
ACTOR ESTABLISHED: I0168
```

At this point, we proceed with the preparation for instantiation. MEMORY now locates the actor in the NFCT template in order to substitute occurrences of the actor in the template by the actor at hand, namely Mary. I0168 is located.

---

```
(ENTERING 1 FINDUNIT)
(FINDUNIT (NIL (*MLOC* _ C0020)
   (ISA _ #CONCEPTS)))
(LEAVING 1 FINDUNIT) NIL
(ENTERING 1 MAKEUNIT)
```

The instantiation is underway. Here MEMORY is attempting to establish the identity of the concepts in the book. Finding no such token, MEMORY creates a token, C0047, to stand for them and marks it as an unspecified structure.

```
(MAKEUNIT (NIL (*MLOC* _ C0020)
   (ISA _ #CONCEPTS) (UNSPECIFIED _)))
(LEAVING 1 MAKEUNIT) C0047
```

Later on, this unspecified token may give rise to a question of the nature: "What is the book about?".

```
(ENTERING 1 FINDUNIT)
(FINDUNIT (NIL (PART _ #MARY1)
   (ISA _ #CP)))
(LEAVING 1 FINDUNIT) NIL
(ENTERING 1 MAKEUNIT)
(MAKEUNIT (NIL (PART _ #MARY1)
   (ISA _ #CP)))
(LEAVING 1 MAKEUNIT) C0051
```

Here MEMORY is instantiating the template which represents the person's conscious processor. Finding no token representing Mary's, MEMORY simply creates one, C0051.

```
(ENTERING 1 MAKEUNIT)
(MAKEUNIT ((*MTRANS* #MARY1 C0047
   C0020 C0051)))
(LEAVING 1 MAKEUNIT) C0054
```

Finally, MEMORY creates the structure representing Mary's reading C0020.

```
COMPLETED INFERENCE: C0055

(*BREAK* . HELLO)
```

The finished inference is generated by asserting that John probably WANTs this reading action, and that this WANT CAUSEd him to WANT Mary to have possession of the book. We will now have a look at the final structure which is the inference.

---

```
C0055: (WANT #JOHN1 C0054)

ASET:
   C0057: (CAUSE # C0041)
   C0056: (TIME # C0022)
RECENCY: 29233
TRUTH: T, STRENGTH: 0.98000000
REASONS:
   C0041: (WANT #JOHN1 C0029)
   C0021: (ISA C0020 #BOOK)
   I0068: (ISA #BOOK #PRINTEDMATTER)
   I0177: (NFCT #PRINTEDMATTER
               I0176)
ISEEN: NIL
```

C0055 is the finished function inference.

Notice that the causal relation between John's wanting Mary to read C0020 and his desire that she possess it has been generated.

Notice the reasons indicating why MEMORY believes this structure: John wanted Mary to have possession of C0020, C0020 is a #BOOK, a #BOOK is #PRINTEDMATTER, and the normal function of printed matter is that it be read.

```
C0054: (*MTRANS* #MARY1 C0047 C0020
               C0051)

ASET:
   C0055: (WANT #JOHN1 #)
RECENCY: 29233
TRUTH: NIL, STRENGTH: NIL
ISEEN: NIL
```

```
C0047: NIL

ASET:
   C0054: (*MTRANS* #MARY1 # C0020
               C0051)
   C0050: (UNSPECIFIED #)
   C0049: (ISA # #CONCEPTS)
   C0048: (*MLOC* # C0020)
RECENCY: 29233
```

This token represents the unknown contents of the book, C0020.

```
C0020: NIL

ASET:
   C0054: (*MTRANS* #MARY1 C0047
                # C0051)
   C0048: (*MLOC* C0047 #)
   C0029: (*POSS* # #MARY1)
   C0027: (*POSS* # #JOHN1)
   C0025: (*ATRANS* #JOHN1 # #JOHN1
                #MARY1)
   C0021: (ISA # #BOOK)
RECENCY: 29233
```

This is the book. Viewed in this form, not all of the information is visable. In particular, there are modifying TIME, TS and TF relations on the structures C0054, C0048, C0029, C0027 and C0025 which are not shown here, but which MEMORY is sensitive to in inferencing, answering questions, and so forth.

```
C0051: NIL

ASET:
   C0054: (*MTRANS* #MARY1 C0047
                C0020 #)
   C0053: (ISA # #CP)
   C0052: (PART # #MARY1)
RECENCY: 29233
```

This is Mary's conscious processor.

```
*PROCEED
(LEAVING 1 WHYWANT) C0055
```

Control is returned to the WANT inference molecule with function inference C0055.

## FUNCTION INFERENCE COMPUTER EXAMPLE 2

In this example, the prediction will be made that the reason Rita went to the store was so that she would be LOCated there, and that the reason she wanted to be LOCated there was that it would enable her to perform an action commonly associated with being LOCated in a store, namely a buying action.

```
RITA WENT TO THE STORE

((ACTOR (RITA) <=> (*PTRANS*) OBJECT
(RITA) FROM (*ONE*) TO (STORE REF
(*A*))) TIME (TIM01))

(TIM00 ((VAL T-0)))
(TIM01 ((BEFORE TIM00 X)))


((*PTRANS* (#RITA1) (#RITA1)
(C0013) (C0014))
(TIME _ (C0016)))

C0019
```

MEMORY accepts the sentence from the analyzer. Its analyzed form, partially internalized form, and final structure pointer (C0019) are shown.

C0013 stands for the (unspecified) location Rita set out from.

```
STARTING INFERENCE QUEUE:
((X 1.0 C0019))

        ......

ABOUT TO APPLY *PTRANS2 TO C0019
C0019: (*PTRANS* #RITA1 #RITA1
                 C0013 C0014)
INFERRING: (*LOC* #RITA1 C0014)
   ALSO GENERATING: (TS C0023 C0016)

        ......
```

Inferencing begins, again with other subpropositions suppressed.

At some point, MEMORY generates the inference that Rita arrives at the store. A motivational inference scan (which is not shown by tracing here) will infer that she went to the store because she wanted to be there.

---

```
APPLYING INF MOLECULE WANT TO C0027:
(WANT #RITA1 (*LOC* #RITA1 C0014))


SEARCHING FOR NFCT OF C0014
SEARCHING FOR NFCT OF #STORE
NFCT FOUND: I0151

(*BREAK* . HELLO)

-----------------------------------
I0151: (NFCT #STORE I0150)

(NFCT #STORE (DUALCAUSE
(*ATRANS* I0142 I0144 I0142 #STORE)
(*ATRANS* #STORE I0147 #STORE I0142)))

(TEMPLATE T)
RECENCY: NIL
TRUTH: NIL, STRENGTH: NIL
ISEEN: NIL
```

The WANT inference molecule eventually intercepts the pattern of Rita's wanting herself to be located at a store, and undertakes a function inference.

MEMORY finds the NFCT of a store. We interrupt the process to have a look at this NFCT structure.

In English: "the normal function of some location which is a store is that a person gives it money in exchange for something. In this structure, I0142 stands for the actor, I0144 stands for a token of money which is to be created, and I0147 stands for some unspecified object. It may of course be possible to specify this object, but this will be intercepted later by some specifier molecule (example: "John needed some bread. He went to the store.").

---

```
*PROCEED
COMPLETED INFERENCE: C0040

(*BREAK* . HELLO)

-----------------------------------
C0040: (WANT #RITA1 C0039)

(WANT #RITA1 (DUALCAUSE
(*ATRANS* #RITA1 C0033 #RITA1 C0014)
(*ATRANS* C0014 C0036 C0014 #RITA1)))

ASET:
   C0041: (TIME # C0016)
RECENCY: 10800
TRUTH: T, STRENGTH: 0.98000000
REASONS:
   C0027: (WANT #RITA1 C0023)
   C0015: (ISA C0014 #STORE)
   I0151: (NFCT #STORE I0150)
ISEEN: NIL


-----------------------------------
C0033: NIL
```

Having seen the NFCT structure, we allow MEMORY to proceed. The completed function inference is C0040. We again break to display this new inference structure.

C0033 is some money, C0014 is the store Rita went to, C0036 is some unknown object.

Here are the reasons MEMORY believes this structure to be true.

This is the money that Rita will probably *PTRANS to the store.

```
ASET:
   C0035: (*ATRANS* #RITA1 # #RITA1
                C0014)
   C0034: (ISA # #MONEY)
RECENCY: 10800

-----------------------------------
C0036: NIL

ASET:
   C0038: (*ATRANS* C0014 # C0014
                #RITA1)
   C0037: (UNSPECIFIED #)
RECENCY: 10800
```

Here is the as-yet unspecified object which Rita probably wants the store to ATRANS her.

```
*PROCEED

        ......

APPLYING INF MOLECULE WANT TO C0040:
(WANT #RITA1 (DUALCAUSE
(*ATRANS* #RITA1 C0033 #RITA1 C0014)
(*ATRANS* C0014 C0036 C0014 #RITA1)))

        ......
```

We again return control to the program. Eventually, this function inference itself will be subjected to the inference mechanism. The WANT DUALCAUSE will lead to predictions that Rita wants the consequences of this DUALCAUSE, Among them will be that she wants to possess this unspecified object and that she wants to cease to POSS the money involved. The latter is intercepted by the evaluation function as contradicting MEMORY's knowledge of normality. For the former, if MEMORY already knew that Rita wanted some object, the evaluation function would detect this pattern as matching that pattern, and infer that the unspecified object in this pattern to be the object she was known to want. Otherwise, some specifier molecule will at some point attempt to predict more about the nature of the object from features of C0014, the particular store Rita visited.

## 6.2 ENABLEMENT PREDICTION INFERENCES

**sample:** John asked Mary where Fred was.
John wanted to give Fred some keys.

**sample:** Andy blew fervidly on the hot meat.

**sample:** Dick looked in his recipe book to find out how to make a roux.

**sample:** I sure hope it is sunny Saturday.
We're going on a picnic.

**sample:** Mary put on her glasses.
Mary probably wants to look at something.

**sample:** John walked over to the hammer.
John might want to pick the hammer up.

Function inferences actually represent an interesting subclass of a far more general class of inferences. We may generalize such observations as "the possession of an object is generally desired so that that object may be put to its normal use" and "the desire to be located at a certain place is probably instilled by the desire to perform an action normally associated with that place" to the more general observation that "states in the world are frequently desired because of the actions they will enable." More concisely,

> **If some state, S, is an important extrinsic enabling state for some common action, A, and if some person, P, is said or inferred to desire that S exist, then is possible and useful to predict that P might also desire to perform A, (and that this desire instilled the desire that state S exist).**

I will term inferences which accomplish this type of task *enablement prediction inferences.*

For the purposes of generating enabling inferences, every action must have associated with it in the memory's inference molecules certain intrinsic and extrinsic enabling conditions, and these enabling conditions are spontaneously generated whenever their associated action arises. As we saw, these inferences, especially the extrinsic ones, are vitally important to the process of expansion in inference space. They can lead to extremely useful lines of inference, and can uncover apparent contradictions.

The goals of an enablement prediction inference lie in something of an inverse relation with the goals of an enabling inference: whereas an enabling inference works from an action back to states of the world which must have been (probably were) true for the action to have occurred, an enablement prediction inference works forward from a state S, which is detected in a pattern of the form (WANT P S), inferring a structure (WANT P A), where a is some action which state S commonly enables (Fig. 6-7). Of course, not all states are commonly associated as an enabling condition for any action action in particular, but for those which are, it is desirable that their relation be accessible in this "reverse mode".

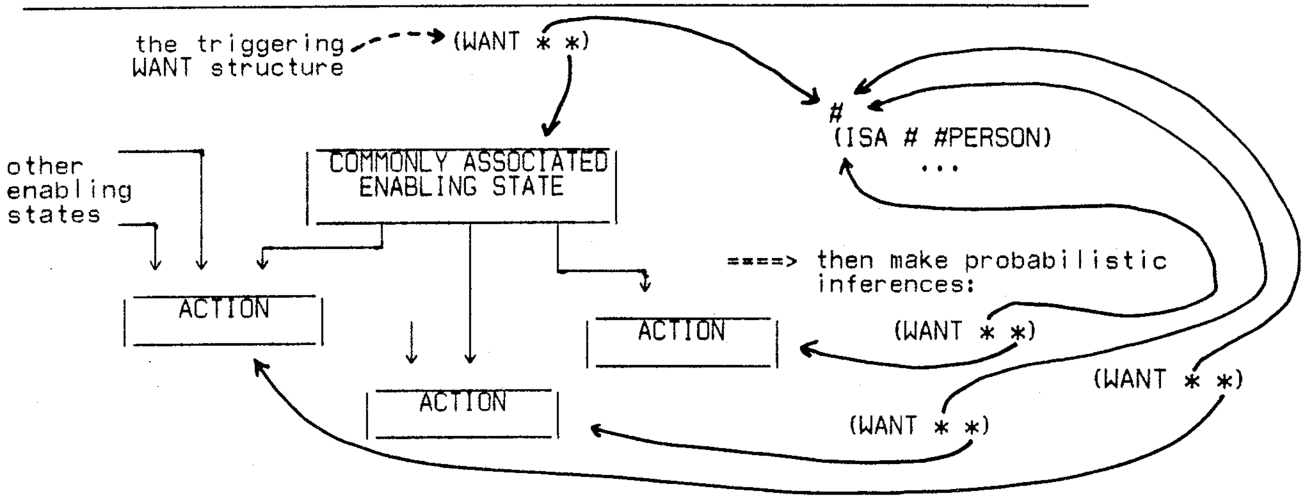



**Figure 6-7. The process of enablement prediction.**

For enabling relationships of any complexity there is no convenient "clean" or explicit data structure which links common enabling states to the actions which they are commonly thought of as enabling. The variability of the objects involved requires that this relationship be far more complex than a simple data structure can conveniently capture. For example, it would be conceptually pleasing to represent the enablement relation "In order for P1 to give P2 (physically), P1 and P2 must have approximately the same location" by a *data* structure such as:

(ENABLES (LOC X Y) (PTRANS X Z X Y)).

But upon closer examination, there are simply too many dimensions which must be tested in particular situations to make this realistic (time aspects, unusual instrumentalities, tolerances on the closeness of the LOC, dependencies on the conceptual features of X, Y and Z, etc.).

Therefore, for any state, S, which is commonly thought of as the primary enabling condition for some action, A, there exists an inference atom which can relate S to A: when applied to structure S, one or more inference atoms in the inference molecule will generate an action structure A as an "enabled action" inference. A simple enablement prediction process will then generate (WANT ACTOR A) predictions. This process and the form of the inference are shown in Fig. 6-7.

### 6.2.1 WHY AND HOW

How does an enablement prediction inference fit into the process of understanding? Consider the second sample above: "Andy blew fervidly on the hot meat." One very predictable resultative inference from such an action is that a hot object across which air is propelled decreases in temperature: (NEGCHANGE X #TEMP). A simple further resulative inference is that a result of a NEGCHANGE tends to yield a low(er) value along the scale on which the change occurs. The memory may therefore conjecture (via a motivational inference), that Andy WANTed these predictable results of his action. These two (of perhaps many) motivational inferences will have the form shown in Fig. 6-8.

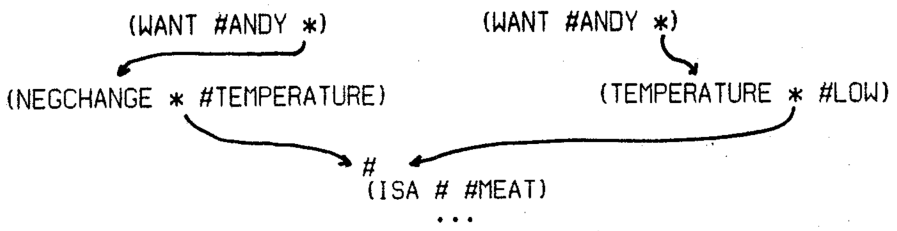



**Figure 6-8. Why Andy might be blowing on the meat.**

At that point, the structure (WANT #ANDY (TEMPERATURE C8324 #LOW)), C8324 being the meat, will be detected (in the WANT inference molecule) as a potential enablement prediction inference pattern. At that point, the WANT inference molecule defers control to the enablement prediction process, which then directs the inference monitor to generate inferences of type EA (enabling action) from the TEMPERATURE structure of Fig. 6-8. This amounts to applying the TEMPERATURE inference molecule to this structure, and filtering out all inferences but those of type EA. Because (ISA C8324 #MEAT) and (ISA #MEAT #FOOD), in this instance the EA inference set returned will consist of the single action (INGEST #ANDY C8324). Notice the need for the TEMPERATURE inference molecule's EA sensitivity to the conceptual features of the objects involved: if C8324 had been a piece of molten glass, entirely different EA inferences would have resulted. Any special information the TEMPERATURE inference molecule applied in making its EA inference are returned as the REASONS property of the new inference structure. In this case, the REASONS would be the two ISA properties of C8324 which relate it to #FOOD.

Having thus generated a set of actions commonly associated with this enabling state (which in this not-atypical case consists of just one) the EPI process creates a (WANT #ANDY (INGEST #ANDY C8324 X Y)), where X and Y are the token's for Andy's mouth and stomach, respectively. It then *transfers* the REASONS property supplied by the TEMPERATURE inference molecule to this new WANT structure, and appends the original (WANT #ANDY (TEMPERATURE C8324 #LOW)) structure to it as another reason. In English, the reasons for the inference end up as: "because Andy wanted the temperature of C8324 to be low, C8324 is meat, and meat is a form of food." The substance of the inference is "The reason a person wants food to be cool is so that he can eat it." Having thus generated WANT-INGEST structure, and supplied it with REASONS, the EPI monitor finally makes explicit the probable causal dependence between the WANT-TEMPERATURE structure, S1, and the WANT-INGEST structure, S2: (CAUSE S2 S1). That is, "Andy's desire to eat C8324 probably caused his desire that C8324 be cool."

This new structure can then lead to other inferences. The important step in this example was *drawing out the idea of eating* from the desire of some temperature state of some object.

### 6.2.2 EXAMPLES OF UTILITY

The utility of an enablement prediction inference to the other samples is similar. In the first sample ("John asked Mary where Fred was. John wanted to give Fred some keys."), the first utterance (whose underlying meaning is illustrated in Fig. 6-9) is that John wants Mary to perform an MTRANS action.

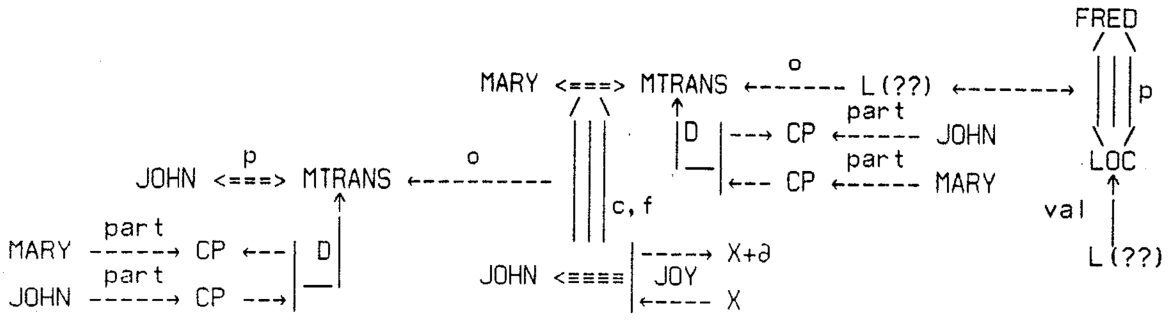



**Figure 6-9. John asked Mary where Fred was.**

In order to discover why John desires this action, the process of generating probable motivational inferences from the structure is undertaken. This process consists of finding the probable results of Mary's MTRANS action if she were to perform it, then conjecturing that John might desire her action because of one or more of the results it could produce.

An immediate resultative inference from such an MTRANS is that John would then know Fred's location. One highly likely motivational inference which follows, then, is that John WANTs to know Fred's location (Fig. 6-10).

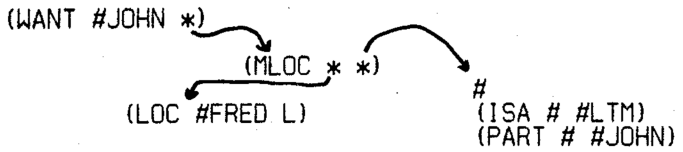



**Figure 6-10.**

But the need to know the location of an object is a common enabling condition for only a few common actions, foremost among them being a PTRANSing action, on the part of the knower, to that location. That is, the question "Why does John want to know Fred's whereabouts?" is reasonably answered by "Perhaps because John wants to go to there." Therefore, at that point the enablement prediction inference that John might want to PTRANS himself to Fred's location, L, is generated (among others -- remember, I'm willing to "waste" some computation in inference space) by the MLOC inference molecule, having been applied to the MLOC substructure in Fig. 6-10. The resulting probabilistic inference is shown in Fig. 6-11. (Note there that it will be a subsequent task for a specifier molecule to specify the D-FROM of John's PTRANS as the location where he and Mary currently are.)

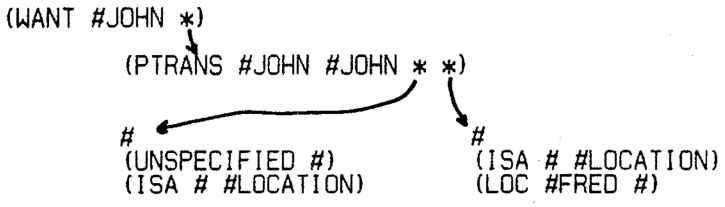



**Figure 6-11.**

But now a similar process may begin anew on the question "Why would John want to be at Fred's location?". Here, although to be at a location can be an enabling condition for a *vast* number of actions, the number of them most frequently associated with the condition is small, and the conceptual features of the location and specific knowledge about the person both can help narrow the set of probable actions. For example, since the location in this example is where another person is, two very commonly associated actions are (a) to communicate (some sort of MTRANS via a SPEAK) or to perform some sort of physical action which involves being at that location (very likely a PTRANS, perhaps a PROPEL, etc.). In other words, these are associations *human* language users commonly make, even though they may be proven incorrect later on. We would like the enablement prediction process at least to make a prediction about the *forms* John's subsequent actions are likely to take at that point, even if it would be ludicrous to try to predict any details of his probable action.

We have come far enough in the analysis of the first line of this sample ("John asked Mary where Fred was. John wanted to give Fred some keys.") to illustrate what happens when the second line arrives: the second utterance explicitly communicates a desire on John's part to perform the PTRANS of the keys to Fred. The hope is that this explicit information *matches* one of the predictions of form the memory was able to make from the first sentence. Although it might appear that I have set up just the right inferences at the right times, this is not what in fact happens in the memory, because there are in general large numbers of inferences which simply don't come to fruition. The claim is that this is necessary in order to discover the few inferences which *do* connect up in inference space.

In the third sample, by enablement prediction, we again can go from Dick's desire for some knowledge to an action that knowledge can enable him to perform (making a roux). As we might expect, the *desire for knowledge* is a very common pattern for triggering enablement prediction inferences: Dick wants to know how to do X because he wants to do X! In the fourth sample, since one "commonly held" enabling condition for some sort of outdoor activity is that it occur during nice weather, by making the prediction that the speaker has some outdoor plans for Saturday, the memory can establish contact with the information communicated by the second line.

It should be clear that it would be undesirable to spend too much effort making enablement

prediction inferences for states which could enable *many* actions. On the other hand it is desirable to be able to make at least predictions about the *nature* of the action someone who desires some state might be expected to perform, because this increases the likelihood of points of contact in inference space.

## 6.3 MISSING ENABLEMENT INFERENCES

**sample:** Mary couldn't see the horses finish.
She cursed the man in front of her.

**sample:** Ellen couldn't read the sign.
She walked over to the light switch.

**sample:** Ellen couldn't read the sign.
She walked closer to it.

As illustrated in the enablement inference section, it is very useful, for each action which is believed to have occurred, to generate predictions concerning the enabling states which must have surrounded the action in order for it to have occurred. However, what can we infer from an action which was attempted, but which failed? In conceptual form, this situation corresponds to a (CANNOT <action>) pattern with which a time is associated, such as "Mary couldn't see the horses finish."

The way I have chosen to view actions and enablement makes the anwser more or less immediate: when an action cannot occur, it is probably because some enabling state is not satisfied. If the memory can make predictions about what the missing state might be, other predictions about what the inhibited actor might do can result from those predictions. When we hear the first sample, we immediately infer that something was not correct for Mary's act of looking at something: either she was sightless, she wasn't in the vicinity, something was in her way, or the horses' finishing didn't exist to be seen in the first place. By making these predictions, the memory stands an improved chance for discovering relations with other information: in the first sample, the second line confirms the prediction about blockage of sight -- something was in front of her. Since another inference that Mary WANTed to be able to perform the action, and since it was another person who caused her not to be able to perform it, the resultative inference can then arise that Mary might feel a negative emotion toward the person because of this. Her cursing also leads to this conclusion, and this knits the first and second lines together at a critical point which makes their underlying causal relation explicit.

Similar remarks apply to the second and third samples. Notice also how this sort of inference can lead to action predictions which help clear up references. In the third sample, one of the missing enablement inferences from the CANNOT MTRANS pattern is that Ellen might not be close enough. This leads to one prediction that she might want to be at a location closer to the sign. From this, the action prediction arises that she might PTRANS herself closer to the sign. When the second line come in, if the pattern can be matched to the prediction, the matching could yield the identification of "it" as the sign Ellen is trying to read.

## 6.4 INTERVENTION INFERENCES

**sample:** Bill saw Mary hitting John with a baseball bat.
Bill took the bat away from Mary.

**sample:** Baby Billy was running into the street.
Mary ran after him.

All actions in the world have intrinsic and extrinsic enabling conditions (states) which must be satisfied for the action to occur (and continue in cases of protracted actions). Furthermore, certain of the extrinsic enabling states are distinguished in that they are commonly thought of as the most vital ones to the performance of the action which they enable. From these observations, I have made three important inference classifications: enabling inferences, enablement-prediction inferences, and missing enablement inferences.

But there is another important facet of this action/enabling state relationship The observation is this: that, *by removing an essential enabling state, it is possible to prevent or curtail an action which it enables.* Since actors have some degree of control over extrinsic enabling conditions -- that is, they can either bring them about, remove them, or cause them not to come about in the first place -- removal of an enabling state is one potential method through which an actor can influence *other* actions around him -- in particular, actions which he believes are likely to cause undesirable states. This phenomenon is of considerable interest to our goal of knitting together an actor's actions and his motivations for performing them.

We are therefore interested in predicting situations in which an actor may desire that an extrinsic enabling state not be allowed to exist, or cease to exist. When we can predict such a desire, let us call it an *intervention inference.* An intervention inference is one source of our ability to make certain predictions about what actors might do in the future, based on their

awareness of their surroundings (Fig. 6-12). The questions are twofold: what kind of actor awareness justifies the generation of an intervention inference, and what is the substance of the inference which is generated?

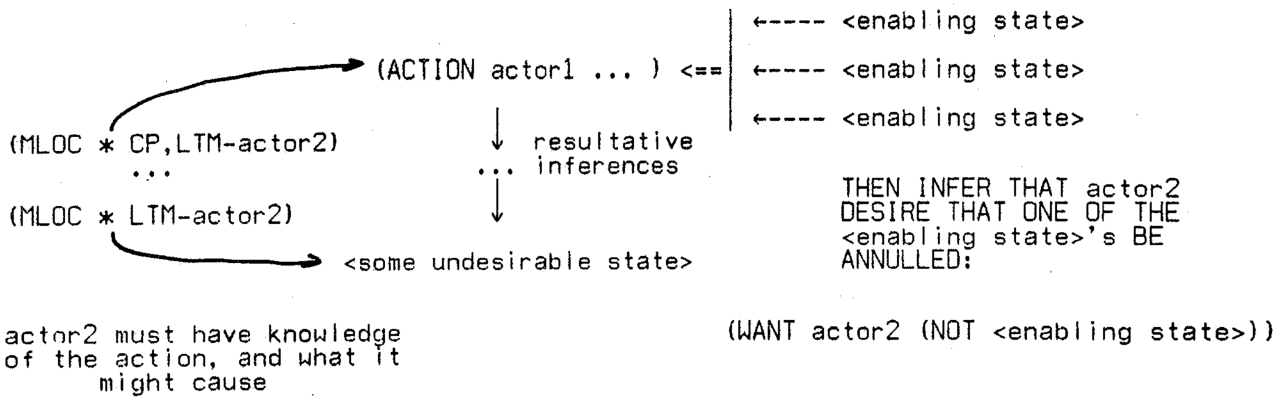



**Figure 6-12. The circumstances surrounding an intervention inference.**

Consider the first sample above. We would like the memory to have the capability, having heard the first line, to predict that Bill might do something to intervene. If this prediction can be made, it will be confirmed by the second line, representing a point of contact in inference space between the two utterances. If the prediction is made, but never confirmed, it can later be of use in generating the question "Why didn't Bill try to stop Mary?"

### 6.4.1 THE TRIGGERING PATTERN

The crucial feature of the pattern in the first sample seems to be that something bad is happening to John, and furthermore, that Bill is aware of this. In the memory, "something bad is happening to P" is a very simple pattern: (NEGCHANGE P SCALE), or (BIGNEGCHANGE P SCALE). Furthermore, this pattern in general lies on some causal sequence in which an action is the immediate cause of the negative change. In this example, MEMORY makes the simple resultative inference from Mary's hitting action with a baseball bat that John is suffering a negative change on his physical state scale. (There are many other interesting lines of inference -- for example that Mary probably MFEELS anger toward John -- which I will ignore here.)

But to predict that Bill will take steps to intervene, we must first have reason to believe that Bill is aware of both this NEGCHANGE and its causing action. In general, inferring who knows what in a story is often difficult because it is often bound up with assumptions and conventions about the structure of stories. Frequently, a sizeable chunk of a story will unfold before it becomes apparent that another person was around all along. (This is either stated explicitly, or can be inferred from his introduction as an actor and the nature of the action he performs), and that he was aware of everything. Because of this, there must be some means for retracing the story, inferring all the newly-inferable awarenesses of this new actor. The second sample is a very simple example of this: it is implicit in the structure of the story that Mary was around and aware of Baby Johnny's action, and it is vital to our understanding of her action that *we know* she was aware. I will not stray into story heuristics here, but only point out that inferring who knows what in a story is not always an easy task.

There is no such problem in the first sample: in a situation such as this the memory will infer that Bill knows of John's negative change by Mary's action, because he knows of her action and he, as well as the memory, can be predicted to know what this sort of action commonly results in. Through these inferences, the following pattern (among others) emerges: a person is aware that another person is undergoing a negative change. This is the basic pattern which we want to trigger an intervention inference, and in memory it has the form shown in 7-13, that is, two units of information are located in P1's conscious processor: that P2 is undergoing a negative change and that action A is the cause of it. Of course, P1 and P2 might be one and the same person: people try to avoid negative changes to themselves!

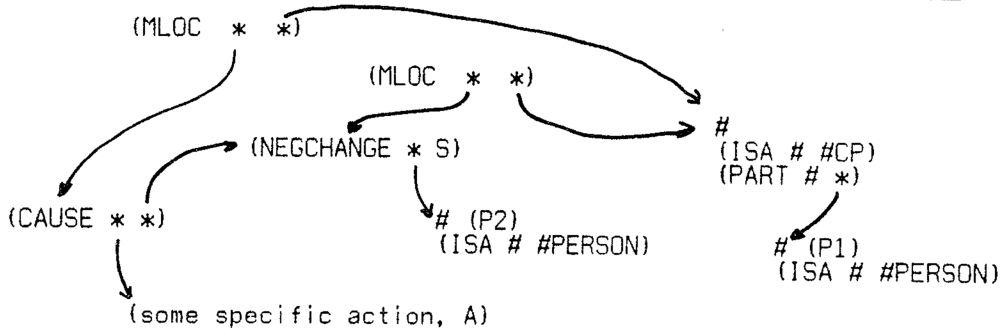



**Figure 6-13. The triggering pattern for intervention inferences.**

In addition to this basic pattern, tests along certain other dimensions must be made in order to make an intervention inference more sensitive to context. The principal check is that P1 must **not presently** have a strongly negative (social) relationship to P2, for otherwise we might rather infer that Bill is enjoying watching Mary give it to John. (Section 7.6 has a computer example where this in fact occurs because of the context established by previous inputs). In the memory, to detect such a negative relationship is to locate a pattern of the form (MFEEL P1 E P2), E being such that (ISA E #NEGEMOTION);

### 6.4.2 THE SUBSTANCE OF AN INTERVENTION INFERENCE

What is the substance of an intervention inference to be, once this pattern has been triggered (ie. detected by the MLOC inference molecule)? The memory's knowledge of extrinsic enabling states plays an important role here. Assuming Bill does not MFEEL a negative emotion toward John, and since Bill knows that the causing action of the NEGCHANGE is Mary's hitting action, and that one way to stop an action is to annul one of its enabling states, it is a reasonable prediction that Bill may desire that one of the common enabling states for Mary's hitting action be removed. Examples of common extrinsic enabling states for this particular action, P1 hitting P2 (underlied by PROPEL CAUSE PHYSCONT), are

1. P1 is LOCated near P2.
2. The object of the propel is LOCated in the hand of P1.
3. P1 is mentally focusing on carrying out the PROPEL action.

There are perhaps others, but in general the number of enabling states for any particular action is not large. The substances of intervention inferences for this example are therefore:

1. Bill may desire that Mary not be LOCated near John
2. Bill may desire that the baseball bat not be LOCated in Mary's hand
3. Bill may desire that Mary cease to CONCeptualize the action she is performing.

By the algorithm I am about to describe, these three enabling conditions will lead to the respective predictions:

(a) Bill might PTRANS the bat away from Mary's hand,

(b) Bill might PTRANS Mary or John away from the LOCation of the other, or

(c) Bill might try to replace the current contents of Mary's conscious processor with something else -- that is, he might try to distract her.

Once again, by making explicit these predictions about Bill's future actions, we stand the chance of linking one of them to a subsequent conceptualization, and this enriches the memory network for this story. The process of detecting and linking such points of contact in inference space is discussed in section 7.5.

How is this ability to make intervention inferences implemented? Again, it is possible to apply the inference monitor to any memory structure, requesting that only inferences of certain theoretical types be generated from that structure. Having detected a pattern of the form shown in Fig. 6-13, the MLOC inference molecule calls the intervention inference process (IIP). The IIP locates the *nearest action* lying on the causal path to the NEGCHANGE (Fig. 6-14), and then, using this feature of selective inference generation, requests the inference monitor to generate the extrinsic enabling inferences from it. In this case, the enabling inferences from Mary's PROPELling action are desired. The result is a list of structures which are the extrinsic enabling states for the propel action.

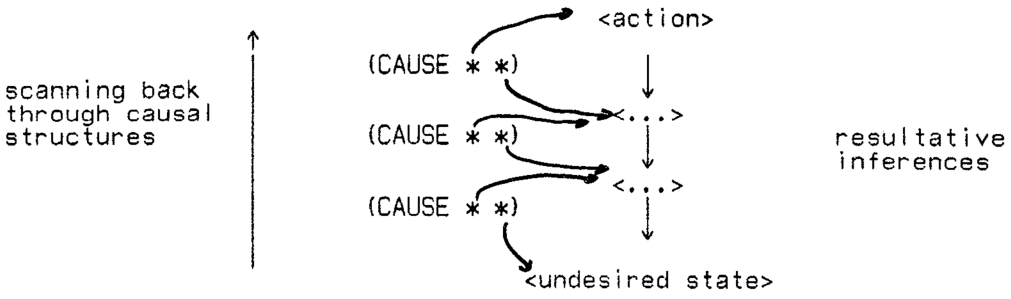



**Figure 6-14. Locating the culprit action.**

The IIP next generates a (WANT P1 (NOT Xi)) structure for each enabling inference, Xi, on this list. At this point in our example, there exists a set of Bill's possible desires. Knowing these desires, it is possible to predict probable sequences of future actions by *Bill* -- what he is likely to do. Since other patterns can also lead to this action-prediction analysis, I distinguish the processing which continues after this point as another class of conceptual inference called *action-prediction* inferences, discussed in the next section. Intervention inferences, therefore,

make predictions about a person's desires. Predictions of what those desires will motivate him to do are the province of action prediction inferences.

### 6.4.3 EXAMPLES

In the first sample, from the intervention inference "Bill possibly wants that the bat not be located in Mary's hand", this action prediction will result in at least the following predictions: "Bill might want to PTRANS the bat away from Mary", and "Bill might PTRANS himself to Mary" (in order to enable himself to PTRANS the bat away from her). And in general, as we will see in the section on action inferences, Bill might perform many other actions which would produce intermediate states lying on the solution path to the goal "get the bat away from Mary."

The solution of the second sample at the beginning of the section is similar. Seeing that Baby Billy's running action can lead, through the result inferences "Billy will be located in the street", and "Someone may PROPEL a car into Billy", to a NEGCHANGE for Billy, Mary will probably intervene. The action "run", is underlied conceptually by a PTRANS (oneself) by the instrument of MOVEing legs and feet. Among others, two important enabling conditions for these actions are (1) that the path to the goal (the street) be unobstructed, and (2) that the feet of the runner be in physical contact with the ground. From this, we may predict that Mary might desire to annul one of these conditions: that she will desire to block his path or pick him up. Since both require (as an an extrinsic enabling condition) that Mary be LOCated near Billy, the prediction that she will PTRANS herself to him can result and provide a point of contact with the second sentence on this sample.

Fig. 6-15 summarizes the intervention inference process.

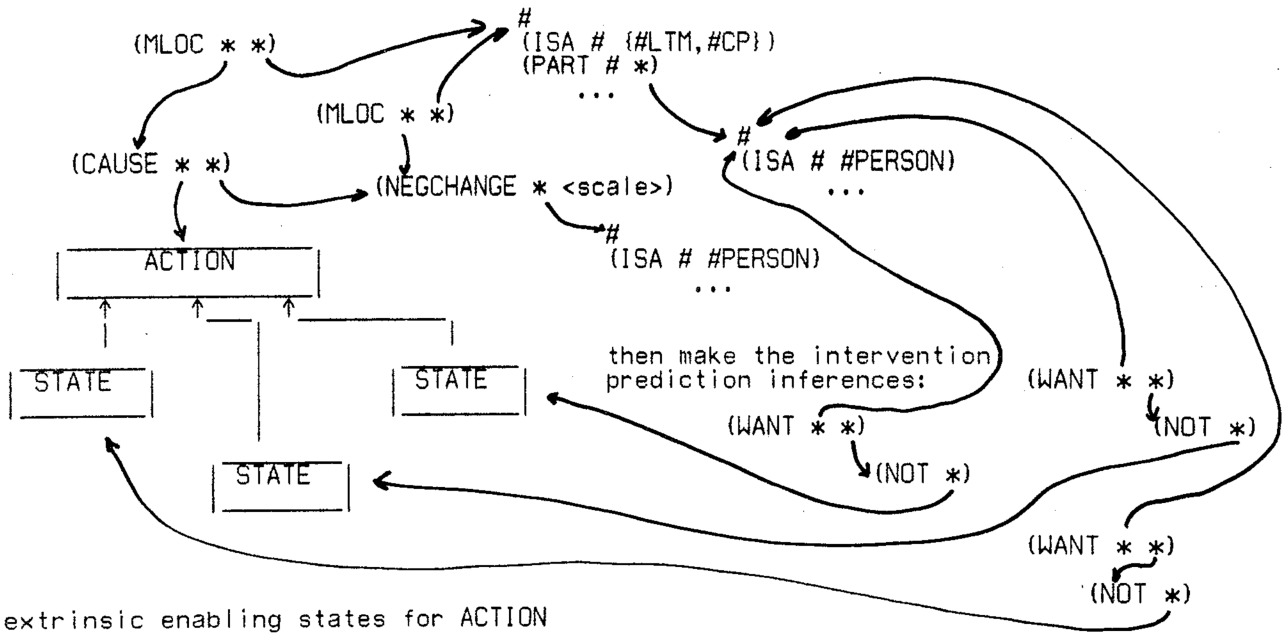



**Figure 6-15. Summary of the intervention process.**

## 6.5 ACTION PREDICTION INFERENCES: APPLYING ALGORITHMIC KNOWLEDGE TO UNDERSTANDING

**sample:** John wanted some nails.
He went to the hardware store

**sample:** Pete needed some milk.
His car wouldn't start.

**sample:** Mary wanted to go to New York.
She called a travel agent.

**sample:** Rita couldn't find her glasses.
She called Bill.

**sample:** John wanted Mary to know how much he loved her.
He bought some flowers.

**sample:** John said the room was cold.
He walked over to the thermostat.

**Knowing a person's WANTs, it is possible and useful to predict his future actions by applying algorithmic knowledge of the world. This includes knowledge of causality relations and extrinsic enabling states.**

In our "naive" psychology whose goal is to understand language, states motivate people to perform actions, which in turn cause other states. In particular, knowing a person's state, and in particular his WANTs, it is often possible and advantageous to predict *future actions* ("future" meaning relative to the time of his WANTs) on his part which might help achieve those WANTs. This section describes how and when this prediction of actions occurs and how it is fundamental to comprehension. I will call inferences which make predictions about a person's likely future actions, based on what he is known -- or can be inferred -- to WANT, *action prediction inferences*. I will often abbrieviate this as simply "predictive" or "prediction" inferences.

Predictive inferences bear something of an inverse relationship to motivational inferences. That is, whereas motivational inferences look at a person's actions and attempt to infer what he might have been trying to accomplish by those actions (that is, what resulting states they WANTed), predictive inferences work forward from a person's wants, attempting to predict what actions those wants might motivate (or have motivated) him to perform.

Although predictive inferences always result from some WANT state of a person, that WANT state may have been inferred from some other source (a resultative, enablement prediction, intervention inferences, for example). In this sense, predictive inferences will in general ultimately result from *all sorts* of mental and physical states of people, since these commonly instill WANTs in the person. However,

> **The process of generating action prediction inferences is always triggered directly by a WANT state of an individual.**

The notion of a predictive inference can be illustrated by two very simple examples:

(1a) John wants some nails.
(1b) John is likely to go to a hardware store.

(2a) John was extremely angry at Bill.
(2b) John might want Bill to suffer some negative change.
(2c) John might do something to hurt Bill.

In both cases, conceptual memory must take some WANT state of an individual and use it to predict likely actions of that individual. (It should be emphasized here that there are of course many other inferences to be made from both (1a) and (2a). I am singling *one* of these out in each

case to illustrate predictions. For instance, the *reason* John wants nails is probably because he is PBUILDing something. However, this is a causative relation which will be generated by the function inference process.)

Both examples show how new predicted action information can be generated from a person's WANTs. In the first example, the want is explicit in the input (1a). In the second, the want (2b) has been inferred via a resultative inference as being likely, and this want leads in turn to (2c) via a predictive inference. The remainder of this section will explain how and when predictive inferences are generated, and how they are vital to understanding.

It should be clear by now that the memory has an implicit algorithmic knowledge of the world. This knowledge is encoded in the form of resultative, causative inferences and in the various forms of enablement inference. The predictive inference process is a realization that one use of this knowledge can be to predict one or more solution paths from a person's current state to some goal state which he is known to desire. To illustrate how this algorithmic knowledge can be applied to make useful predictions about entire sequences of likely actions by a person who is known to want some state, let us trace through the deceptively simple example (1) about wanting some nails. Because it will be easy in this example to lose track of the general goals, let us first describe the general principle of a predictive inference.

### 6.5.1 OVERVIEW OF THE ACTION PREDICTION PROCESS

The abstracted schematic illustrating the idea of action prediction inferences is shown in Fig. 6-16. The general algorithm of the prediction process is as follows: S is a state desired by P. Generate a set of general actions, {A1,...,An}, which could cause S to exist. For each action by P which could achieve this result, infer that he may want to perform that action. Find the enabling states for this action and infer for those which cannot be assumed to exist already that P might also want *them* to exist to enable him to perform Ai. Each of these WANT-enabling states may in turn lead to more action prediction inferences. Do this until no new actions arise, and all enabling states have been satisfied.

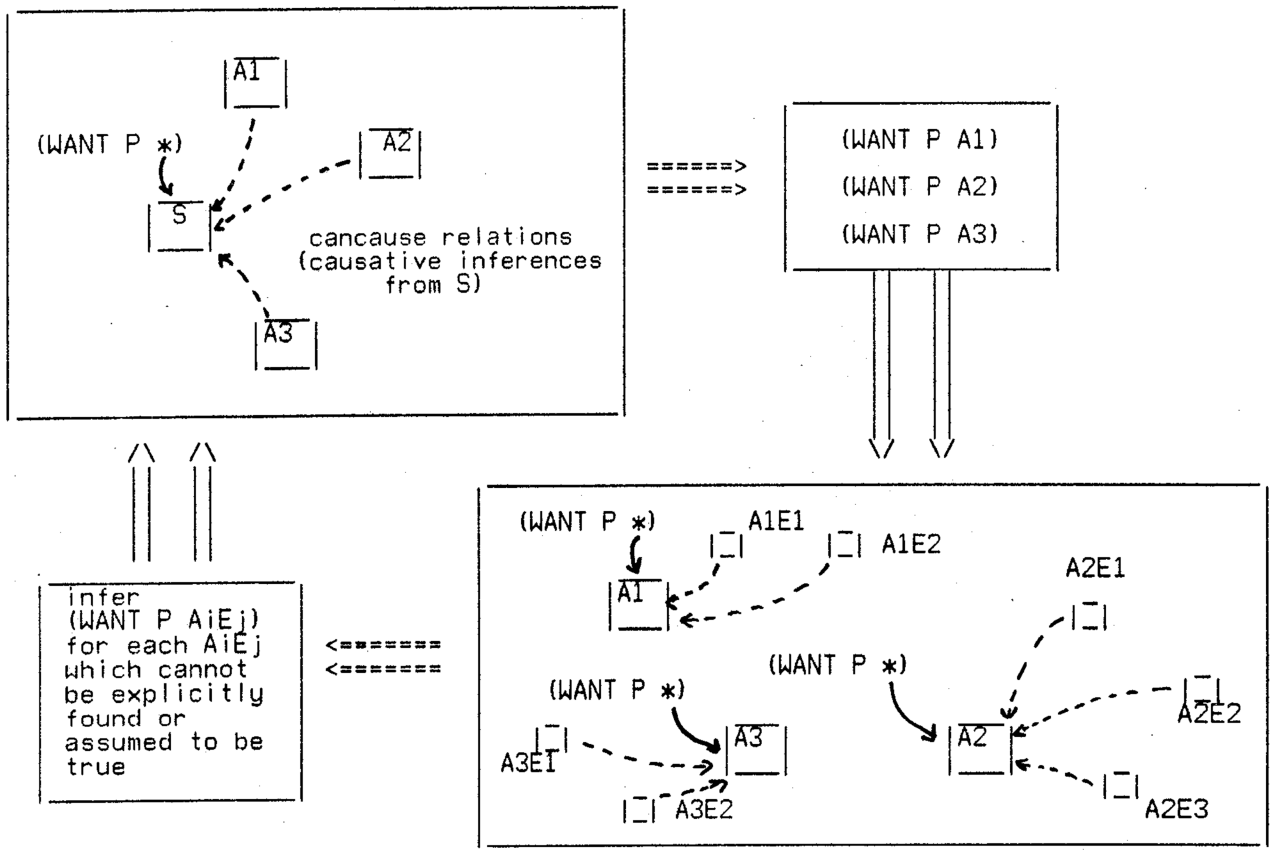



**Figure 6-16. What the action prediction inference process tries to do.**

### 6.5.2 AN ACTION PREDICTION EXAMPLE

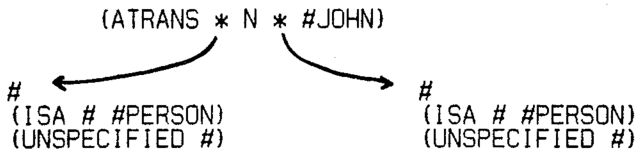
We will now follow through the predictive inference process as it would be performed in response to "John wanted some nails": The question is this: how might John achieve this POSS state which he desires:

(WANT #JOHN *)
↓
(POSS N #JOHN)

where N is a token for some nails (we will ignore the fact that this is a set of objects). The most

general answer to this question seems to be: "by performing some action whose result could be this POSS structure."

To determine what John could do to bring this state about, the predictive inference mechanism requests the inference monitor to generate *causative* inferences from the POSS structure. Stored within the POSS inference molecule is the knowledge that POSS relations are normally caused by ATRANS actions. That is, a normal way to begin possessing something which is not currently possessed is to receive (get, take) it from someone else. This general knowledge is a simple causative inference (stored in the POSS inference molecule). For this example, this means that the monitor's application of the POSS inference molecule to find causative inferences will result in the prediction:




It should be pointed out that there will in general be several possible causes (but still a small number of them) generated as causative inferences from some state structure. The memory must pursue them all in the same way as I will describe for this single case.

At this point the predictive inferencer can thus infer that John may want an action of this form to come about, namely

(WANT #JOHN (ATRANS P1 N P2 #JOHN))

where P1, P2 stand for these as-yet unknown people. Notice that the POSS inference molecule which generated this causative ATRANS inference is able only to predict the *general nature* of the action. Who P could possibly be is not the concern of the causative inference. Hence, at this point there is still no *specific* action prediction, and in particular, no potential action by *John* has yet arisen by this process.

The P1 and P2 which were marked as unspecified by the POSS inference molecule represent general forms of missing information. As such, they are potentially specifiable by a

specifier molecule. The prediction inference process is on the lookout for this type of missing information, since causative inferences typically are only good for predicting general patterns. Because of this, it recognizes the presence of these unspecified entities and requests that the ATRANS *specifier molecule* be applied to this predicted ATRANS structure in an attempt to supply reasonable guesses about the identity of these unspecified people.

In this case, the ATRANS specifier molecule senses that it is being called upon to answer the question: who is the most likely candidate for ATRANSing nails from someone (possibly himself) to John? In this case, the operation of the ATRANS specifier molecule would be the following: the conceptual features of the object of the ATRANS -- the nails -- are examined. In particular, the ATRANS molecule is on the lookout for two general patterns associated with the object. The first pattern is one of the form shown in Fig. 6-17. There, Y stands for some class concept lying on N's ISA set chain at some level. That is, the molecule will use NFCT information to determine whether or not there is some entity whose normal function is to ATRANS things like #NAILs to a person.

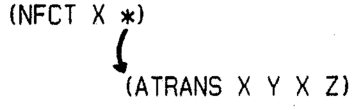



**Figure 6-17.**

The second pattern the ATRANS specifier molecule will attempt to satisfy has the form shown in Fig. 6-18. There, X is someone else who has some nails and who John knows. In a realistic specifier molecule, there would of course be many other similar heuristic tests such as these, and even for these, the level of detail would have to be quite a bit greater. For instance, to narrow the set of potential X's who might be candidates for the ATRANS, the memory might have to check which of them live nearby, which *of those* John is on good terms with, and so forth. I am illustrating here the kinds of things which are realistically attainable in the current implementation.

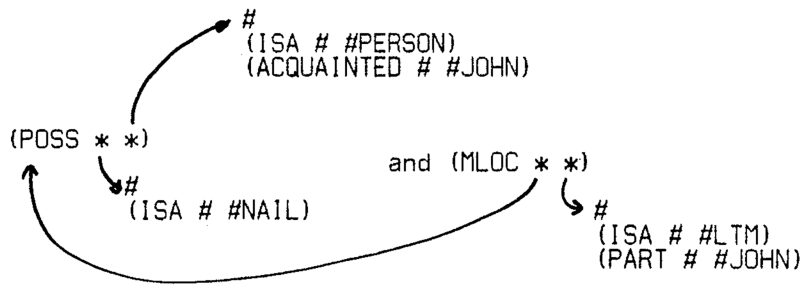



**Figure 6-18.**

In testing for the NFCT pattern of Fig. 6-17, no NFCT will be found involving the ATRANS of either N, or #NAIL. However, since (ISA #NAIL #HARDWARE), the NFCT pattern

(NFCT #HARDWARESTORE *)

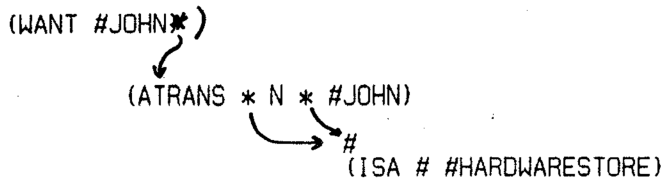
(ATRANS #HARDWARESTORE #HARDWARE #HARDWARESTORE #PERSON)

("the normal function of a hardware store is to dispense hardware") would be located. Recall that entities like #HARDWARESTORE refer to concepts which are bundles of conceptual features. I am using common names only to discuss these concepts.

Having located this pattern, the specifier molecule can thus predict that P1 and P2 might be some hardware store. It will make this prediction, and supply as the REASONS the following structures: because (ISA N #NAIL), (ISA #NAIL #HARDWARE), and (NFCT #HARDWARESTORE (...)).

Currently, specifier molecules stop when one specification is decided upon. Notice, however, that it would be desirable in this application for a specifier molecule to have the ability to return an entire *set* of candidates for missing information, sorted from most likely to least likely. Each prediction would then give rise to an ATRANS structure, some being more likely than others.

After the specifier molecule has been applied and has decided upon #HARDWARESTORE, the previously unspecified WANT-ATRANS structure will have the predicted form




where C2 now has the specification (ISA C2 #HARDWARESTORE). (Personification of things like stores and machines isn't exactly correct, but is sufficient to illustrate the kinds of things the memory is trying to do.)

### 6.5.3 SEEKING AN ACTION ON THE PART OF THE WANTER

But the memory still has not arrived at an action on John's part. Because of this, the predictive inference process again poses the question: what could John do to cause the hardware store to perform this action? That is, it again seeks causative inferences, but this time from this inferred causative ATRANS structure. Because (ISA #HARDWARESTORE #STORE), the most likely candidate causative inference, and the one generated by the ATRANS inference molecule, is that the person must first ATRANS money to the #HARDWARESTORE:

(ATRANS P M P S)

where P is a person, M is some money, and S is some store. In other words, this action can be a cause of the store's ATRANSing something to P. Had the specifier molecule predicted that John might attempt get a friend to ATRANS him some nails, the causative inference would have been quite different: rather than ATRANSing money to the friend, he would probably just ask. This again illustrates how very sensitive all inference molecules must be to features of the involved objects.

Using this new causative inference, the following is inferred:

(WANT #JOHN (ATRANS #JOHN M #JOHN C2))

where M is a token for some money. The prediction process has thus finally arrived at an action for John, the original WANTer. But things do not stop here.

### 6.5.4 PREDICTING THE DESIRE FOR ENABLING STATES

Having found a likely future action by John, the memory must decide whether the action is possible in the current situation. Other actions may first be necessary to get the extrinsic enabling conditions right for the ATRANS. Having arrived at a predicted action by John, the task then becomes to determine (a) which conditions must exist for this ATRANS, and (b) which of these cannot be assumed to exist at the time John wants to perform the ATRANS.

This task is accomplished by calling for the generation of *enabling* inferences from the predicted ATRANS structure. In this example, the following enabling inferences will be returned:

1. (LOC #JOHN H), ie. that John have the same location as the hardware store
2. (POSS #JOHN M), ie. that John have some money to give the store

Having generated these enabling conditions, the memory is interested in determining which can be found already to exist explicitly, or can be assumed to exist based on a knowledge of what is normal (section 6.7). At this point, therefore, the predictive inferencer looks in turn for each of the enabling conditions. For (POSS #JOHN M), the explicit lookup will probably fail: that John possesses some money is simply not likely to be stored explicitly. But the normative inference process will assess this as being very compatible with the memory's knowledge of what is normal in the world: people normally possess money. (We are of course ingoring *quantities* of money here.) Because of this match, no further processing will be done on this POSS enabling inference.

However, in the case of (LOC #JOHN H), if John is not explicitly known to be at the store, it cannot nominally be assumed -- based on a knowledge of what is common in the world -- that he is there. In this case, therefore the predictive inferencer will predict that John WANTs to be at the store:

(WANT #JOHN (LOC #JOHN H))

because otherwise he could not perform the ATRANS action which he may desire to perform.

But we have completed the cycle! The process I have just described can now go to work on this new WANT state, predicting other actions on John's part. In this example, one prediction will occur immediately, since a highly probable cause of something being in a location is that it was PTRANSed there. That is,

(WANT #JOHN (PTRANS X #JOHN Y H))

(John wants to be transported from where he is, X, to the hardware store, H. The PTRANS specifier molecule will be called, and indicate that this usually occurs by the person doing the PTRANS himself. No significant event-enabling inferences will result, and the memory will conclude:

(WANT (PTRANS #JOHN #JOHN Y H))

that is, John might *go* to the hardware store to buy some nails.

It should be pointed out that during this process, causality relations between each of John's successive WANTs have been preserved by the inference mechanism in explicit CAUSE structures so that the memory does not "forget" why various actions are likely to occur. The importance of preservating causal relations has already been emphasized, and this is simply another point where they are important.

### 6.5.5 ACTION PREDICTION INFERENCES' UTILITY

The importance of being able to make intelligent predictions at each point in a conversation or story about what is likely to happen next cannot be overemphasized. It is a fruitful endeavor for two reasons. First, it establishes many new points in the inference space, some of which stand a good chance to be related to subsequent input. Whereas other inferences "reach backward" in this venture, action prediction inferences reach forward. By predicting what is likely to happen, and why, when new information is perceived which matches these predictions, the recognition of new causal and enabling relationships can be almost spontaneous.

Second, by predicting (expecting) certain kinds of things to happen next, many new concepts are drawn into memory's "immediate memory", and these can be of extreme importance to understanding subsequent language forms and references. The second sample at the beginning of this section:

Pete needed some milk.
The car wouldn't start.

is a good illustration of this sort of thing: how is it that we don't balk at a reference to some *car* at this point? We almost certainly would balk if the second sentence had occurred out of the "context" established by the first sentence. Action prediction inferences seem to play a vital role here: knowing that Pete needs some milk, we somehow seem subconsciously and automatically to know in a general way what this situation emcompasses, and what Pete might do in such a situation. That is, we somehow anticipate at that point what *sorts* of things Pete might do, and one of them involves a sequence of actions which would get him to a grocery store. Having drawn out these actions which involve some sort of transportation, it is no surprise at all to hear about a car in the next sentence.

I interpret this sort of phenomenon as supporting evidence for the action prediction inference process. Although the example I carried through above may seem a bit awkward and tedious (perhaps because I am trying to have the memory be too specific in its predictions), the feeling is that it represents a very real thing in people. This is a first step toward a "fuzzier" predictive capability.

### 6.5.6 AN INADEQUACY

One flaw with the approach to predictive inferences as I have described it may have become evident. It is this: as with just about every inference the memory makes, action prediction inferences are effectively modeling *another person*, rather than the memory itself. Because John's knowledge about how to go about acquiring nails might be totally different from the memory's, if we don't model *his* knowledge, the predictions the memory generates may be totally irrelevant. This is a recurring theme, and it has been addressed to varying degrees in the solution of other classes of inferences. I will have more to say about it in section 6.6. Even where this need to model other people's knowledge *has* been taken into account, its implementation in the model is weak, and the whole topic requires much more research. Nevertheless, at the level of information complexity at which we are dealing, the assumption that everyone possesses approximately the same knowledge is not at all unrealistic. For this reason, this inability to model other people in any detail is not really yet a handicap.

### 6.5.7 GENERALIZING ACTION PREDICTION INFERENCES: ENLISTMENT PREDICTION

We have seen the sort of capability the memory should have in order to predict actions of a person by considering those actions he himself could perform which would *directly* achieve his goals. However, a person frequently enlists the services of others to help achieve goals which he either cannot, or prefers not to, achieve alone.

We saw an example of this as it related to predicting a person's motivations in the computer example at the end of section 6.10. There, P1 tells P2 that P3 wants an X. One possible motivation which was discovered was that P1 *may* have done this so that P2 would know that P3 needed an X, and as a consequence of this, would perhaps give P3 an X. That is, P1 *enlisted* the services of P2 as one means of satisfying his own want, namely that P3 have an X. It would be desirable to have the ability not only to work "backward" in motivation-establishing mode, but also to work forward under certain circumstances to predicted enlisting actions on P1's part. In this example, this would mean starting at P1's desire that P3 have an X (P3 can of course be P1!), and working forward to predictions about how P3 might enlist someone else's service to accomplish this goal.

Although I do not propose to delve into general problems of knowing when and how to predict one person will attempt to enlist the services of another, there is one obvious point in this action prediction process where the idea of enlistment fits. It is this: when an action on the part of an actor can be predicted, and there is one or more extrinsic enabling conditions which does not exist and which *cannot be caused to exist by the actor*, it is reasonable to predict that he may request that someone else who *can* cause the necessary condition to exist either perform the action for him, or do some other action which would cause the missing condition to exist. For instance, when, on a vacation, we remember we left the water running at home, we want the water to be stopped. An immediate prediction is that we will want to turn the lever on the fixture. But an enabling condition that we have the same location as the fixture to perform this action is not easily met. Under these conditions, the memory could search for someone with whom we were acquainted, and who satisfied this LOC property, then predict that we might request of this person that he perform the action for us.

There is much potential for research in this area, and much of it spills over into the domain

of conversation, since instilling desires in other people is more central to the idea of an enlistment inference than to the other types of conceptual inference I have been discussing. I merely want to point to it as an unexplored topic related to action prediction inferences.

## 6.6 KNOWLEDGE-PROPAGATION INFERENCES

**sample:** Pete told Bill that Mary hit John with a baseball bat.
Bill knew that John had been hurt.

**sample:** John saw Bill kiss Mary.
John probably believes that Bill feels
a positive emotion toward Mary.

Many particular inferences in classes I have been discussing rely upon information, either explicitly conveyed or inferred, about what information and knowledge of the world is available to a person at a particular time. For example, we have seen how result, intervention and motivational inferences, respectively, require information about who knows what, using this information to infer other information of various types. The realization is that the *knowledge* of some state in the world, rather than simply the existence of the state, is the crucial factor in motivating the actor to act. For instance, in section 6.4, we were able to generate an intervention inference about Bill because we could infer that he *knew* of John's NEGCHANGE and that it resulted from Mary's ation. The intervention inference did not arise simply because of John's NEGCHANGE. It is therefore of immense interest to the memory to keep extensive models of who knows what, and when. Let us call inferences which implement this modeling *knowledge propagation* inferences.

The rough idea of a knowledge propagation inference is this:

> **if P knows information X, and (I1, ... ,In) are inferences (of all theoretical classes) which arise from X in the memory, then it is possible that P also has knowledge of (I1, ..., In).**

That is, assuming P has access to the same knowledge the memory has access to, he is likely to be aware of many of the same consequences (inferences) of that knowledge that the memory is.

The first sample illustrates a very simple instance of knowledge-propagation. In this example, two immediate inferences in the memory are (a) that Bill probably believes that Mary

was hitting John, and (b) that Mary probably was hitting John (ie. the memory will also believe Pete). Further, among other things it will probably be inferred (1) that John probably suffered a negative change in his PSTATE, (2) that it was Mary's action that caused it, (3) that Mary was probably mad at John, (4) that her anger motivated her to want John's NEGCHANGE, and hence the hitting action, (5) that previous to this incident, John might have caused some kind of NEGCHANGE for Mary, and that this is perhaps what caused her anger, (6) that John might have become angry at her as a result of this incident, (7) that Mary and John were near each other at the time of the incident, and so on. But, since Bill has also become aware of the incident (that is, this information is now located in Bill's CP is a direct resultative inference from the underlying MTRANS from Pete), each inference which stemmed from it *in the memory* might also be an inference which *Bill* made upon hearing this news from Pete. Furthermore, since Pete MTRANSed this information from his CP to Bill's CP, and since a very important enabling condition for an MTRANS is that the mental object first have the location from which it is MTRANSed (Pete's CP), *Pete* himself may be predicted to know much of this inferred information as well.

This draws out two important questions: (a) does a human language user really make all these knowledge-propagation inferences, and (b) how are knowledge propagation inferences sensitive to differences between the knowledge available to MEMORY and the knowledge available to another person who is involved in the memory's knowledge propagation inferences?

Because an awareness of who knows what at any given time in a particular situation seems to be so vital to the other kinds of conceptual inference we have been and will be discussing, we must conclude that the generation of many knowledge propagation inferences at this presumed subconscious stratum is a reality. Of course, as with all conceptual inferences at this cognitive level, many of these inferences may not prove to be of much use. Still, they must be generated in copious detail in hopes of "fueling the fire" -- of discovering what *will* be useful toward discovering interesting lines of inference.

The second question is not a simple one to answer: any two people stand the chance of making slightly or even totally different interpretations of a given experience. And this happens, of course, mainly because the knowledge they apply to their interpretation (the inferences they generate) is slightly or totally different. For instance, if Bill knows that John is a masochist, and that this incident occurred as part of Mary and John's Saturday night ritual, he will reach totally

different conclusions from Pete, who doesn't know about John's peculiarities. By the same token, if the memory knows about John, but Pete and/or Bill do not, it would certainly be incorrect to infer that Pete and/or Bill believed that John was deriving pleasure from the event.

In their "pure" sense, these are fairly deep, perhaps unsolvable, philosophical issues. But they are not beyond our grasp at the level at which we require a solution: the fact is that human language users are capable of modeling other users' knowledge and of putting this capability to use in understanding. The method by which the memory can be sensitive to differences in people's knowledge will be outlined shortly. But first I will describe the how and when of the mechanism which generates knowledge propagation inferences in MEMORY.

### 6.6.1 GENERATING KNOWLEDGE PROPAGATION INFERENCES

Knowledge propagation inferences --just as motivational inferences -- must be generated at a different time from most ordinary conceptual inferences. Each of the potentially numerous conceptual structures in an utterance is identified, and passed to the inference monitor for expansion in inference space. This process will normally result in a large number of inference spheres about each starting structure. When this process terminates, the result is a list, !INFERENCES, of all inferences which have arisen from the starting structures.

At that point, the POSTSCAN process is entered. (This process is also related to the generation of motivational inferences). Relative to the task of generating knowledge propagation inferences, the postscanner looks for inference structures (with TRUTH=TRUE -- the memory must believe the structure to some degree) of the form shown in Fig. 6-19. Call any structure of this form S.

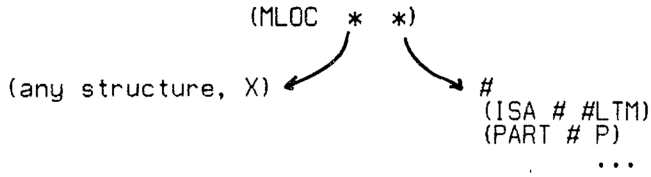



**Figure 6-19.**

That is, the postscanner detects structures which indicate that some information, X, is MLOCated

in someone's LTM, L -- that he believes, or knows X. For each structure satisfying this pattern, the postscanner will attempt to generate knowledge propagation inferences.

In a simple model which is not sensitive to differences between its own knowledge and the knowledge mentally located in L (which is part of P), the next step is to retrieve the OFFSPRING list for structure X (Fig. 6-20). Recall that OFFSPRING and REASONS together preserve "inference connectivity", a record of what arose from what, in inference space. Hence, the OFFSPRING set for X is a set, OFFS(X), of other inference structures in whose generation X played a part *in the memory*. If the memory operates under the simplifying assumption that P's knowledge and conceptual inferences are the same as its own, then the inferences, OFFS(X), it was able to infer from X were (are) probably also inferable by P. Using this assumption, the memory can then generate new inference structures of the form (MLOC I i L) for each I i in OFFS(X).

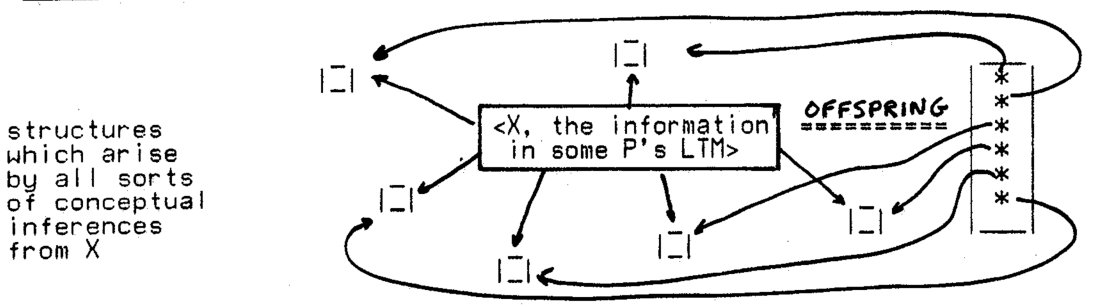



**Figure 6-20. The offspring set.**

### 6.6.1.1 MODELING THE KNOWER'S KNOWLEDGE

This works accurately enough for many inferences which are based upon common world knowledge. For instance, if a person knows that John kissed Mary, he will normally infer that John felt a positive emotion toward Mary, that he was near her at the time, and so forth. But it is in general not adequate because no model of the hearer's knowledge has been incorporated. Where then does the modeling of the other person fit?

We must view OFFS(X) as simply a set of *candidates* for what P possibly infers from X. For each I i in this set, there is an associated REASONS set, which records exactly *which other information* in the memory played a part in the generation of I i as an inference from X. Before

generating a knowledge propagation inference from any I i in OFFS(X), the REASONS set, R(I i), for I i must therefore be examined. For each reason, *r*, in this set the question must be asked: "I (the memory) knew r, but is P likely to have known it"? To answer this question, the postscanner attempts to locate a (MLOC *r* L) structure. A failure (as is frequently the case) to find such a structure indicates one of two things: (a) (MLOC *r* L) was not found simply because it truly does not (cannot be assumed to) exist, or (b) it was not found simply because it has never been made an explicit structure in the memory, but it is something which a person who meets certain minimal requirements (say, a normal adult) can be *expected* to know.

This abuts with memory's assumptions about what is normal in the world (next section). There I will describe how patterns such as "Most people know that the normal function of a book is for reading" can be stored. This example, for instance, is a predication that this knowledge about books is commonly located in the abstract #LTM which is PART of this abstract concept #PERSON. But to represent and match passive patterns of this complexity is not a convenient approach. Hence, much "common knowledge" of this sort has been implemented in the form of programs, called *normality molecules* (N-molecules). N-molecules are not sensitive to context, since their purpose is to relate specific information to "default" assumptions about the world. N-molecules give the memory a much larger *apparent* storehouse of knowledge by applying these default assumptions to specific instances.

Briefly, an N-molecule (next section), when applied to some *specific* memory structure such as "John Smith knows that most people sleep at night", will return a STRENGTH, which is a measure of the likelihood of that pattern, based on the memory's "default" knowledge of the world.

Therefore, when the knowledge propagation process fails to locate an (MLOC *r* L) explicitly, it creates an (MLOC *r* L) structure, and requests that the MLOC N-molecule be applied to it. The result of the N-molecule's assessment of the structure will be a STRENGTH which is a measure of the likelihood of this MLOC. If the N-molecule can judge it, and its judgement is non-zero, the knowledge propagation inference continues its examination of OFFS(X). If the assessed STRENGTH is zero, the knowledge cannot be assumed to have been possessed by the knower, and a knowledge propagation inference cannot be made.

If all the candidates in OFFS(X) can be determined also to have been known, the knowledge propagation inference that the knower of X also knows this I i which arose from X -- (MLOC I i L) is generated, and assigned as REASONS X, and all the (MLOC *r* L) structures which are the reason for *the memory's* belief of I i. The entire process is illustrated in Fig. 6-21.

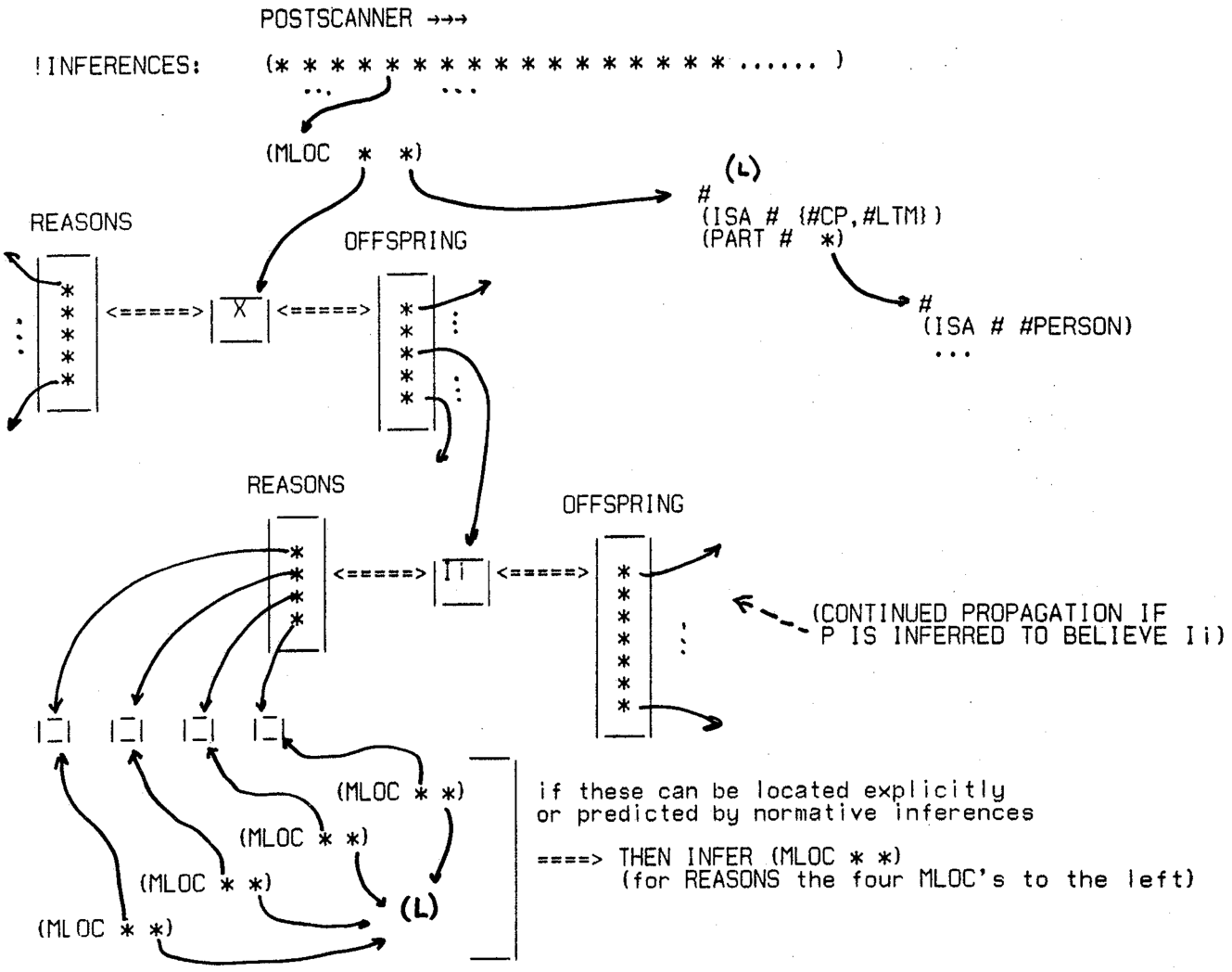



**Figure 6-21. The knowledge propagation inference process.**

### 6.6.2 AVOIDING THE "HE KNOWS THAT HE KNOWS THAT ..." PROBLEM

The process of generating knowledge propagation inferences occurs after the expansion of inferences by the normal monitor process by a postscanner. The inferences thus generated will go onto the inference queue where they will be expanded by subsequent inference passes.

But because inferences in the inference queue will normally undergo at least another pass through the inference monitor, and because knowledge propagation inferences augment the OFFSPRING set of some of the existing inferences on the queue, there is a problem. Suppose X gives rise to knowledge propagation inference Y. Y is put on the inference queue, and it is also recorded on X's OFFSPRING set. On the next pass through the inference monitor, X would be examined for knowledge propagation inferences, and this new member of the OFFSPRING set, Y, would be seen and a knowledge propagation inference generated for it. But the substance of this new inference would be "P knows that P knows that X", and this is not desirable, especially since it will happen all over again on the next pass.

To prevent this, the postscanner tags with its own special tag all inferences on the queue which it has processed once for knowledge propagation inferences. On subsequent passes, if this tag is detected, no further knowledge propagation inferences will be generated for any of the new offspring which have the form (MLOC ...) and which were generated on a previous pass.

## 6.7 NORMATIVE INFERENCES

**sample:** Does John Doe own a book?
Probably. Middle-class business executives normally own books.

**sample:** Was Mary Smith at work Tuesday morning?
I don't know, but she has a job, so she probably was.

**sample:** Was John at home Tuesday evening?
I don't know. There's a good chance of it, though.

**sample:** Does Pete have a gall bladder?
It's highly likely.

**sample:** Is the normal length of time required to read a book a few minutes?
No, not usually.

**sample:** Is it unusual that John was asleep a 3PM yesterday?
Mildly unusual. He normally is at work then.

**sample:** Does Mary know that John normally sleeps at night?
Probably. Most people know that people sleep at night.

**sample:** John saw Mary at the beach Tuesday morning.
Why was John at the beach then? He normally is at work in the morning.

**sample:** John loves Mary. Does John want Mary dead?
Extremely unlikely.

A human language user applies *staggering* amounts of knowledge to the understanding of even the simplest utterances. Part of this knowledge is specific from situation to situation, and from special case to special case. But part of it is implicit in common assumptions and knowledge of the world. In the description of the specification inference process, it was illustrated how filling in implied but unspecified information can rely heavily upon a knowledge of what is normal in the world in the absence of overriding context. There, this normative knowledge was used to predict - to add on - missing information in the hope that this would draw out implicit references, open up new lines of inference, and so forth. Also, we have seen how applying assumptions about the normal functions of objects can lead to quite interesting new sectors in inference space. In reality, *every* class of inference makes implicit reliance upon assumptions about what is normal in the world in given contexts. This reliance is so pervasive that I would like to draw it out and identify it as a form of conceptual inference. What should the nature of such an inference be?

The key point is this: by recognizing *specific* patterns as instances of *general* patterns of what is normal in the world, a language user can operate as though he possessed an *apparently limitless* amount of specific world knowledge. The idea is that, even though very few instances of

a general pattern ever actually come into existence as explicit memory structures, the *potential* for generating instances is always there, and should be applied when some specific instance of a normative pattern would be of use to some process. For example, if the knowledge propagation inference mechanism needs to know (say, in order to generate an inference) whether John Smith knows that a kiss is a sign of affection, the general knowledge that just about everyone has this knowledge ought to be applied to this specific case, even though this would hardly ever be stored explicitly.

What I propose, then, is that the memory should recognize that much of its knowledge is stored only as a *potential*, which is embodied in many general patterns. This is a departure from precise systems (in which the world consists of a well-defined data base of explicit facts), to a "fuzzier" system in which much more is actually known than what is explicitly stored. In this fuzzy system, failures to locate a needed piece of information explicitly should not be interpreted to mean the information is not true. Rather, the conclusion that the information is false or improbable should be assumed only after an attempt to verify it as an instance of a more general pattern fails. This means that

> **Every time an information lookup fails to locate information which is necessary to some inference process, the memory should attempt to apply its knowledge of normality to that information before concluding the information does not exist.**

I will call such a successful attempt a *normative inference.*

The approach to storing normative information which involves the least reliance upon large, passive data patterns seems the most attractive for the level of complexity at which we are dealing. That is, whereas it may be desirable to encode the simple knowledge "Almost everybody has a gall bladder" in a passive pattern (PART #GALLBLADER #PERSON) (in other words, by predicating a PART relationship between the abstract concepts, with the convention that this is interpreted as a predication about people and gall bladders in general), it is less desirable for patterns like "All veterans of World War II who were living in Minnesota earn at least $15,000 a year", or "Most healthy adults can drive a car, but few children can, no matter how healthy they are." This is because more complex patterns involve many dimensions, many conditions in general must obtain, and *which* conditions which must obtain is often a function of many complicated conceptual features of the objects, times, locations, etc., of the entities involved in the pattern.

### 6.7.1 NORMALITY MOLECULES

Because of this, normative inferences in the memory are made by LISP programs called *normality molecules*, which we can abbrieviate as N-molecules. As specifier and inference molecules, N-molecules are organized by conceptual predicates: there is an ATRANS N-molecule, an MLOC N-molecule, a PART N-molecule, and so forth. The function of an N-molecule is this: when applied to a memory structure, X, involving predicate P, the N-molecule for P performs tests on X and returns a STRENGTH, S(X), which is a real number between 0 and 1. This S(X) is a measure of how strongly the molecule "believes" the specific structure, X, insofar as X conforms to its encoded knowledge of what is normal in the world. That is, X will be a measure of how "normal" or "unusual" the structure is. In the terminology of fuzzy set logic [Z1], this S(X) is a measure of the *compatibility* of X -- how compatible it is with assumptions (pattern information) about the world.

This number, S(X), *is* the normative inference for structure X. A normative inference therefore differs from other types of conceptual inference in that its content is not a new memory structure, but a number which assesses the compatibility of an existing structure such as (PART C1321 #JOHN) (C1321 being a token of a gall bladder), or (MLOC C8768 9924), C9924 being Pete's LTM, and C8768 the conceptual structure for "Jim owns a car.". For these two examples, the numbers returned would indicate the STRENGTH with which these structures can be believed, based on the knowledge of normality contained in the PART and MLOC N-molecules, respectively.

#### 6.7.1.1 ASSESSING A STRUCTURE'S COMPATIBILITY

It may seem as though I am proposing to solve a very difficult problem simply by compartmentalizing it in some abstract process which magically assesses an arbitrary memory structure's compatibility with assumptions about normality. This is not the case; I am not proposing some sort of alchemy whose goal is to get something for nothing. While it *is* true that this compartmentalization is convenient, there is nothing mysterious about an N-molecule; it is a very candid construction. Fig. 6-22 shows one.

For *every* piece of normative information in the world we desire the memory to possess, there must be an "N-atom" within the appropriate N-molecule which will test of a structure

whether that structure conforms to its pattern. These tests are not fuzzy. The only fuzzy component is the S(X) which is returned: it is an estimate of X's truth, and as such it will be propagated by inferences which rely upon X. S(X) is *not* a measure of the degree to which the N-molecule was successful in matching X to some normative pattern. Rather it is the STRENGTH associated with some normative pattern which is *fully successful* in matching X: the process of matching merely serves to select some well-defined compatibility (Fig. 6-22). Fig. 6-23 shows a very simple specific N-molecule.

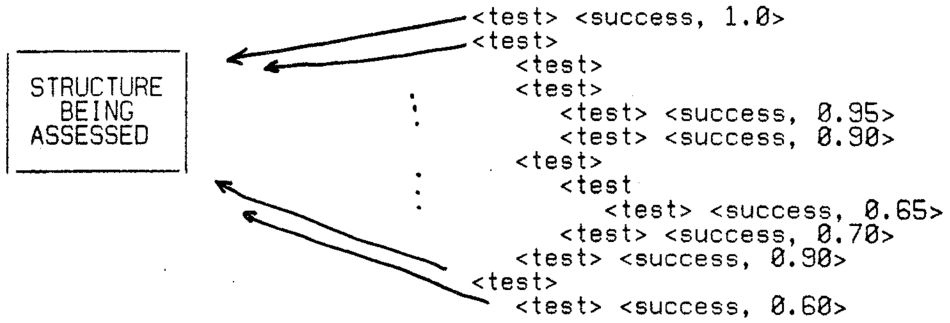



**Figure 6-22. Precise testing to arrive at a fuzzy compatibility.**

```
is P a member of a pure communal society, or is it an infant?
        if so, very unlikely that P owns X
        otherwise, does X have any distinctive conceptual features?
                if so, assess each one, form the product of likelihoods, and call it
                M. M will be used at the end to mitigate the likelihood which would
                normally be assigned.

        is X living?
                if so, is X a person?
                        is P a slave owner, and does X possess characteristics
                        of a slave? if so, likelihood is low but non-zero
                        otherwise likelihood is zero
                otherwise, is X a non-human animal or a plant?
                        if so, is X domestic in P's culture?
                                if so, does P have a fear of X's or is
                                P allergic to X's of this type?
                                        if so, likelihood is low
                                otherwise, likelihood is moderate
                        otherwise, is X related to actions P does in any special
                        way?
                                if so, likelihood is low, but non-zero
                                otherwise, likelihood is near-zero
        otherwise, does X have a normal function?
                if so, does P do actions like this normal function? (Note here
                that we would want to look at P's profession, and actions commonly
                associated with that profession.)
                        if so, likelihood is moderately high
                otherwise, is X a common personal item?
                        if so, is it's value within P'means
                                if so, likelihood is high
                        if not, likelihood is low, but non zero
                otherwise, is X a common household item?
                        if so, is P a homeowner?
                                if so, is X within P's means?
                                        if so, likelihood is high
                                        otherwise, likelihood is moderate
                                otherwise, likelihood is low, but non-zero
and so on ...
```

**Figure 6-23. How we might go about deciding whether person P owns an X.**

### 6.7.2 INSIDE AN N-MOLECULE

What kinds of tests can a typical N-molecule be expected to make in testing whether a specific structure matches some general pattern about the world? Although heuristics will certainly vary from case to case, there are three general types to which all tests should be sensitive.

Consider the OWN N-molecule's assessment of this (OWN C7536 #BILL) structure, where

C7536 is a token of a hammer. The first obvious test is that C7536 not be OWNed by someone else other than Bill at the time in question. In this case, because of specific knowledge about C7536, we would want S(X) to be 0. Notice that if this is the case, it will be detected only in the N-molecule, because the original attempt to locate the (OWN C7536 #BILL) structure will fail, but not because another OWN structure existed, only because this particular one did not exist.

A second general heuristic is that an N-molecule must watch out for "over-specified" objects in the structures they assess. Consider the following progression of queries:

1. Can I assume that John Smith owns something?
2. Can I assume that John Smith owns a hammer?
3. Can I assume that John Smith owns a claw hammer with a wooden handle?
4. Can I assume that John Smith owns a 16 oz. Stanley claw hammer with a steel-reinforced wooden handle and a tack puller on the claw?

We would certainly want the answer to be an unequivocal "yes" for the first case, if John is an adult and not a member of a pure communal society. On the other hand, queries 2, 3 and 4 specify progressively more conceptual features of the hammer (we might just have well specified progressively more features of John), and in general, each query is less likely because of these features. S(X) for (4) should be very near zero, while S(X) for (2), depending on other features of John, should be closer to 1.0. Although there are clearly well-defined ways to make an N-molecule sensitive to such overspecification, and though this is intuitively the correct theoretical approach, it is equally clear that we are only on the periphery of an awesome infinity of pattern information about the world. It takes the average adult 15 or 20 years of experience to accumulate enough knowledge of normality in the world to answer questions 2-4, so we cannot expect to make much of a dent in this knowledge with the few simple N-molecules which exist in the memory!

However, there is one interesting interaction which helps ease this apparent infinity of pattern knowledge about various things. N-molecules themselves ask many questions in order to assess S(X) for structure X. And the questions they ask will in general look no different from the questions asked by other inference processes whose queries invoke N-molecules in the first place. Why not give each N-molecule the potential for calling upon *other N-molecules* to answer tests for features which are not explicitly locatable? Suppose for example, that, in the process of

assessing (3) above, some N-atom in the OWN N-molecule decides that the compatibility of John possessing something which ISA #HAMMER is 0.95. But it can't decide what effect the additional features (the composition of it's handle, and the shape of its head) should have upon the ultimate S(X). At that point, it could call other appropriate N-molecules to assess these two properties of hammers: "C7536 has a wooden handle" and "C7536 has a claw-shaped head". It could then use the assessments of these two N-molecules to equivocate the 0.95 which would be returned in the absence of these features. The resulting S(X) would then reflect the following lines of thinking: "Yes, it's pretty likely that John OWNS a hammer, because John is a middle-class, male suburbanite. And since many hammers have wooden handles, and most have claw-shaped heads, the chances are good that this hammer which he is likely to own fits these descriptions. I will therefore assess this with a moderately high compatibility."

Modeling other people's knowledge is essential to certain types of inference. It can be expected therefore that various processes will frequently require the assessment of (MLOC X #LTM), that is, does some person, P, know (believe) X. Rather than encode in the MLOC N-molecule the explicit knowledge of what people normally do and do not know, the MLOC N-molecule defers most such decisions to the N-molecule which assesses X directly, with the constraint that, should any REASONS be returned along with the S(X) returned by the sub-contracted N-molecule, P must be verified to have knowledge of these as well.

Of course, just as tokens and concepts must be checked by N-molecules for "overspecification", so too must an information-bearing structure. That is, the assessment of structure X must be sensitive not only to the conceptual features of the objects X relates, but also to conceptual features *of X itself*. A good illustration of this is to contrast

John was asleep at 3AM.
John was asleep at 3PM.

Assume in both cases that the main structure to be assessed is "John is asleep", that is, (AWARE #JOHN -5). In order to assess either structure, the AWARE N-molecule defers most of the decision to the N-molecules which assess the features on the (AWARE #JOHN -5) structure's occurrence set. Here, only TIME features are present. The TIME N-molecule can therefore assess (TIME (AWARE #JOHN -5) 3AM) much higher that (TIME (AWARE #JOHN -5) 3PM), unless, by its own special heuristics, it detects some special information such as "John is a night watchman".

It was mentioned that there are three general heuristics common to all N-molecules. The third is that there must in general be tests which "massage" information which is stored in one form with potentially equivalent information, but which happens to be stored in another form. For example, section 8.1 will illustrate how the process of inferencing interacts with the process of reference establishment. In the example discussed throughout that section, a crucial realization turns out to be that (AGE #ANDY #ORDERMONTHS) (ie. that Andy's age is on the order of months) is highly compatible with (TS #ANDY #7MAR72) (ie. that Andy was born March 7, 1972). The AGE information was generated as an inference which, when discovered to match more closely with (TS #ANDY1 #7MAR72) than (TS #ANDY2 #1JUN48), serves to choose baby Andy Rieger rather than adult Andy Jones as the referent of "Andy" in the example sentence. The point is that the process of discovering compatibility in that example was based on special knowledge contained in the AGE N-molecule which relates certain forms of AGE structures with certain forms of TS structures another. It is this ability of individual N-molecules which allows MEMORY to perform and use fuzzy matching: because the AGE N-molecule knew, among other things, to check for applicable TS relations, the compatibility of these two structures was realized, and helped solve a reference in that case.

#### 6.7.2.1 SUPPLYING THE REASONS FOR THE ASSESSMENT

There is one final issue of normative inferences as they have been implemented. It is imperative that the memory preserve a record of connectivity in inference space: that MEMORY record the REASONS and OFFSPRING for each information unit it stores. It is therefore also a requirement of the normative inference process to supply any reasons (a list of other structures in the memory) which lead to its assessment S(X) of structure X. In cases where X remains as a memory structure after assessment (for example it plays a part in the generation of another inference), these reasons are attached to structure X as its REASONS property. This means that the N-atom which successfully assesses X must make explicit those facts it used. These facts correspond exactly to those successful tests it made which lead up to some fuzzy compatibility, as shown in Fig. 6-22.

### 6.7.3 WHERE N-MOLECULES ARE USEFUL

The normative inference process is not an isolated one. Rather, its purpose is to serve other processes which need access to this kind of fuzzy knowledge. Requests for normative inferences arise principally at the following points in conceptual processing:

(1) when some inference molecule requires a unit of information, which can not be located explicitly, in order to generate its inference.

(2) when an inference has been generated and the inference monitor needs to compare it to its knowledge of normality for the purposes of determining what is potentially most interesting. This gives a slight goal direction to the process of spontaneous expansion of a structure in inference space.

(3) when an question has been asked of the memory for which no explicit answer can be found.

Requests in the second category require only the S(X) which is the result of the normative inference. Requests in categories (1) and (3) however generally result in the creation of a new memory structure if the normative inference returns an S(X) greater than 0. For instance, if an action prediction inference needs to know whether Bill owns a hammer, and this is not stored explicitly, a temporary structure, X, which represents this ownership, (OWN C7536 #BILL), where C7536 is a token of a hammer, must be created in order that the OWN N-molecule have a structure to assess. If the assessment, S(X), turns out non-zero, or greater that 0.5, or whatever the process which requested its assessment requires it to be, then the structure can remain, in its now explicit form, with STRENGTH equal to the S(X) supplied by the OWN N-molecule. Thus, just as a *specifier* molecule gives rise to a new unit of information as the result of a missing case, an *N-molecule* can give rise to a specific instance of a general pattern when it is needed by a process in category (1) or (3).

#### 6.7.3.1 HOW N-MOLECULES MASSAGE FUZZY MATCHES

I have characterized the central purpose of an N-molecule as being to assess how compatible some new structure which enters the memory by inference is with the rest of the memory's knowledge. That is, should a required fact for some inference not be explicitly locatable, an N-molecule should then be called to attempt to assess the likelihood of X as a specific instance of more general patterns in the world which are believed to be true. It should be emphasized that this process will in general not be a clean one, but rather it will rely on case-to-case special heuristics. The N-molecule construct is where these heuristics can exist.

The N-molecule is where data lookups can be given a certain degree of fuzziness. For example, suppose some inference needs to know whether X is touching Y, that is (PHYSCONT X Y), but only the information (PHYSCONT Y X) is stored in the memory. The PHYSCONT N-molecule is the ideal place to encode PHYSCONT's symmetricity: when the straightforward lookup fails, the structure (PHYSCONT X Y) is simply created, then assessed by the PHYSCONT N-molecule, which, among others, applies the special heuristic that PHYSCONT is symmetric. In a sense, this is a very primitive form of fuzzy matching, and it is not hard to imagine many subtler forms.

The notion of an N-molecule is reminiscent of a theorem in a traditional, task-oriented system. That is, given that some information cannot explicitly be located in the data, what can a special procedure (a theorem) do to help out. In a sense, each N-atom is a theorem which brings a knowledge of normality and special heuristics to bear on specific units of information. However, an N-molecule is viewed as "something to do when all else fails", whereas the traditional utility of a theorem is a far more central process. That is, because the system tries spontaneously to make everything as explicit as possible, the assumption is that most information which is true will be drawn out explicitly, and this leaves little work for the traditional theorem. However, I am not proposing that all information will be drawn out, or that it is desirable to go too far in this process. The concept of an N-molecule will undoubtedly have to be extended to accomodate the traditionally more involved operations of a theorem prover. But, however extensive this capability is, it will remain ancillary to the spontaneous expansion of structures by conceptual inference.

### 6.7.4 SUGGESTIONS FOR RESEARCH

Assessing the "normality" of a memory structure is a very sophisticated process. It, however, provides a very important focus for memory research. By asking "What other knowledge could affect the likelihood of X being true?", we spill over into every conceivable topic of memory and knowledge. Yet doing it is fun, useful, and provides a direct paradigm by which we can get into some tougher issues of inference and deduction. My feeling is that anyone who desires to attack *any* issue of comprehension should begin by analyzing the kinds of information he would need to assess the normality of a piece of conceptual information in the way outlined in this section.

## 6.8 STATE-DURATION INFERENCES AND THE FRAME PROBLEM

**sample:** Johnny was mad at Billy last week for breaking his toy.
Is Johnny still mad at Billy?
Probably not.

**sample:** John handed Mary a book a moment ago.
Is Mary still holding it?
Perhaps.

**sample:** John handed Mary a book yesterday.
Is Mary still holding it?
Almost certainly not.

**sample:** John started eating dinner at 6pm.
It's now 6:15. Is he finished?
Perhaps, but probably not.

In the conceptual memory, the temporal truth of a structure is not merely a function of that structure's presence or absence in memory, but rather is a function of explicit time relationships, time-related inferences *and* time normality knowledge. In other words, every structure (concepts and tokens included) has time dependencies. In order to determine the truth of a structure, X, at time T, much more work has to be done than simply asking whether X exists (disregarding time attributes) in memory.

Any model which deals with a constantly changing world is beset by the classic "frame" problem. Briefly, this problem is the following: given some piece of information which is true at time t1, under what conditions will this information be true at a future time, t2, and how and when should it be updated to reflect this passage of time? This problem is compounded when no piece of information is either true or false, but rather is "believed to some fuzzy degree." The frame problem is a very real issue for conceptual memory.

Consider for example the following sequence:

John handed Mary a book.
Is Mary holding the book?

Too simple a proof procedure which was sensitive only to explicitly stored time information would say "no" to this query, simply because knowing that Mary was holding the book at some *past* time (regardless of how near in the past) has no logical relation to Mary's holding the book now. The proof would simply fail, not realizing how close it came to locating the desired

information. In a conceptual memory, solutions to this aspect of the frame problem (keeping temporal knowledge up to date) are provided by *state-duration inferences.*

### 6.8.1 POSSIBLE APPROACHES

There are two basic approaches to the problem of knowing what is and is not true #NOW, based on what is known to have been true sometime in the past. The first approach is based on the philosophy that this updating should be constantly in progress as some sort of background monitor. While perhaps aesthetically pleasing because it keeps the memory "clean", it is hard to envision either a theoretical or practical means of implementing this type of scheme in a truly large memory. There are, in addition, strong psychological arguments against this method. People simply do not periodically scan through their entire memory updating all old facts!

A more realistic approach, both computationally and psychologically is to have the ability to *detect* information which may have become dated, and *update it before using it.* This has the same effect in theory if the detection and updating are done at a very low information retrieval level because then only temporally true information will then be "seen" by the processes which request the information retrieval (in particular, all sorts of inference molecules). This ability to detect and update information is based heavily on a knowledge of *normal durations* of states and protracted actions in the world.

### 6.8.2 NORMAL DURATIONS

Recall that with any proposition, P, whose truth has a temporal component (ie. is not a timeless truth) is stored at least one time proposition using one of the following predicates: TIME, TS, TF. The question is, what happens when some process needs to know whether P is (was, will be) true at some time T whose relation to one or more of P's explicitly stored time aspects is known? That is, if at 3pm we say "John is eating lunch", the memory will make the resultative inference that John becomes satiated. Then, if we come back at 4PM and ask the memory "Is John hungry now?", we would want the memory to answer "Probably not. He ate at 3, and entered a state of hunger satiation then, and this particular state typically lasts 4 or 5 hours in John's culture." Although this is a fairly sophisticated example, a fairly simple, and very general, mechanism underlies it.

This mechanism is one which converts a knowledge of normal durations for various states and actions into compatibility measures, ie. STRENGTHs. In the memory, knowledge about the normal duration of states and actions is organized around the time predicate N-molecules. Suppose P is an information structure with which no TF time feature is associated, but which has a TS time feature, as shown in the left of Fig. 6-24.

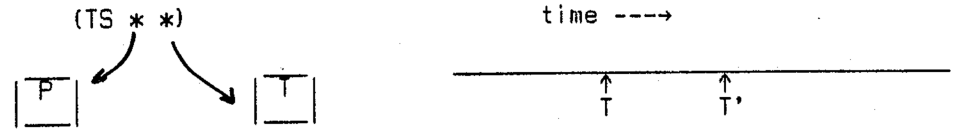



**Figure 6-24.**

The problem to be solved by a state duration inference in its simplest form is this: what is the approximate likelihood of P being true at T', where T' is some other time whose value is known? That is, if an inference process needs to know whether P is true at T', how can this informaton be related to the TS information of Fig. 6-24?

Clearly, if T' is *before* T, this information cannot be of much help! The interesting question in this case occurs when T' is *after* T. (Of course, if some other structure has only a TF, with no TS relation, the opposite is the case: the interesting T' is one which lies *before* T.) To ascertain P's likelihood at T', we must know something about the specific action or state P: how long do actions or states of this sort normally last. Notice that the answer is *always* highly dependent on the concepts involved in P, not just upon the conceptual predicate or even the form of the conceptual structure. For example, how long will it take an elephant to walk from San Francisco to Washington D.C.? How about if his right hind leg is broken? What if his trunk is sore? For what order of magnitude of time might a person continue to grasp a small object he is handed: what if it's a hot potato, what if it's a sentimental diamond ring from a departing lover. How long will Mary be gone shopping: where is the store, what does she need to buy? This is again a hint that to attempt to encode knowledge about the normal durations of actions and states in passive data structures might lead to undesirably complicated data structures. There are in genenal simply too many dimensions, and too many places where special heuristics are needed (for instance calculating the elephant's walking time from an estimate of his speed, and of the distance from San Fransisco to Washington D.C.) to attempt to encode duration information in passive patterns.

Instead, it would seem to be far more desirable to have an organized system of procedures which, when applied to a state or action structure, P, will return an "order of magnitude" estimate of P's duration. Where are such procedures to fit? The question "What is the normal duration of structure P likely to be" can be viewed simply as a statement of fact with part of the statement unspecified: (NDUR P X?). This leads naturally to the notion of an NDURation *specifier molecule* which is a collection of specifier atoms designed to handle all sorts of patterns.

The process of making a duration prediction for structure P would then consist of creating another structure (NDUR P X?) (where the X? here stands for another token in memory which has been marked UNSPECIFIED), then applying the NDUR specifier molecule to this new structure. A successful specification would result in the unspecified entity, X?, having been specified by some particular duration concept, D. If D is a "precise" duration concept (the time of a TV program for instance), then P is either true or not true at time T', depending on whether T' lies in the interval T to T+D. Otherwise, the duration is a fuzzy duration concept, and some more computation, which we will get to in a moment, must be performed. But notice that by handling the problem this way, a very desirable byproduct is produced: the specified NDURATION structure will remain and can be associated as one REASON behind any inferences which rely on P's truth at time T'. That is, the memory makes explicit what would otherwise have been an implicit duration inference. Hence, if we ask the memory why it believes John not to be hungry at 4pm, it can respond "Because he was satiated at 3pm, the normal duration of such a state is usually on the order of several hours, and it's only 4pm now."

Should some specifier atom within the NDUR specifier molecule successfully specify a fuzzy duration, it can attach whatever reasons it used to make its decision to the NDUR structure as this structure's REASONS.

What should become of this duration which has been specified for structure P? Suppose, for example, the NDUR specifier molecule specifies for P the fuzzy duration #ORDERHOUR. Although "on the order of an hour" here is indeed a fuzzy duration concept, in this context it has a very concrete interpretation: all "order-of" duration concepts can be mapped onto a compatibility, C (a measure of the degree to which some structure, P, can be believed), such that C is some function of the *difference* between the T associated with a TS, TIME or TF feature of P and the T', for which P's truth is being ascertained. Thus, for example, if the normal duration for

P can be specified as "order of an hour" and there is a (TS P T) relation stored, P's truth at T+50 minutes would be very likely, whereas P's truth at T+3 hours would be very unlikely.

### 6.8.3 MAPPING FUZZY DURATIONS ONTO COMPATIBILITIES

Assuming some specifier atom can determine a likely fuzzy duration for structure P, how does the fuzzy duration concept become a compatibility, based on T'-T? Associated as a property of every fuzzy duration concept in the memory is a function, F, which specifies the STRENGTH with which P might believed, based on the value of T'-T (recall that T is the known TS of P, T' is the time at which P's likelihood is being assessed). In general, such an F will be a continuous function of (T'-T), having the characteristic shape shown in Fig. 6-25. In general, it is necessary not only to have fuzzy duration concepts for all orders of magnitude, it is also necessary to have sharply-falling and gradually-falling versions of the same order of magnitude to characterize states which come to generally abrupt halts after some approximate duration as well as those whose likelihood trails off more gradually after some approximate duration.

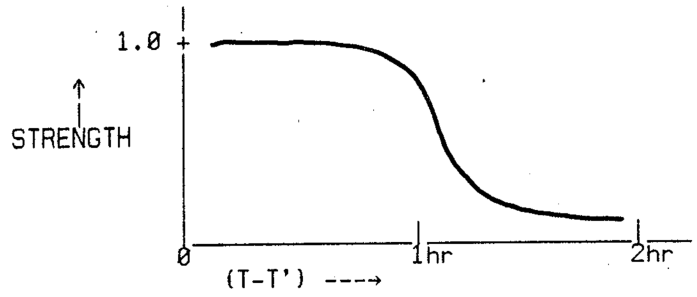



**Figure 6-25. A typical STRENGTH function for fuzzy duration #ORDERHOUR.**

In the memory, functions F which map T-T' onto a STRENGTH are implemented by simple LISP lists which store STRENGTHs as a *step function* of T'-T, rather than as a continuous one. This approach simplifies the problem of designing complicated continuous functions, and it makes the correspondence easier to read and adjust experimentally. Such a list is shown in Fig. 6-26 and takes the form of window-strength pairs. For fuzzy duration concept, D, this associated step function list is attached as the property "CHARF": the fuzzy duration's *characteristic* curve.

( (W1MAX S1) (W2MAX S2) (W3MAX S3) ... (WnMAX Sn) )

$0 \leq T'-T < W1MAX$ has strength S1
$W1MAX \leq T'-T < W2MAX$ has strength S2
...
$T'-T \geq WnMAX$ has strength 0

**Figure 6-26. The format of a fuzzy duration concept's step function.**

### 6.8.4 THE COMPLETE PROCESS

The complete process of a state-duration inference is the following: P is some memory structure with TS=T, and the likelihood of P still being true at time T' is desired. An (NDUR P X?) structure is created and the NDUR specifier molecule applied to it. The molecule performs tests to match P with some more general pattern with which a duration concept, D, is associated. If P is successfully matched, the X? is replaced by (the usually fuzzy) D, and whatever distinctive features of P figured into this decision are attached to the NDUR structure as REASONS. Next, the CHARF property for D is retrieved, and the quantity T'-T is calculated. A window in the CHARF step function -- with which a STRENGTH, S, is associated -- is then selected. This S is a measure of the likelihood that P is still true at T', and is the essence of the state duration inference. However, if some other inference molecule needs the NDUR structure as a REASON for its own inference, a (TIME P T') structure is created ("P exists at time T'"), and this structure is given STRENGTH=S and a REASONS property consisting of two structures: the original (TS P T) structure, and the newly-created (NDUR P D) structure.

There are two more loose ends which I have not mentioned. First, before creating the unspecified NDUR structure and calling a specifier molecule, the state duration inference process first looks to see that such an NDUR structure does not already exist from some previous assessment of P. If it does, the duration already specified can be reused. Second, if the assessed likelihood of P's truth at T' turns out to be extremely low (say, less than 0.10), the state duration inference process should generate the explicit terminating (TF P T') structure. That is, it should make explicit the fact that P has probably ceased to be true. This is the "automatic" updating of temporal aspects of memory structures mentioned at the outset.

Fig. 6-27 illustrates the process of a state duration inference.

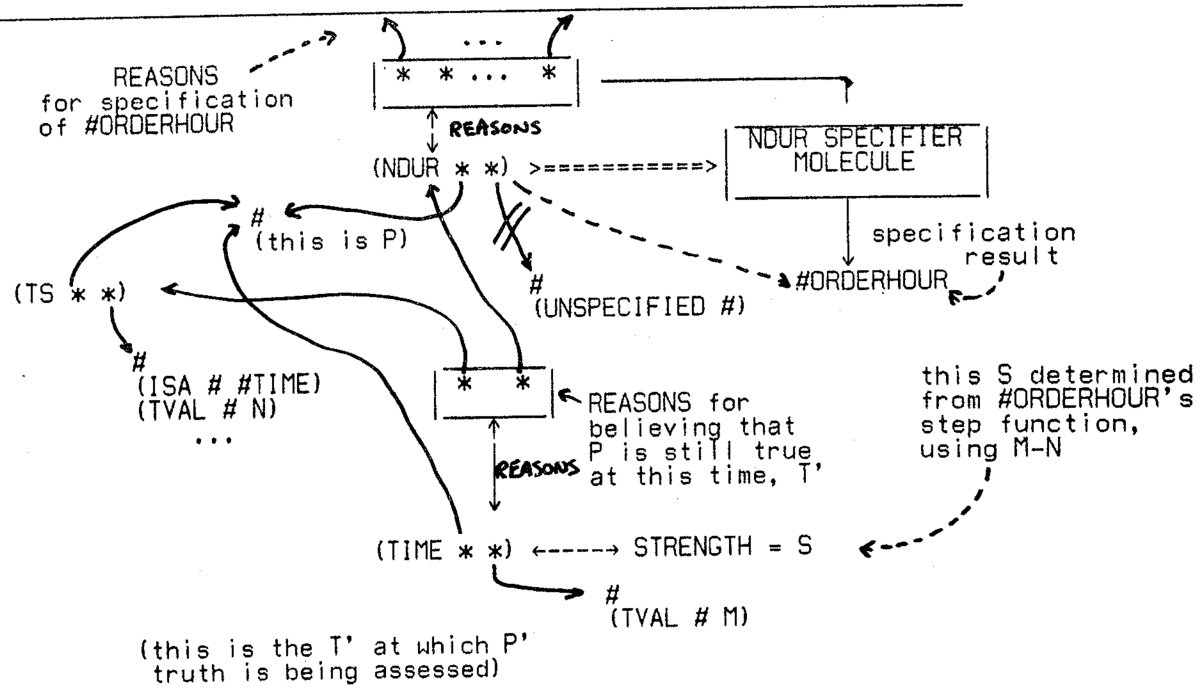



**Figure 6-27. The process of making a state duration inference.**

## 6.9 FEATURE AND SITUATION INFERENCES

**sample:** Andy's diaper is wet.
Andy is probably a baby.

**sample:** John's weathered face and gray beard intimidated Johnny.
John is probably an old man.

**sample:** Fred wagged his tail.
Fred is probably an animal of some sort.

**sample:** Fred bit the postman on the leg.
Fred must be a dog.

Most of the classes of conceptual inference I have discussed so far can be thought of as dealing with the more important facets of a large "motivation- action- state-cause- enablement-knowledge" complex: how each aspect, when allowed to react spontaneously to language meaning stimuli, contributes to understanding by expansion in inference space. Although this complex seems to provide the main architecture for processing the meaning content of utterances, there are other classes of inference which have far less structure, but which play very important roles

in the expansion process. These are typically inferences which are more closely related to the ideas of chapter 4 where I discussed some desirable word and feature "activation" capabilities, through which features of complex situations to which language alludes are drawn out. Of course, as I have argued, this is the purpose of *all* inferences. But I want here to illustrate a very large class of inferences which are founded more on simpler *associative* relationships between information in complex world patterns, rather than on the more rigid cause-effect, or action-enablement relationships of many of the previous sections.

There seem to be two distinct classes of these kinds of inference which are based more on "association" than on "logic": (a) those which predict new conceptual features of concepts and tokens, based on their old conceptual features and upon the situations in which they appear, and (b) those which make explicit features of a pattern in the world -- a situation -- to which some other information alludes. I will call inferences in class (a) *feature inferences*, those in class (b), *situation inferences.*

### 6.9.1 FEATURE INFERENCES

A feature inference draws out new features of a token or concept from existing (known) features of that token or concept. That is, by knowing a small number of "distinctive" features of an entity, it is often possible to have the ability to predict (make explicit) more features of that entity which are commonly associated with those already known. This is an extremely simple idea, but it is something at which human language users are quite facile, and it provides an important source of expansion in inference space. In section 8.1 there is a computer example where a feature inference plays a key role in the memory's ability to understand a simple utterance.

I will illustrate the idea of feature inference using that example: "Andy's diaper is wet". The underlying meaning graph of this utterance is shown in Fig. 6-28.

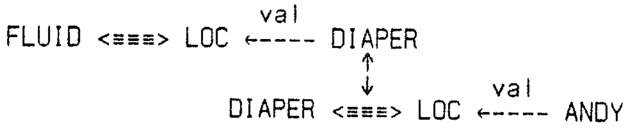



**Figure 6-28.**

Hearing this, we automatically conclude that Andy is a very young person. Although it is conceivable that an intelligent analyzer, while constructing this meaning graph, could be equipped with special heuristics to realize that "Andy" is probably an infant, it is not clear how general or desirable such an ability would be, or exactly how it would be done. On the other hand, tasks of this nature have a very natural solution in the memory where such information can be quite useful. It would be acceptable for the analyzer to render the descriptive sets shown in Fig. 6-29 for the objects involved in this conceptualization, as long as the memory is prepared to extend, refine or correct them subsequently by applying its broader knowledge of the world to make further predictions. But exactly how and when can this knowledge be called into play?

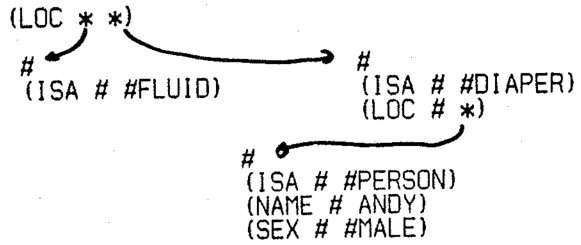



**Figure 6-29. "Andy's diaper is wet."**

To answer this, it is important to recall the process by which the memory extracts *all* the information from each conceptual input. This is especially relevant to feature inferences because most features of objects are communicated only incidentally in utterances, through RELative conceptual links. In this example, the subpropositions MEMORY extracts which are relevant here are twofold:

1. (LOC C4516 C2308)
2. (LOC C2308 C7211)

assuming C4516 is some fluid, C2308 is some diaper
which is located on C7211, some person named Andy

**Figure 6-30.**

The important point is that *both* of these subproposition structures will be allowed to expand in inference space. But in particular, doing so will permit (2) to give rise to the feature

inference: "C7211 is not only a #PERSON, because a diaper is located on it, it is likely to be a very *young* person". In other words, the new feature inference (AGE C7211 #ORDERMONTHS) can be made, with REASONS being the fact that (LOC C2308 C7211). By noticing one feature of C7211, the LOC inference molecule, when applied to structure (2), can predict another feature of C7211 on the basis of a common association between the old and new features. This is a typical feature inference.

The four samples above all illustrate similar feature inferences. Three interesting questions arise, though: what are the differences between feature inferences and (a) specification inferences, (b) certain forms of intrinsic enabling inferences, and (c) what is the relation between the PART inference in the example above and the process of relation pathfinding which predicted the PART relation in the first place.

The first question might be phrased this way: couldn't the fact that Andy's exact nature was unknown have been explicitly marked by the analyzer, to be filled in by a specification inference? The answer is "Yes, probably". Clearly, the descriptive set for this "Andy" could have been marked as an UNSPECIFIED, later to be detected by a specifier molecule. But there are three reasons why the feature inference approach is more convenient. First, the conceptual analyzer tries at all times to make "best guesses". In this case, since its best guess predicts that "Andy" refers to a male person named Andy, *for the analyzer's purposes*, "Andy" *IS* specified -- age information is simply not relevant to the process of constructing the meaing graph of Fig. 6-28. In general, the analyzer will not be expected to be aware that much more about Andy in in general inferrable. That is, how, in general, is the analyzer to know what is and is not ultimately "fully specified" in cases such as this? Second, if the nature of "Andy" in this example were to be solved by a specifier molecule, that molecule would in general become extremely complex, having to have special heuristics for searching for telltale existing features already known about the entity it is further specifying. A feature inference, on the other hand comes about naturally by the process of inferencing on all information in a meaning graph "simultaneously". Third (and most important), more often than not an entity will be "fully" specified, in that it uniquely identifies a token in memory, yet we still want the ability to collect more features of it. Specifier molecules are out of the picture in such cases, so that feature inferences can be quite distinct from them. Thus, the process of *specification* is viewed more as a means for inferring the

identity of truly *missing* information, rather than as a process which collects more and more conceptual features of an entity which might already be fairly richly endowed with features.

The second question -- what is the difference between certain types of feature inferences and certain types of intrinsic enabling inferences -- is really a non-issue. The answer is that they are frequently the same sort of thing, but feature inferences are more general. For instance, if the memory hears "John ate a gronk", it will certainly make the intrinsic enabling inference that a gronk, whatever it is, is capable of being eaten: (EDIBLE #GRONK) becomes a new feature of this concept with name "GRONK" (there is still no information about what happens to you if you eat one -- only that one can be ingested). In other words, what I have called a feature inference, might also arise via an enabling inference. But this is unimportant. What is more important is that feature inferences are more general: they are capable of inferring new features, even where no enablement is implied. The samples above illustrate this.

The third question in this: if, by the relation pathfinding technique, the memory can discover the LOC relationship between "Andy" and "diaper" in the first place, why can't the information that Andy is very young be inferred at that point? Again, we must say that it could, but that it is more naturally done later on. The problem in this case is that it is convenient to classify "*diaper*" as an article of clothing whose relation to a person is the same as all other articles of clothing. The path which the relation pathfinding process yields will serve only to relate #DIAPER with #PERSON *as a thing to be worn.* Nowhere is age involved in the clothes-person path, and rightly so: a person of any age can wear a diaper. It is only a (highly likely) *inference* that, if some person is wearing one, he is very young. By recognizing that this is just another inference -- that is, by implementing it in an inference molecule -- it can be made quite sensitive to unusual contexts. In this example, for instance, before generating the (AGE X #ORDERMONTHS) inference, the LOC inference atom can test for special information about X which could affect the inference (for example, what if X a paralyzed adult).

### 6.9.2 SITUATION INFERENCES

**sample:** Mary is going to a masquerade.
Mary is probably wearing a costume.

**sample:** John is asleep on the subway.
He is probably sitting slouched over on the seat.

**sample:** John is picnicing in a meadow.
John probably smells flowers, and sees grass.
He might be stretched out, relaxing.

It is not difficult to extend this notion of a feature inference of a concept or token to feature inferences of *entire situations*. That is, just as some feature(s) of an entity can lead to other features of that entity which frequently co-occur with X, so can one conceptualization serve to draw out an entire situation which consists of many other conceptual patterns. I will simply illustrate this idea here with some examples. In practice, this kind of inference is too unstructured to perform extensively in a practical way. But it suggests an interesting topic for further research.

By drawing out implicit information and features of entities, feature inferences have some hard-to-capture, but intriguing, relationship with notions of "visual imagery" and "iconic memory". That is, they seem in some sense to be capturing, in a discrete, propositional form, something of what it means to "imagine a situation" or an object.

Modulo some more directed research into this type of inference, it might at some point prove not unreasonable to conjecture that visual imagery and iconic memory are nothing more than this drawing out -- this activation -- of features of objects and implied information in a situation. This is perhaps not a new idea, but it has a tangible expression in the memory formalism, and hence is a concrete conjecture to make in this context.

## 6.10 UTTERANCE INTENTION INFERENCES

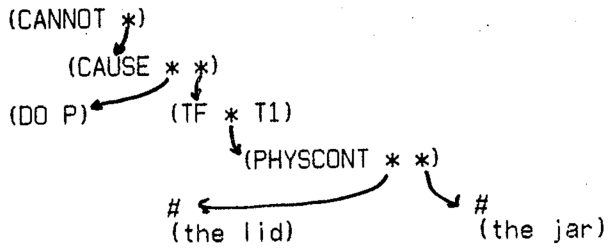
**sample:** Mary couldn't jump the fence.
Why did she want to?

**sample:** Don't eat green gronks.
OK. But you mean I can eat other kinds of gronks, right?

I have for the most part avoided inferences which lie more in the domain of conversation models. There are many classes of inferences bound up with the speaker's intention for saying something, or saying something with a particular emphasis, which bear no immediate logical

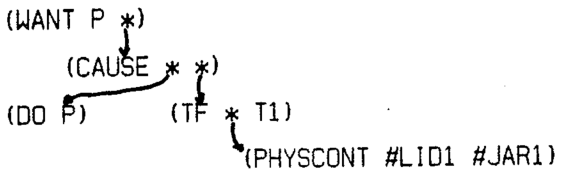
relation to what he has said, but which people clearly employ most fruitfully. While these inferences are connected with many specific conventions of language *use*, they are nonetheless real conceptual inferences. They simply exist at the "higher" level, within a model of conversation. I will call inferences in this class *utterance intention* inferences, and abbreviate this by "U-intent". In this section, I will point out some simple instances of this class.

What are some examples of these mysterious inferences? Two of them are based on very simple patterns: (CANNOT X) and (NOT X). Suppose we hear P say "I can't get the lid off this jar.", ie. P has just communicated a (CANNOT X) form to us which is roughly represented as:




(P is unable to cause the lid to cease to be in physical contact with the jar). What did P mean by this? That is, what effect does P believe it will have on the hearer? In everyday use, it can safely be construed as a request for help. But what inference makes this known to the hearer?

The answer in this case appears to lie in a very simple inference which, once generated, will lead to other inferences and actions dependent upon the context. This inference is that P WANTs X to be the case. In this example, this leads to the structure




This structure has the interpretation: P wants some causal structure to be true; P must therefore also want the results of the causal if it were to be true; P therefore wants (DO P) and (TF (PHYSCONT #LID1 #JAR1) T1). Knowing both the CANNOT and the WANT, the hearer might perform some action that would help P, having applied some other belief pattern that when a friend can't achieve something he desires, he needs help.

There seems to be one additional criterion for this inference type, however. That is that the CANNOT structure have a time associated with it- that it not be a timeless statement of fact. The interpretation of a time associated with a CANNOT is that its associated action was *attempted*, but was unsuccessful in achieving its probable consequences. Put this way, this kind of inference's utility can be viewed as setting the stage for motivational inferences from the unsuccessful action. As an example of why timeless CANNOT structures should not give rise to a U-intent inference, consider the statement "Ralph can't swim." Ralph simply never learned to swim, and it is not implied that he in fact wants to be able to swim. In other words, the implication that he has ever attempted to swim at time X is not present.

There are many illustrations of how this type of inference can serve as a critical link in understanding:

1. John was unable to start the fire.
2. Bill couldn't find his keys yesterday.
3. Rita wasn't able to go to the fair.
4. Pete prevented Sally from climbing the flagpole.
5. John doesn't seem to be able to sell his car.

### 6.10.1 OTHER EXAMPLES OF U-INTENT INFERENCES

Inferences which can stem purely from the way in which a thought is *phrased*, or from what information the speaker decided to *include* constitute a seemingly limitless class. I will not go very deeply into it here, but merely point out that there is a wide-open domain for research. I will briefly discuss two of the more obvious and useful types of inferences in this U-intent class.

The first is sensitive to possible causality relationships between an actor's action and *extra attributional information* in the sentence concerning an object involved in that action. For example, Linda said to Chuck the other day

I threw out the rotten part of the fig.

Contrast this with the similar, but simpler thought "I threw out part of the fig." The first sentence

carried a referentially nonessential attribution about the part of the fig which was discarded; that is, in the context of this utterance, it would have been quite possible to identify the fig and its "part" even without knowing anything else about it, as in the second sentence. Since the extra information was not included to identify *which* fig and part were discarded, why was the attribution about the part's rottenness included? Apparently, Linda had included this extra attribution to indicate that she wanted Chuck to infer that the part was discarded *because* it was rotten.

There seems to be a mini-principle here: when more information is communicated about an object than is requisite for the referential disambiguation of that object, the extra attribution might stand a chance of being *causally* related to the action involving that object. In this particular example, the extra attribution is somewhat redundant, since we normally assume that fruit is thrown out because it is rotten. However, in the sentence "I lambasted the man who was standing in my way", this extra attribution serves to specify what would otherwise be an unexplained cause of the lambasting. This language use of the REL link to communicate underlying causals is illustrated in Fig. 6-31.

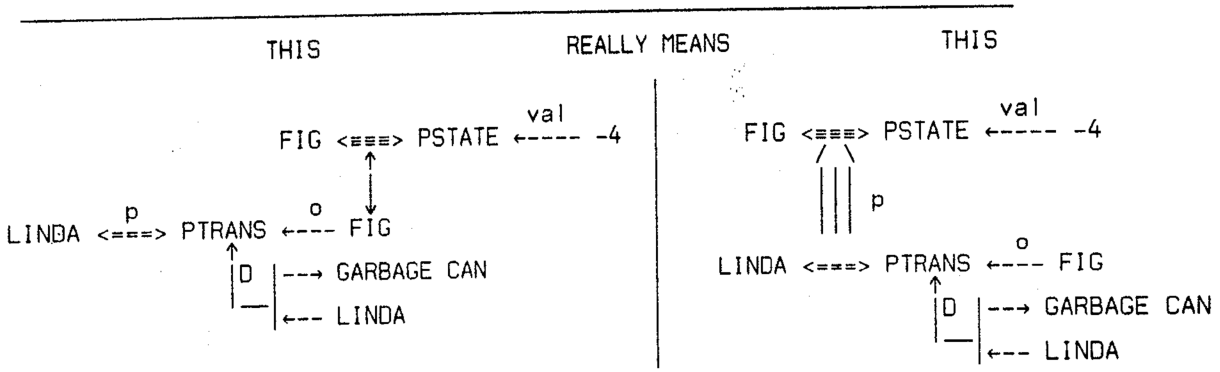



**Figure 6-31. An underlying causal communicated conceptually by a REL link.**

The second representative of this large class of U-intent inference types also concerns the reasons a speaker chooses to include more information about an object than is necessary for referential distinction. However, from this type of additional attribution, it is often possible to infer attributes of the object or concept itself, rather than causality relationships. For example,

someone might tell you "Don't eat gronks." If you're friends, you can safely infer that the existence of a gronk in one's stomach probably leads to bad things; in other words, gronks are inedible. If, however, he says "Don't eat green gronks.", you may infer with nearly equal safety that gronks are OK to eat, it is only a *green* gronk which will do you in! Because of one additional and seemingly inconsequential attribution, you may infer that gronks are edible. This could save your life on a desert island some day!

Both of these are very general rules. To implement them in a theory of conceptual memory would obviously require a much more detailed analysis, especially of when special cases and circumstances should override these normal U-INTENT inferences. They have been cursorily described here only to represent a large "missing" class of inferences in the current theory and implementation of conceptual memory.

## U-INTENT INFERENCE COMPUTER EXAMPLE

This example illustrates the usefulness of one type of U-intent inference: inferring (WANT P X) from (CANNOT P X). The input sentence is "John was unable to get an aspirin." MEMORY will, in the absence of context, predict that John wanted an aspirin, and this inference opens up an otherwise inaccessible line of inference which terminates in the presumption that John underwent a negative change on his health scale. Since, having generated this inference, MEMORY has no means of determining the cause of the negative change, a reasonable question for MEMORY to ask is drawn out by this line of inferencing: "What caused John to be sick?"

```
JOHN WAS UNABLE TO GET AN ASPIRIN
```

Here MEMORY reads the input sentence. Its analyzed representation is shown to the left.

```
((ACTOR (JOHN) <=> (*ATRANS*) OBJECT
(ASPIRIN REF (*A*)) FROM (*PERSON*
SPEC (*U*)) TO (JOHN)) TIME (TIM01)
MODE ((*CANNOT*)))

TIM00: ((VAL T-0))
TIM01: ((BEFORE TIM00 X))
```

```
((CANNOT ((*ATRANS* (#JOHN1) (C0001)
(C0005) (#JOHN1)) (TIME _ (C0008)))))

C0013
```

This is the partially integrated MEMORY structure. C0001 is the token of aspirin, C0005 is the person from whom John could not get C0001, C0008 is some past time. Notice that C0005 has arisen as an unspecified concept. It will in the absence of context in this example be filled in as some instance of a #DRUGSTORE.

```
STARTING INFERENCE QUEUE:
((X 1.0 C0013))


APPLYING INF MOLECULE CANNOT TO C0013:
(CANNOT (*ATRANS* #JOHN1 C0001 C0005
          #JOHN1))

ABOUT TO APPLY ⊗CANNOT1 TO C0013
INFERRING: (WANT #JOHN1 C0011)
   ALSO GENERATING: (TIME C0018 C0016)

        ......

((P 1.0 C0018) (M 1.0 C0026)
(M 1.0 C0029) (C 1.0 C0039)
(M 1.0 C0048) (M 1.0 C0051)
(M 1.0 C0058) (C 1.0 C0063))
```

The starting proposition relevant to this example is simply the main one. All others have been suppressed.

The first inference to occur from this structure is that John probably wanted to *ATRANS* himself an aspirin. The CANNOT inference molecule which generates this inference first checks that the CANNOT event has a time associated with it (ie. this implies that John attempted the *ATRANS* action. C0008 is found as the time.

Other inferences are made possible from this CANNOT → WANT inference. These are shown at the left, together with their type (peripheral, motivational, causative).

---

```
C0013: (CANNOT C0011)
(CANNOT (*ATRANS* #JOHN1 C0001 C0005
          #JOHN1))
RECENCY: 5766
TRUTH: T, STRENGTH: 1.0
OFFSPRING:
   C0019: (TIME C0018 C0016)
   C0018: (WANT #JOHN1 C0011)
ISEEN: (⊗CANNOT1)
```

This is the original structure as it appears after inferencing.

Here, the crucial WANT inference has been recorded as its sole offspring.

---

This is the inferred structure after inferencing

```
C0018: (WANT #JOHN1 C0011)
(WANT #JOHN1 (*ATRANS* #JOHN1 C0001
                C0005 #JOHN1))
ASET:
   C0031: (CAUSE C0029 #)
   C0028: (CAUSE C0026 #)
   C0019: (TIME # C0016)
RECENCY: 6816
TRUTH: T, STRENGTH: 0.89999999
REASONS:
   C0013: (CANNOT C0011)
OFFSPRING:
   C0031: (CAUSE C0029 C0018)
   C0030: (TIME C0029 C0016)
   C0029: (WANT #JOHN1 C0022)
   C0028: (CAUSE C0026 C0018)
   C0027: (TIME C0026 C0016)
   C0026: (WANT #JOHN1 C0021)
ISEEN: (⊗POSTSCAN)
```

Notice how it has lead to other inferences.

```
(P 1.0 C0018)
(WANT #JOHN1 (*ATRANS* #JOHN1 C0001
 C0005 #JOHN1))
        C0031: (CAUSE C0029 C0018)
        C0028: (CAUSE C0026 C0018)
        C0019: (TIME C0018 C0016)
```

This is the resulting set of inferences from the original structure: John must have wanted to get an aspirin.

```
(M 1.0 C0026)
(WANT #JOHN1 (TF (*POSS* C0001 C0005)
 C0008))
        C0028: (CAUSE C0026 C0018)
        C0027: (TIME C0026 C0016)
```

John must have wanted to *ATRANS* an aspirin to himself because of the predictable results of that action. One of these is that whoever he tried to get it from would cease having it.

```
(M 1.0 C0029)
(WANT #JOHN1 (*POSS* C0001 #JOHN1))
        C0041: (CAUSE C0039 C0029)
        C0031: (CAUSE C0029 C0018)
        C0030: (TIME C0029 C0016)
```

Another result of *ATRANS*ing would be that John begins possessing the aspirin.

```
(C 1.0 C0039)
(WANT #JOHN1 (*INGEST* #JOHN1 C0001
 C0032 C0035))
        C0053: (CAUSE C0051 C0039)
        C0050: (CAUSE C0048 C0039)
        C0041: (CAUSE C0039 C0029)
        C0040: (TIME C0039 C0016)
```

The probable reason why John might want to possess an aspirin is to use it in its normal function. The normal function of an aspirin is found to be that it be ingested.

```
(M 1.0 C0048)
(WANT #JOHN1 (*LOC* C0001 C0035))
        C0060: (CAUSE C0058 C0048)
        C0050: (CAUSE C0048 C0039)
        C0049: (TIME C0048 C0016)
```

The reason John probably wants to ingest an aspirin is for its predictable consequences. One predictable consequence is that the aspirin begins being located in John's insides (C0035).

```
(M 1.0 C0051)
(WANT #JOHN1 (TF C0001 C0044))
        C0053: (CAUSE C0051 C0039)
        C0052: (TIME C0051 C0016)
```

Another predictable consequence of the ingesting is that the aspirin ceases to exist. John might therefore possibly want this. Hopefully, heuristics in the evaluator demote this inference.

```
(M 1.0 C0058)
(WANT #JOHN1 (POSCHANGE #JOHN1 #HEALTH))
        C0065: (CAUSE C0063 C0058)
        C0060: (CAUSE C0058 C0048)
        C0059: (TIME C0058 C0016)
```

A predictable result of a medicine being located in someone's insides is that the person will undergo a positive change on his health scale. Therefore, it can be inferred that John wants such a positive change to occur.

```
(C 1.0 C0063)
(NEGCHANGE #JOHN1 #HEALTH)
        C0068: (CAUSE C0063 C0066)
        C0065: (CAUSE C0063 C0058)
        C0064: (TIME C0063 C0061)
```

But if a person wants a positive change on some scale, it must be because he had previously suffered a negative change on that scale. During the post-inferencing scan for missing causality, this inference (possibly among others) will be detected as lacking a causal explanation. This gives rise to the potential question shown to the left: "What caused John's negative change in health?"

```
((CON (*?*) <=> ((ACTOR (JOHN) <=>F
(*HEALTH*) <=>T (*HEALTH*)) INC (-2)
TIME (C0061))))
```

## 6.11 SOME THOUGHTS ABOUT COMPREHENSIVENESS

Having read this and the previous chapter on inference types, you may have framed the following question: "what is the relative scope of these inferences?" That is, of all the types of inference and deductive mechanisms people use to understand language, what portion can be accounted for by this system of classification? How comprehensive is this catalog of inference types?

Any answer to this question is bound to be speculative. In addition, there is an inherent fuzziness concerning whether some particular inference is of type X or type Y, or even whether it can be viewed as one type in one context and another type in another context. What *can* be said concerning comprehensiveness, however, is this: I believe I have attacked the central core of the human inference ability. By doing so, the real success lies, not in the percentage of inference capability accounted for by this classification, nor in its variety, but rather in the demonstrations of how inferences interact among themselves and with language. Certainly there are other classes of inference which have not even been alluded to in these two past chapters. Section 6.10 suggested one such class, and there are many which are more logically a part of a theory of conversation. To attempt to discover and classify by function all types of inference a human language user employs is a noble goal indeed, and it needs to be done. However, it will be encyclopedic, and this is not my immediate goal!

Instead, we have the beginnings of a synthetic and computationally effective theory for modeling the abstract flow of information in the human brain as it concerns language understanding. We must put into focus the larger issues which concern the *utility* of an inference class rather than its *descriptive* ability. In this way, the stage has been set for integrating more and more classes of inference -- based upon their efficacy to the understanding process -- into the larger picture of information flow in response to language stimuli.

## 6.12 SOME THOUGHTS ABOUT PRACTICALITIES

In a sense, I have constructed a monster. If all the inference powers I have described these last two chapters were unleashed at once, it would not be unrealistic to expect 500 or 1000 inferences to arise from each utterance. This is invigorating, because it is the essence of the theory: that each utterance expands into a very broad spectrum of surrounding information, and this spectrum interacts with the spectra of other utterances. When we consider orders of magnitude, 1000 inferences, viewed as a wave of activity in a parallel neural net of over ten billion nodes might be quite insignificant. While the program *can* perform in this "all-at-once" mode, it will often require 5-10 minutes of real time on a day when the system is not too heavily loaded. This is obviously unacceptable on a real-time basis, and it makes debugging very tedious.

The theory is no less desirable because of this. What I envision ultimately is a system of genuinely parallel processes, which are based on the various reference and inference mechanisms, and which all work cooperatively and simultaneously on each utterance. This is a big order in practice, but it is an exciting goal which we could set out toward today on a small network of existing "mini" computers: one mini to determine referents, another to perform state-duration inferences at the lowest level of information lookup, another to generate action prediction inferences from each input, another to generate enabling inferences, another to maintain RECENCY and TOUCHED tags, and so on.

Also, the theory is in immediate need of a good, effective theory of forgetting. Clearly, it is neither psychologically real, nor practical, to retain all the 500 or 1000 inferences which can arise from each simple utterance. This is particularly true, since many of them represent a calculated waste. Those, however, which have been successful in enrichening the memory's connectivity -- those which have made interesting contacts with other memory structures, or those which have lead to interesting contradictions -- should remain as the net effect of the utterance in the context in which it was perceived. This forgetting function might also be conveniently viewed as a parallel process which runs constantly "beside" the memory as it generates the inferences I have described.

But for the immediate future, the progress will lie in upgrading the current program to the present state of the theory, and in encoding many more inferences and data about the world in

general. Until this is done, we would perhaps only be skirting the tough issues by getting involved in parallel processing.

In the next chapter I will cover some of the programming topics which have been defined by the inference capabilities I have described.

# CHAPTER 7

## THE INFERENCE CONTROL STRUCTURE, THE STRUCTURE MERGER, AND OTHER ASPECTS OF THE PROGRAM

This chapter is devoted mainly to the memory's inference mechanism, which has been referenced throughout the two previous chapters, but not yet explained in programming terms. In particular, the major topics to be covered are the inference monitor and evaluator, inference molecules, and the structure merger.

## 7.1 IMPLEMENTING THE INFERENCE CAPABILITY

**How are the inference capabilities described in chapters 5 and 6 implemented? What is the nature of the inference control structure?**

There are three parts to this question: the first one concerns the familiar dilemma of whether to use data structures or program structures for what will eventually become a very sophisticated pattern matching process. The second part concerns the inference *control structure*, and the third concerns the nature of an *individual inference*.

### 7.1.1 DATA VS. PROGRAM VS. DATA VS. PROGRAM VS. ...

What is the difference between information which is stored as "data" and information which is stored as "program"? I use sneer quotes here because a philosopher would perhaps tell us there is no ultimate distinction between the two: he can perhaps always argue that a program is simply a data structure which is interpreted by some higher process, and hence that it is simply data. Alternatively, he can view the "program" which interprets what he chooses to call "data" as "some higher process", then it is no longer data, but a program written in the language of this interpreter. So why pose the question? There are genuine pragmatic distinctions between program and data in a pattern matching system with requirements such as those we have defined. At some point, *something* needs to cause changes in the memory. Whatever this is at the time, it must be "process" rather than "data".

The main question is whether we want to view pattern matching as a very general higher level process which attempts to compare two *data* patterns with one-another, or whether it is more desirable to view pattern matching as a collection of many very specific, lower-level processes which attempt to *match themselves* to one one specific data pattern. These two alternatives are abstractly illustrated in Fig. 7-1.

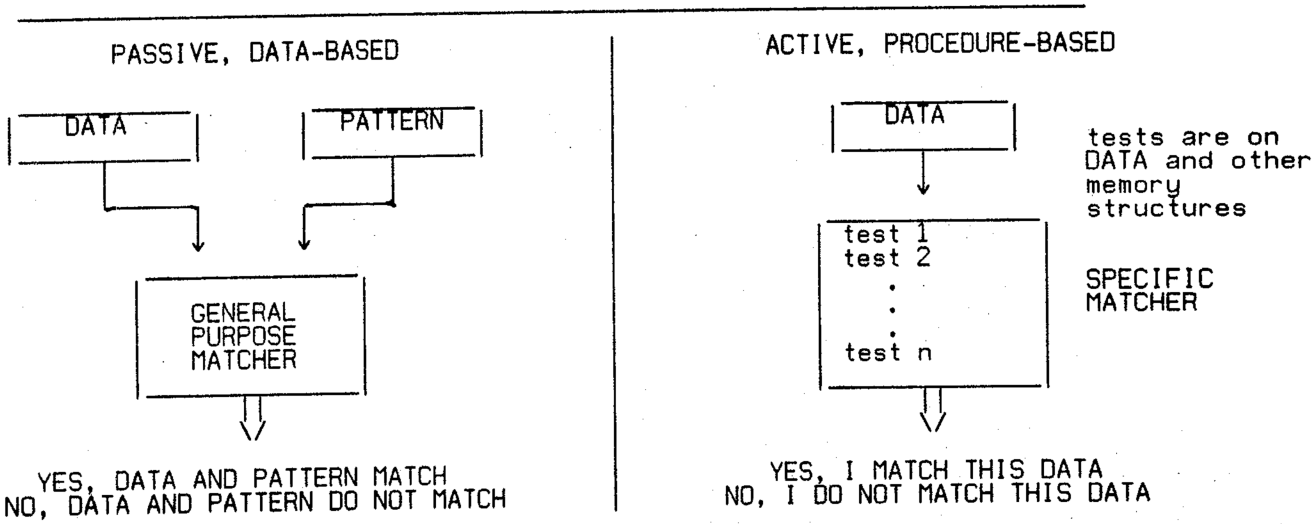



**Figure 7-1. "Passive" vs. "active" pattern matching.**

A system which employs data-based matching is often termed "declarative", whereas one which relies mainly upon active, program matching is often termed "procedural". There are arguments on both sides of the fence about declarative versus procedural pattern matching:

1. Data-based pattern matching provides a clear-cut distinction between that which is timewise relatively static process, and that which is timewise relatively dynamic data.
2. There is a greater need to standardize data structures which store data than to standardize those which store process. Patterns encoded as *process* will by definition already be subject to the requirements in form of the programming language in which they are written.
3. Data-based matching, with suitable formatting routines, permits easy extension of a data base. This includes the addition of new rules.
4. Program-based matching is as fast and to-the-point as the pattern it is matching will allow. Case-specific heuristics are more easily encoded from pattern to pattern in a program-based matching system. Data-based methods are inherently slower and harder to organize in optimal ways, on today's computers. Whereas a program-based matching system can look exactly at what is relevant from pattern to pattern, and can perform the matching in different orders for different patterns, based on special-case heuristics, heuristics which tell a data-based matcher which is relevant and what is a good order of matching are elusive, awkward and hard to encode in data structures.

5. In data-based matching, since the patterns are data themselves, they can be referenced just as any other data. Also, to communicate new patterns and rules to a data-based system is more straightforward than to communicate new patterns and rules to a program-based matcher. This is because, in the program-based matcher, a new process must be synthesized which will perform the matching required to detect the new pattern. However, if they are formatted in predictable ways, programs can be treated as data when necessary. This permits rule extensions, but methods for building programs from rules are not well understood yet.

6. Program-based matching allows a convenient means of escape to arbitrary subroutines during the match with no interruption. Although Tesler, Enea and Smith [T2] have demonstrated how to approach this problem in a data-based matcher, the solution is part of a very sophisticated system.

7. Program matching is quite straightforward when variable binding must occur to points in a matched pattern. In data-dased matching, special, often tedious, provisions must be made in order to extract features from the matched data as a byproduct of the match.

One other practical drawback of data-based matching relative to the development of a large memory relates to (2) above. In a data-based matching system, some fairly rigid and comprehensive data format must be decided upon early in the research *before it is fully known what the potentials of the system should be.* It is much easier at this stage of development to write programs, keeping an eye out for recurring patterns of processing, than it is to define an all-encompassing data format for a data-based matcher. We must tolerate sloppy, cut-and-pasted processes for the time being, and this is an admission that we simply don't know enough yet to commit ourselves. Once it is discovered with a degree of confidence what needs to be done, we can worry about encoding it in a pleasing homogeneous data formalism. But until then, we should not compound the problem by constraining ourselves to what will probably turn out to be inadequate or unwise choices of data structures for the dynamic (inference) processes in the memory.

From the bias evident in these pros and cons, and in the previous chapters, it should be clear that program-based pattern matching has been used wherever possible in the memory, and the bulk of the pattern matching occurs by active inference procedures.

## 7.2 THE INFERENCE CONTROL STRUCTURE

The inference monitor is a LISP procedure called INFERENCES. It, in conjunction with specifier and normality molecules, is the supervising process by which all the various types of conceptual inferences I have discussed are generated. The monitor consists logically of the following components:

1. *queues* of memory structures which have undergone inferencing (!INFERENCES), and of memory structures which are awaiting inferencing (!NEWINFS)
2. the *basic monitor*, which maintains these queues, locates applicable inference-generating packages of procedures, called *inference molecules*, and applies them to successive structures on the queue
3. the *structure generator* which helps inference molecules generate inferences. This is a very simple interface function which actually creates new memory structures and adds them to the inference queue upon demand from specific inference molecules. As we will see, the structure generator is sensitive to the theoretical type of each inference it is requested to generate a memory structure.
4. the *inference evaluator*, which looks at each inference after it is generated. The evaluator detects contradictions, reorders the inference queue, and requests *merges* of identical and similar structures which have been generated from independent sources. It is the principal agent by which new points of contact in inference space are recognized.
5. the *structure merger* which physically constructs the new points of contact by merging two memory structures into one

The reaction to each utterance involves several iterative passes through the inference monitor and reference-establisher. I am about to describe the character of the *first pass through the inference monitor*. This will then be extended to multiple passes which realize the important inference-reference interaction.

### 7.2.1 THE BASIC MONITOR

#### 7.2.1.1 THE QUEUES

The main inference queue is simply a top-level list, !INFERENCES, of pointers to memory structures which represent information from which inferences are desired. The *starting* inference queue for each utterance, U, consists of the set of subpropositions which the internalization process extracted from U's meaning graph. In the example of Fig. 4-16 for instance, three were extracted, so that the starting queue would be as shown in Fig. 7-2. This

queue will grow in length as new inferences are made, and will eventually end up as a (typically) very long list of memory structures which were inferred as a reaction to U.

The inference queue is a temporary construction for each utterance, U: it is reset to NIL before inferencing from each U is begun. The lasting tangible effects of the utterance on the memory are the actual structures which result from inferencing and structure merging; the inference queue is simply a temporary record of the memory structures currently associated with the utterance under inferencing.

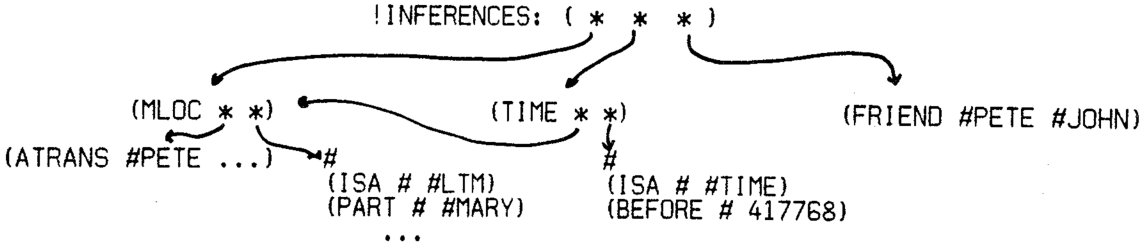



**Figure 7-2. A typical starting inference queue.**

The basic monitor implements a breadth-first inference expansion of this starting queue of structures, S1,...,Sn. It begins with the first structure, S1, on the queue, and proceeds to the right, performing the operation I am about to describe on each Si in turn. The inferences which arise from S1,...,Sn are collected on another temporary queue, !NEWINFS. When S1,...,Sn have been exhausted, !NEWINFS is appended to the main queue, !!INFERENCES, and !NEWINFS becomes the new queue to expand. Thus, each !NEWINFS represents the next level in the breadth-first expansion, and at the end, !!INFERENCES, will have collected every inference generated by this level-by-level expansion. The relationship of these two queues is illustrated in Fig. 7-3.

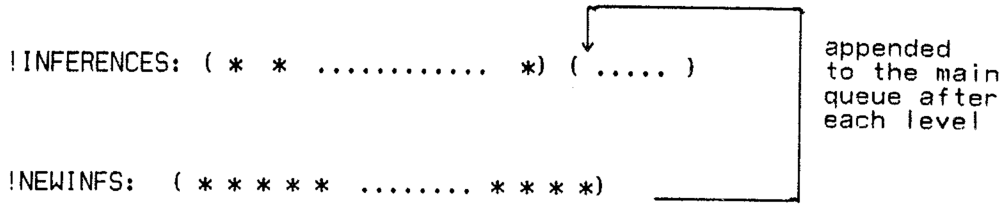



**Figure 7-3. The queues which collect the breadth-first expanded inferences.**

### 7.2.1.2 APPLYING INFERENCE MOLECULES TO EACH STRUCTURE ON THE QUEUE

For each Si to be expanded by conceptual inference, the following occurs. First, if any UNSPECIFIED entities are detected in Si, the appropriate specifier molecule is applied to it. Next, the monitor examines Si's conceptual predicate, Pi, and retrieves the executable inference molecule which is associated with Pi as the property IPROG(Pi). The relationships between the queue, Si, Pi, and IPROG(Pi) are shown in Fig. 7-4. The monitor then applies this inference molecule to Si, and the molecule will ask many questions of Si, as characterized in the right half of Fig. 7-1, generating inferences of many theoretical types from Si in the process.

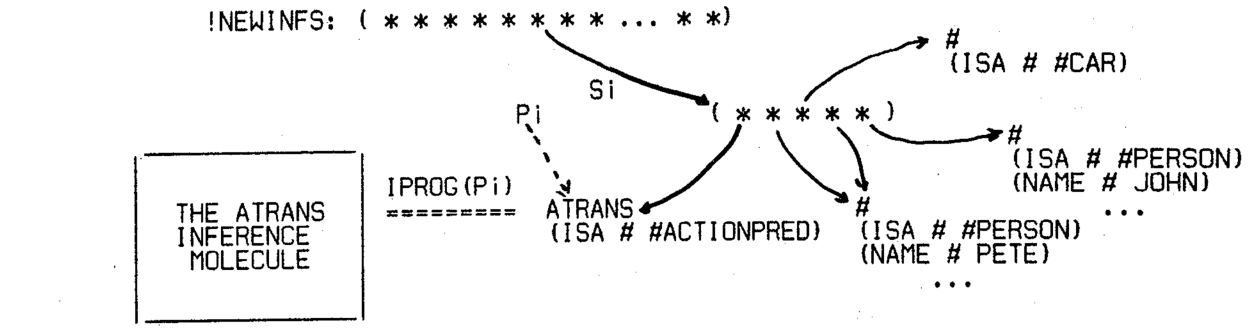



**Figure 7-4. Relationships between the queue, the structure, the predicate, and the inference molecule.**

To apply IPROG(Pi) to Si, the monitor first does a small amount of bookkeeping by locating and assembling information about Si into the standard calling arguments expected by all inference molecules. These calling arguments are similar to those of specifier molecules:

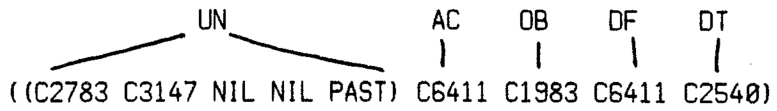
**UN** a list of the following form:

( S TIME TS TF FRAME )

where S is the structure the IPROG is being applied to, TIME, TS and TF are the time aspects of the structure if they could be located (NIL otherwise), and FRAME is "PAST", "PRESENT" or "FUTURE". FRAME is based on TIME, TS and TF, and is determined by a simple time proof procedure which attempts to establish BEFORE relations with #NOW.

**AC OB DF DT** the actual entities in the bond, bound individually as ACtor, OBject, DFrom and DTo. Which, and how many, of these there are of course are specific to the particular conceptual predicate.

To illustrate, the calling arguments thus constructed for the structure underlying "John gave Bill the book" would be:




where C2783 is the structure itself, C3147 is its TIME, C6411 is this person, John, C1983 is the book, and C2540 is the person Bill. The two NIL's indicate that no TS or TF time relations are associated with the structure.

Also, as the IPROG is called, a sub-queue, !INFS, is set to NIL. This sub-queue will be used to collect the set of inferences which arise from IPROG(Pi)'s application to Si.

#### 7.2.1.3 THE STRUCTURE GENERATOR

The inference molecule, IPROG(Pi), generates individual inferences by calling the *structure generator.* When, for instance, an inference molecule has performed tests, and decided that it can generate resultative inference, R, from structure Si, it calls the structure generator with enough information to put R into a new memory structure. This information is listed in section 7.3.2.1.

As we will see, the structure generator can, based on the theoretical type of the inference, decide not to generate the requested inference under certain circumstances. However, when it does generate the inference, it performs the following 5 tasks:

1. it calls lower level bond and superatom creation functions which store the inference in a new memory structure
2. it attaches to this structure the REASONS for making this inference (supplied by the inference molecule) to the newly-created structure
3. it computes the structure's STRENGTH based on the strength factor supplied by the inference molecule and on the STRENGTHs of each structure on the REASONS list
4. it attaches the value of the special atom #NOW to the new structure as its RECENCY. #NOW was set to the time of the system clock when the utterance was received from the conceptual analyzer
5. it records the new structure on a temporary sub-queue, !INFS, which are collecting all inferences made by IPROG(Pi) about Si. This will be appended to the main inference queue after the inference molecule returns control to the basic monitor. The mnemonic denoting the inference's theoretical type is recorded along with the new structure on !INFS.

These five steps occur for each inference IPROG(Pi) requests to be generated.

### 7.2.1.4 EVALUATION AND REORDERING

When IPROG(Pi) has finished, it returns control to the basic monitor, which then retrieves !INFS, and calls the *inference evaluator* to evaluate each new inference on !INFS in turn. The results of this evaluation will be (a) to discover confirmations (points of contact) and merge two memory structures together, (b) to discover contradictions, and (c) to assign each new inference a significance factor which will be used later to reorder the inference queue. We will get to the evaluation process in section 7.5.

After the evaluator has evaluated each member of !INFS, this subqueue has the following format:

```
( ( <theoretical-type-mnemonic> <significance> <memory-structure-pointer> )
  ( <theoretical-type-mnemonic> <significance> <memory-structure-pointer> )
                                        .
                                        .
                                        .
  ( <theoretical-type-mnemonic> <significance> <memory-structure-pointer> )
)
```

That is, it is a list of triples, each triple representing an evaluated inference which has just been made by IPROG(Pi). !INFS is then added to the end of !NEWINFS, which is collecting next-level inferences from all of S1,...,Sn (the current level).

When all of S1,...,Sn at the current level have given rise to the next level of inferences, !NEWINFS is appended to the main queue, !INFERENCES, and reordered on the basis of its STRENGTH and significance factor assigned by the evaluator. Those inferences which lie below a threshhold on this measure are cut off, and placed on a "dead" queue called !CUTOFFINFS. These will not continue in the inference process.

In practice, we want cutoff to occur very seldomly, since the technique for assessing the significance of a given structure is still quite crude, and could erroneously exclude very interesting inferences. Indeed, it is not clear whether there should be *any* cutoff mechanism for a theory of this sort which relies on large quantities of probablistic inferences. Remember, that by modeling the human brain, we are simulating a very sophisticated parallel processor which can perhaps afford to pursue many lines of inference in depth. I will have more to say about this later. In any event, we will not know what is "correct" until the memory becomes much larger. In the current implementation, the inference queue is rarely cut off.

This process of breadth-first inference expansion of the starting queue occurs until (a) no new inferences can be generated from the structures on the most recent level (those on !NEWINFS), or (b) until a pre-determined maximum depth is reached. This depth is currently set to 15, and is rarely achieved in practice because of the relative paucity of inferences currently in the memory. When the process ceases, !!NFERENCES is a list of triples as shown for !INFS above, each triple representing one inference which was generated in response to U. At that point, !!NFERENCES represents the memory's initial inference response to the utterance.

### 7.2.1.5 SUMMARY OF THE BASIC MONITOR

The overall flow of information in the inference monitor is shown in Fig. 7-5. This is not yet the complete picture of the reaction to each utterance.

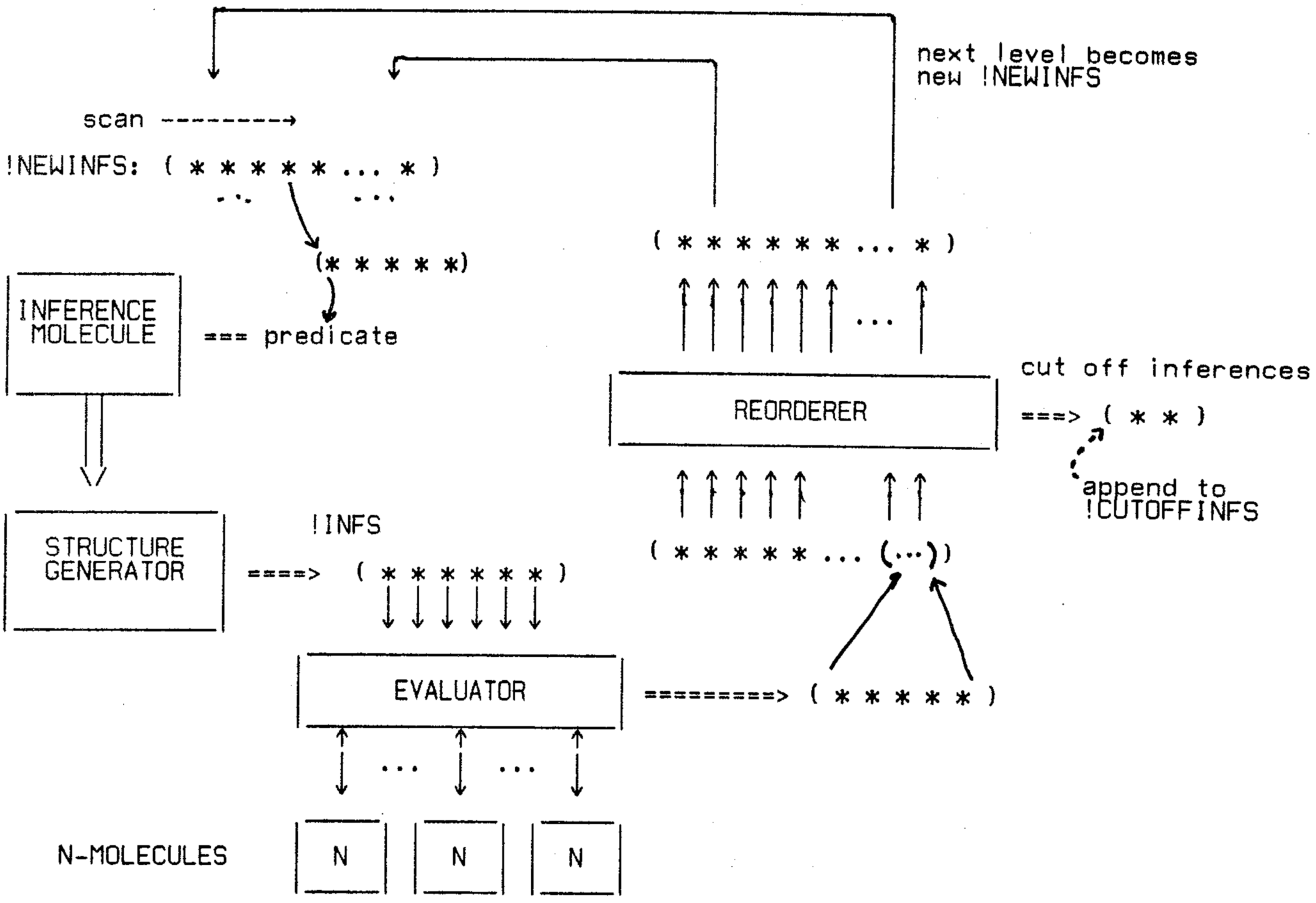



**Figure 7-5. The inference monitor.**

### 7.2.2 THE INFERENCE POSTSCANNER

The basic monitor is the heart of the inference capability, but it is not adequate for certain classes of inference. Recall that there are classes of inference which, in order to function for structure S, require an "after-the-fact" access to *other* classes of inference which have arisen from S. Specifically, two examples of this are *motivational inferences*, which are based upon a knowledge of the resultative inferences from an action, and *knowledge-propagation inferences*, which are based upon a knowledge of the OFFSPRING and REASONS sets of memory structures. In other words, motivational inferences from S cannot arise until *resultative* inferences from S have been generated, and knowledge propagation inferences from S cannot arise until inferences of *all* types have arisen from S. Other classes of conceptual inference will probably emerge which will require similar after-the-fact information.

To accomodate such classes of inference, there is a special process, POSTSCAN, which rescans !INFERENCES after the basic monitor has ceased. Currently, POSTSCAN searches for structures of the following three varieties on the inference queue:

1. Action structures
2. WANT structures
3. MLOC #LTM structures

As each structure on !INFERENCES which satisfies one of these three patterns is detected, the POSTSCANer invokes the appropriate process: the motivational inference generator for the first two cases, and the knowledge propagation inference generator for the third case.

These processes will return a list of inferences, which are collected on !NEWINFS as POSTSCAN scans !INFERENCES. After all the postscan inferences have been collected, each in turn is evaluated by the evaluator, and a theoretical (type/significance/structure) triple is assembled for each. The resulting list is appended to !INFERENCES, which represents at that point the results of attempts to generate inferences of all theoretical types from the starting subpropositions extracted from the input utterance.

### 7.2.3 RELAXING THE INFERENCE NETWORK

But even after this postscan process, the first-pass through the inference monitor is still not complete, because there is still a potential for generating more inferences from this first pass inference queue. To understand why, observe that the monitor is an inherently sequential modeling of what I abstractly envision to be a parallel, breadth-first expansion of inferences. The problem is this: as the monitor generates inferences from structure Si on the inference queue, some inference, X, from Si may be *almost* applicable, except for one important missing fact, F. At that point, X can therefore not be generated. But suppose *F arises later down the queue*, as an inference from some other structure, Si+j. If the monitor were to stop after the POSTSCAN process, X might still not exist, even though there would *then* be sufficient information to generate it. Since X may itself be an important inference, and might lead to other important inferences, it should not be missed because of "bad timing". This undesirable situation is illustrated in Fig. 7-6.

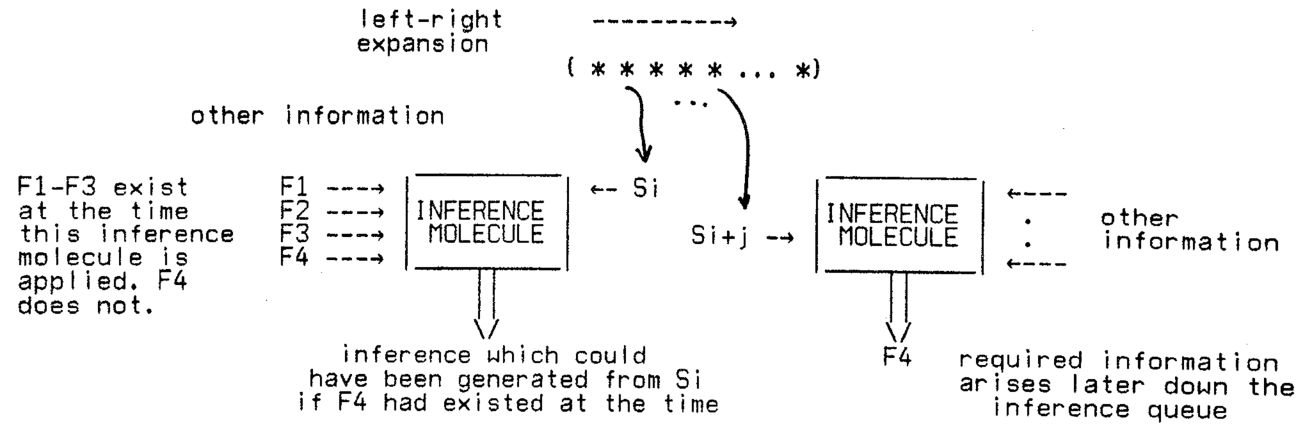



**Figure 7-6. "Bad timing."**

In practice, such bad timing is quite common, because of the way in which two of the most critical classes of conceptual inferences arise: motivational and knowledge propagation inferences can be generated only after the first crop of inferences has arisen through the basic monitor. If things were to stop there, significant inference potentials could go unrealized, because many inferences rely heavily upon what actors *want* and *know* at any particular time. This observation is borne out empirically in the current implementation.

To prevent this order-sensitive characteristic of the inference mechanism from causing important inferences to be missed,

> **Each structure on !INFERENCES is reexamined, re-applying inference molecules to each in an attempt to generate inferences which were missed on the first round because of order dependencies.**

Any new inferences which arise from this scan are evaluated and appended to the main queue, then subjected to the POSTSCANner just as first-round inferences. This rescan/re-postscan sequence is repeated until no new inferences arise. At that point, what I have called the "first-pass" through the inference monitor has ended.

This rescanning mechanism is a form of relaxation processing on what is in reality a very large, parallel inference network. But the ability to rescan comes not without a price: the monitor and inference molecules must be smart enough not to duplicate work which was done on previous passes. We will see in section 7.3.4 how this problem has been solved.

### 7.2.4 INFERENCE-REFERENCE-INFERENCE INTERACTION

At that point, !INFERENCES is a list of all first-pass inferences generated in response to the utterance. Of course, this list is not of much significance in itself.

> **The real effect lies in the existence of all the new structures which have been created, and in the structure merges and contradictions which are discovered during this process by the evaluator.**

Also, this list will serve as the beginning inference queue for subsequent *inference-reference* passes. I will outline here the general form of the interaction between the reference and inference mechanisms, and the reasons for this interaction.

The basic observation is this: some very interesting inferences may not be generated on the first inference pass because of incomplete features of entities in the structures on the inference queue. For instance, if some "John" could not be unambiguously identified by the reference mechanism *before* the first pass of inferencing began, the entity which represents this unidentified person will be a temporary token which in general will have nowhere near the

richness of features of any particular "real" token of a person, John. Because of this temporary token's lame-duck occurrence set, there is a good chance that many interesting inferences from structures which involve it cannot be generated on this first pass.

But fortunately there is another side to this coin: *the process of inferencing can contribute features to this temporary entity.* The crucial point is that these new features might be able to identify it as, say, John Smith, the carpenter, if only the referencer had another attempt to identify it, using some of the newly-inferred inferences from the first inference pass.

To account for these phenomena, there is a higher form of inference-reference relaxation processing in the memory. After the first pass of inferencing, in case there are some pending unidentified references either in this utterance, or from previous utterances -- this condition is signaled by a non-null !REFDECISION or !REFNOTFOUND list -- *the referencer is reentered.* The hope is that new features of unidentified entities have been produced as a byproduct of the inference process, and, by using these new features, the referencer can select one reference candidate over the rest. If this can in fact be accomplished, the information-rich occurrence set of the identified entity (say John *Smith*) will become available.

But then, because of all the newly-accessible features, new inferences may be possible by rescanning the existing inference queue for new inferences *which weren't previously possible.* If new ones can be made, the relaxation processing described in the previous section is performed, and then still another round of reference-inference interaction is performed. This is depicted in Fig. 7-7.

By this interaction, the inferencer helps the referencer, which in turn helps the inferencer, and so on. Whereas the relaxation processing described in the previous section was necessitated by practical issues of implementation,

> **this form of relaxation processing realizes an important theoretical interaction between the memory processes of reference and inference.**

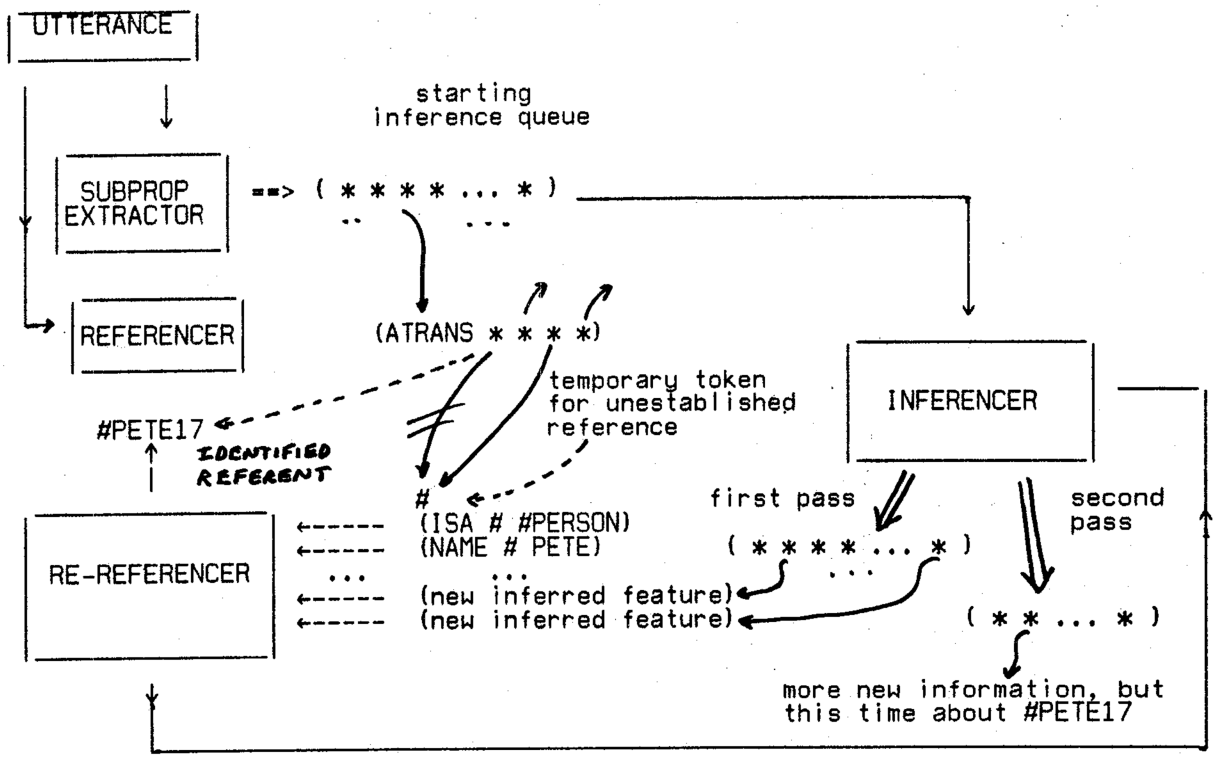



**Figure 7-7. Multiple reference-inference interaction passes.**

## 7.3 INFERENCE MOLECULES

**How are inferences organized? What does an individual inference look like in the memory?**

Any procedure-based system capable of inference must ultimately be no more than sequences of tests on the features of the information from which an inference is desired, interspersed with inferences to be made when tests succeed. In the kind of memory I am proposing, there will generally be *many* applicable inferences from each unit of information. The goal is *not* to choose among them, but to *make them all*, and see what happens. Thus, the effect I want to simulate is one in which many associatively-triggered inferences are simultaneously applied to relevant memory structures as those structures arise in the memory (either from the

outside world, or from other inferences). Each conceptual inference is imagined to be an active process which is constantly on the lookout for its applicability, and which spontaneously contributes its inference when appropriate.

### 7.3.1 EXTERNAL ORGANIZATION

What kind of inference organization can achieve this goal? Recall that the input which the memory receives has already undergone a significant amount of processing by the conceptual analyzer from its "raw" sentential form. As we have seen, the intent of this processing is to reduce each utterance in context to its underlying conceptual meaning. From the standpoint of inference organization, this means that a tremendous quantity of fairly sophisticated pattern matching has already been performed. To reemphasize how significant this can be, consider the three sentences

Mary gave John a beating.
Mary gave John a pencil.
Mary gave John some responsibility.

All three have very similar *surface* forms, but all have radically different *underlying meanings*. Since it is the role of the conceptual analyzer to capture in a conceptual graph the most likely underlying thought of the *language form*, recognizing the many variations of language forms which communicate the same thought is *not* a concern of the conceptual memory.

Because of this, it is possible to organize inferences about "real" GIVE actions (the transfer of an object's possession) under the conceptual predicate ATRANS, without having to know or care about the actual sentence *form* which communicated the thought or, for that matter, even the *language* in which the utterance was spoken.

**Conceptual inferences can get directly to their business of dealing with how the *meaning* of each utterance interacts with other knowledge, without having to cope with all the additional variety of language form.**

This leads to a very natural and simple organization of inferences in the memory: inferences are organized by conceptual predicates. By this I mean that every inference which could ever be applicable to any memory structure which stores information involving predicate P should be

associated with the predicate P in the memory. Obviously, this is only a very general organization, and there will typically be an extremely large number of inferences grouped by this organization under any one conceptual predicate. I have called the large cluster of inferences associated with each predicate an *inference molecule.*

Logically, we can view an inference molecule as a very large, "sloppy" discrimination net which can yield multiple responses. Each response is a conceptual inference of a certain theoretical type, and the molecule will in general yield responses of many types at once

Physically, an inference molecule is an executable LISP PROGram which contains all inference potentials for some conceptual predicate in the system. That is, if inferences are desired from some structure in memory which is an ATRANS action, the inference molecule which is associated with ATRANS -- and only this one -- can generate them. Furthermore, essentially no pattern matching is performed to *locate* the relevant inference molecule, since, to generate inferences from memory structure S which involves conceptual predicate P, the inference monitor simply retrieves the P inference molecule and applies it to S.

### 7.3.2 INTERNAL STRUCTURE

I want to shy away from sophisticated or prematurely elegant inference structures until our comprehension of the complete picture of interesting tasks for such a memory as this has has time to mature. The internal architecture I am about to describe

(a) is unclever as data structures go
(b) is about as straightforward as possible
(c) does not make very efficient use of storage or time.

But it has made experimenting with the memory quite simple, pleasurable -- and possible! I view the next major step in the memory's development as being to clean up the internal structure of inference molecules.

An inference molecule is not a totally random piece of program, however. Fig. 7-8 shows the general form of all inference molecules. Each conceptual inference rule within a molecule is called an *inference atom.* Inference atoms can be totally independent of each other, or can share

common tests heirarchically, being structured more like a large decision tree than like a bundle of independent test packets.

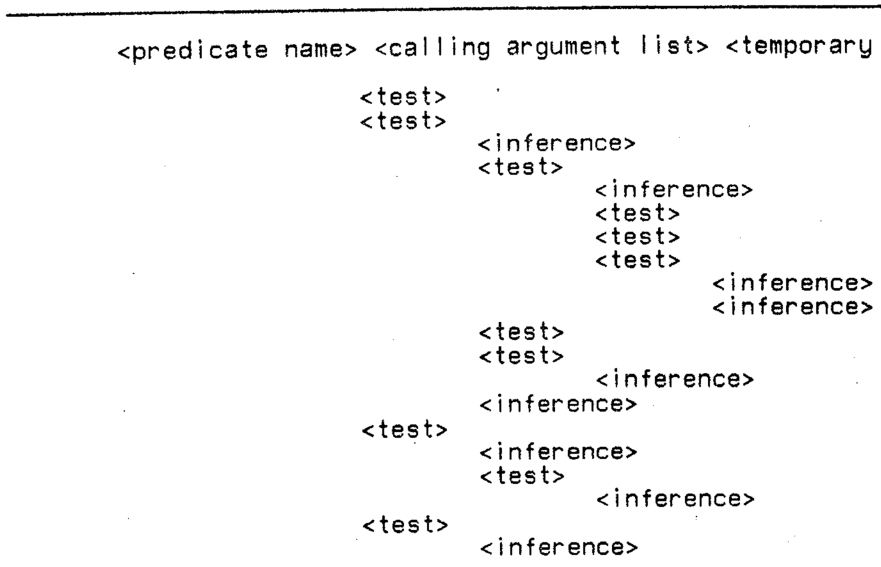



**Figure 7-8. The logical internal organization of a typical inference molecule.**

7.3.2.1 INFERENCE ATOMS

Each inference atom (<inference> in Fig. 7-8) has the internal structure shown in Fig. 7-9. This structure consists of the following 8 parts:

1. the *theoretical type* of the inference about to be generated
2. the *reference name* of the inference atom
3. a list which is to become the *bond* of the main structure which is about to be created as the inference
4. a *default significance factor* in case the evaluator cannot assess the new inference's significance
5. a *reason list.* This will become the new structure's REASONS, and is each inference atom's way of making explicit what other information in the memory has been used to generate the inference.
6. a *propagation strength factor.* This is the strength factor with which the new inference is to be generated. When multiplied by the strengths of each member of the REASONS list, the product will become the STRENGTH of the new structure.
7. lists which represent *modifying structures* for the main inferred structure. These will become the main structure's starting occurrence set, and are most frequently time modifications.

8. propagation strength factors and reasons for each modifying structure

```
<theoretical type> <reference name>

        <main structure>
        <propagation strength> <reason list>

        <modifying structure>
        <propagation strength> <reason list>

                ...

        <modifying structure>
        <propagation strength> <reason list>
```

**Figure 7-9. The structure of an inference atom.**

Fig. 7-10 shows an actual inference molecule which will be described shortly.

### 7.3.2.2 COMMON LOW-LEVEL PATTERN MATCHING TESTS

Inferences in an inference molecule can arbitrarily sensitive to features of the structure it is testing, and to other contextual information, since the basic structure of the molecule is test-branch-infer, where the *test* phase may access arbitrary functions, and arbitrary features of entities in the structure from which the inference atom is attempting to generate inferences. Although the number of tests has continually been on the increase as the memory has developed, there are a few recurring tests which I can list here to give a feeling for the types of questions inference atoms ask to determine their relevance to the structure under inferencing. Some of the most common are:

**(find x)** locates all structures with bond X, returning as its value the list of structures. If X contains one occurrence of "underbar", then FIND interprets X as a simple template, and returns a list Y1,...,Yn of all structures which could be substituted for the underbar. In this case, the list of structures where Y1,...,Yn were found are recorded on a special global list !GLOBALFIND. Find will not accept modifications of X.

EXAMPLES:

(FIND (ATRANS #JOHN2 C1876 #JOHN2 #MARY1)) returns a list of all ATRANS structures of this form in the memory.

(FIND (LOC #JOHN1 _)) returns a list of all of John's locations (independent of time).

Also, !GLOBALFIND stores the list of the memory structures where these LOC information units were stored. Thus, a typical response for this query might be
(#AILAB #SANFRANCISCO #FRANCE)

with !GLOBALFIND set to
(C2371 C9762 C1103)

if these three structures stored the information (LOC #JOHN X) for these three X's.

**(findl x)** same as FIND, but will return only the first item found

**(findunit x)** X has the form (<main structure><modifier>...<modifier>). A list of all memory structures which have the form of the main structure, and which further satisfy the modifiers is returned. As with FIND, the main structure may have an underbar in it. Each modifier has one underbar denoting the main structure, and the main structure may be either a simple entity, or a bond. If it is a simple entity, a NIL appears and the modifiers are assumed to be its defining features. Any concepts or tokens within the main structure or modifiers may itself be described by a template suitable for use by FINDUNIT. This allows arbitrary nesting of features.

EXAMPLES:

(FINDUNIT ((ATRANS #JOHN2 C2315 #JOHN2 #MARY1)(LOC _ #AILAB)(TIME _ T1))) returns a list of ATRANS structures of this form which occurred at the AI Lab at time T1.

(FINDUNIT (NIL (ISA _ #PERSON)(NAME _ #JOHN)(POSS _ (NIL (ISA _ #CAR)(COLOR _ #RED))))) returns a list of all people named John who own a red car.

(FINDUNIT ((LOC #JOHN _)(TIME _ (NIL (BEFORE T1 _)(BEFORE _ T2))))) returns a list of all places John was located during the period T1-T2.

**(eq x y)** this is the LISP test for equality of pointers.

**(whatisit x)** returns entity X's immediate ISA class. For concepts and tokens, this involves a simple (FIND1 (ISA X _)). However, since (by convention) an information bearing structure will not be explicitly classified by an ISA relation, WHATISIT must examine its predicate to determine what the class of the structure is. The possible ISA classes of information bearing structures are: #ACTION, #STATE, #STATECHANGE, #CAUSAL, #TIMEREL.

**(hasprp x y)** searches for feature Y of X, or for feature Y of any ISA superset concept of X. Thus, if feature Y cannot be directly located for X, HASPRP locates X's superset via a (FIND1 (ISA X _)), then attempts to find Y for this superset concept. This is continued until Y is found, or until the ISA superset sequence has been exhausted.

EXAMPLE:
(HASPRP #JOHN (PART _ #HEAD))

**(isastar X)** returns X's ISA set sequence in increasing generality.

EXAMPLE:
(ISASTAR C1135) would return (#HAMMER #TOOL #PHYSOBJ) if C1135 were some token of a hammer.

**(event x)** tests X's time aspects to determine whether X is a real event, namely that it has actually occurred in the past, or is presently occurring. This distinguishes it from other structures which have been stated or predicted to occur in the future, or which are timeless statements of fact about the world.

EXAMPLE:
(EVENT C2734) is true is C2734 is some information bearing structure representing something which has actually occurred in the world.

**(causer x)** X is assumed to be some information bearing structure. CAUSER traces back X's CAUSE relations until some action structure is found. The actor is then extracted from this action, and returned as the CAUSEr of X. That is, CAUSER traces down the actor most immediately responsible for the existence of some action or state structure. The actor and the structure representing his action are returned as a LISP dotted pair.

EXAMPLE:
(CAUSER C8764) would return (#JOHN3 . C6513) if C8746 were the structure (NEGCHANGE #BILL #PSTATE), and if this NEGCHANGE had been caused by John's action.

### 7.3.3 AN INFERENCE MOLECULE EXAMPLE

Fig. 7-10 shows an actual inference molecule used by the program. There are currently only about 25 inference molecules, and a typical molecule contains just 3 or 4 inference atoms. This is little more than a token beginning, since I envision future inference molecules as containing thousands of atoms of about the same complexity as those shown. Undoubtedly, many new issues of effective organization which I have not yet addressed will arise.

Let's now take a look at the NEGCHANGE inference molecule shown in Fig. 7-10.

```
(IPROG NEGCHANGE (UN PE SC) (X1 X2) (
(COND ( (EVENT UN)
        (COND ( (F1 (@ISA PE @#PERSON))
                (IR @NEGCHANGE1
                    (@WANT PE (GU (@POSCHANGE PE SC)))   ~PEOPLE OFTEN WANT TO BETTER
                    (0.95 1.0 (CAR UN))                  ~THEMSELVES AFTER SOME NEGCHANGE
                    (@TS @* (TI UN))
                    (1.0 (CAR UN)))
                (COND ( (AND (SETQ X1 (F1 (@*MFEEL* @_ @#NEGEMOTION PE)))
                             (SETQ X2 !GLOBALFIND))
                        (IR @NEGCHANGE2
                            (@POSCHANGE X1 @#JOY)
                            (0.9 1.0 (CAR UN) X2)        ~PERSON GETS HAPPY WHEN ENEMY
                            (@TIME @* (TI UN))           ~SUFFERS NEGCHANGE
                            (1.0 (CAR UN)))
                      )
                )
                (COND ( (AND (SETQ X1 (CAUSER (CAR UN)))
                             (NOT (EQ (CAR X1) (α2 (CDR X1)))))
                        (IR @NEGCHANGE3
                            (@*MFEEL* PE @#NEGEMOTION (CAR X1))  ~PEOPLE DON'T LIKE
                            (0.95 1.0 (CAR UN) (CDR X1))         ~OTHERS WHO HURT THEM
                            (@TS @* (TI UN))
                            (1.0 (CAR UN)))
                      )
                )
              )
              ( (HASPRP PE (@ISA PE @#PHYSOBJ))
                (COND ( (AND (SETQ X1 (F1 (@*OWN* PE _)))
                             (SETQ X2 (CAUSER (CAR UN)))
                             (NOT (EQ X1 (CAR C2)))
                        )
                      (IR @NEGCHANGE4
                          (@*MFEEL* X1 @#NEGEMOTION (CAR X2))  ~IF X DAMAGES Y'S PROPERTY
                          (0.85 1.0 (CAR UN) X1 (CDR X2))      ~THEN X MIGHT FEEL ANGER
                          (TS @* (TI UN))                      ~TOWARD Y
                          (1.0 (CAR UN)))
                      )
                )
              )
      )
    )
)
))
```

**Figure 7-10. An inference molecule used by the current program.**

Fig. 7-10 shows the form in which the molecule appears in the inference data file. The IPROG tells the initialization function that what follows is the inference molecule for the

conceptual predicate NEGCHANGE. UN, PE and SC are the three calling arguments for the molecule which are extracted from the structure under inferencing: UN is the list whose first element is the structure itself, and whose remaining elements signal the time aspects of the structure as described above. PE is the entity which underwent the NEGCHANGE, and SC is the scale on which it occurred. X1 and X2 indicate to LISP that the molecule will be using these two temporary variables during the testing it will perform.

The first tests determine whether the NEGCHANGE actually occurred, and whether the entity which underwent the NEGCHANGE is a person. The first three inferences in this simple molecule are designed for actual NEGCHANGE events which occur to *people*; the fourth concerns NEGCHANGES to *objects*. If other inferences dealing with *future* or *timeless* NEGCHANGES existed, they would follow at the end of the molecule (the false branch of the EVENT test).

If the EVENT test is satisfied, one inference is immediately requested: that the person who underwent the NEGCHANGE may desire to undergo a POSCHANGE on the same scale. The component

```
(IR ⊗NEGCHANGE1
    (@WANT PE (GU (@POSCHANGE PE SC)))
    (0.95 1.0 (CAR UN))
    (@TS @* (TI UN))
    (1.0 (CAR UN)))
```

is an inference atom. The "IR" calls the structure generator and signals that the inference is of type RESULTATIVE. ⊗NEGCHANGE1 is this inference atom's reference name, and will be recorded under the property list of the structure from which this inference is being generated.

The next line is the bond which represents the inference:

```
(WANT PE (GU (@POSCHANGE PE SC)))
```

namely, that the person might want to undergo a positive change to compensate for his negative change. GU is a call on function GETUNIT, which creates the substructure (POSCHANGE PE SC) for reference by this inference.

The next line

```
(0.95 1.0 (CAR UN))
```

gives the strength propagation factor for this inference (0.95), a default significance measure (1.0) to be used in case the inference evaluator cannot assess this inference's significance, and the remainder of this line enumerates the REASONS to be attached to the new inference, and whose STRENGTHs will be used to compute the STRENGTH of the new inference. In this first inference atom, the only REASON supplied is the NEGCHANGE structure itself, (CAR UN).

The next lines,

```
(@TS @* (TI UN))
(1.0 (CAR UN))
```

specify that this new inference structure is to be modified by a time relation: that the person *begins* his wanting at whatever the time of the NEGCHANGE was. The time of the NEGCHANGE is retrieved from the time vector set up by the inference monitor by a simple function, TI. The asterisk refers to the *main* structure, (WANT ...). The 1.0 is the strength factor for the modifying time structure, and the (CAR UN) is the reason supplied for the modifying structure's existence. In general, time modifiers are given the same STRENGTH and REASONS as the main structure.

The @NEGCHANGE2 and @NEGCHANGE3 inference atoms are similar. Notice in @NEGCHANGE2 however that *two* reasons are supplied:

```
(0.9 1.0 (CAR UN) X2)
```

Here, X2 will be pointing to the structure which stores the information that some other person MFEELs a negative emotion toward the person who underwent the NEGCHANGE. The inference that this person might become happy because of the other person's NEGCHANGE is thus based on both the NEGCHANGE structure, and on this MFEEL information. Although the particular substance of this inference -- as are most of the inferences the memory currently makes -- is more appropo of a soap opera, it illustrates the desired underlying mechanism.

The inference atom, @NEGCHANGE4, implements the inference that if

1. the entity undergoing a NEGCHANGE is a #PHYSOBJ
2. the scale is #PSTATE
3. the owner of the object knows that the NEGCHANGE occurred
4. some other person was the CAUSER of the NEGCHANGE

then it is possible to infer that the object's owner might feel #ANGER toward the CAUSER. This is of course also crude in substance, but as many other discriminating tests as necessary could quite easily be inserted.

### 7.3.4 MULTIPLE INFERENCE PASSES: SMART INFERENCE ATOMS

The inference monitor is in reality simulating a large parallel inference network via breadth-first expansion of inferences. Because this is a serial simulation of a parallel process, there are several undesirable characteristics which must be overcome by a iterative relaxation technique which involves a rescanning of the inference queue, retesting for newly-applicable inferences from each structure on the queue.

The following four points summarize why this relaxation processing is essential:

1. some information which is vital to one inference may not turn up until later in the expansion (perhaps along another line of inference). This is undesirable, because it is purely an artifact of the sequential simulation of a parallel inference network.
2. the inferences contributed to the inference queue by the postscan process are available only after the first inference pass finishes. These can lead to more interesting inferences, especially in combinations with some of the inferences generated on the first monitor pass. Without subsequent passes, the inferences contributed by the postscanner would never be considered again.
3. there are interactions concerning the establishment of references which cannot be solved by a simple one-pass breadth-first inference mechanism.
4. there can be cycles in the inference network

This ability to rescan the inference queue incurs two new problems which require solution. In particular, the rescanning process should be able to function

1. without duplicating much computation
2. without re-generating any inferences it made on previous passes

This multiple pass capability is achieved at the inference atom/structure-generator interface: the memory has "smart" inference molecules.

Fig. 7-11 shows how each inference atom is made smart enough so that its inference made on a previous pass will not be duplicated on subsequent passes. Associated with each inference

atom, A, is a unique identifier, I(A), which serves to "name" the atom within its inference molecule. Whenever execution passes to A and A calls the structure generator, I(A)'s existence on the property list of the superatom of the structure, S, under inference, is tested. If I(A) exists on S's property list under property ISEEN, this means that inference atom A has already generated its inference from S, and that the structure generator should not re-honor its request. Notice that it is essential that a record be kept that the inference atom has already operated on *this particular S*, and not just that it has operated on *some* S on a previous inference pass: the same inference atom might be applicable to numerous distinct structures during inferencing. For example, "John hit Mary" might lead to "John is mad at Mary", and to "Mary is (now) mad at John", which are both treated by the same inference atom.

If I(A) is found not to exist under property ISEEN on S's property list, this means that inference atom A has not previously successfully generated an inference from S. If A's applicability tests are still not successful, nothing else happens: no inference is generated and I(A) is not placed on S's ISEEN list. However, if the tests are successful this time, A generates its inference and I(A) is placed on S's ISEEN list.

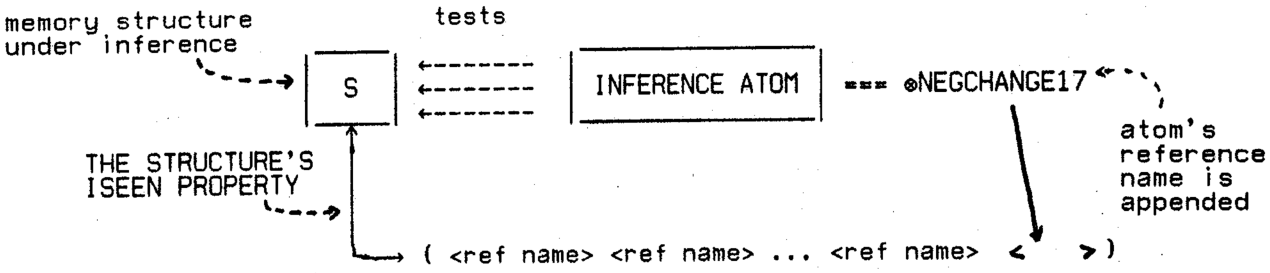



**Figure 7-11. Recording the successful application of an inference atom to a memory structure.**

This ability to rescan the inference queue has an intuitive psychological analog which merits brief mention. The notion of passing information through the inference network more than once roughly corresponds to "rehashing the problem in the light of new evidence." As the rehash proceeds, the human language user remembers each avenue of inferencing he pursued from some piece of information, saying for each either "Now, I have already examined the implications of this, and don't believe any more will come of it" or "this didn't mean much a minute ago, but in

light of this new information, it might mean something now." This is admittedly a rough analogy, but aside from the four practical considerations described at the beginning of this section, the concept of a "smart" inference atom has this intuitive psychological appeal.

#### 7.3.4.1 CONCERNING THE REFERENCEABILITY OF INFERENCE ATOMS

By choosing program-based pattern matching over data-based matching for inference molecules, I have in effect made part of the memory's world knowledge inaccessible to "introspection" and reference. That is, because inference molecules are programs rather than data, they cannot be "discussed" or expressed outside of the system in the same way passive data structures can. Although the memory can *apply* the rule contained in a conceptual inference atom to generate new information, and supply REASONS for having generated the new information, without a label for each inference atom no relation between the new information and the actual rule of conceptual inference which generated it would be possible. By placing each inference atom's identifier on the inferred structure's REASONS list, this relationship between every structure in the memory and the conceptual inference which caused it to be generated could be preserved.

Also, using this labeling scheme, inference rules could be made accessible in data form: each inference atom identifier is a unique LISP atom such as ⊗NEGCHANGE3. On this atom's property list, a "passive" *data* representation of the inference rule which the inference atom realizes could be stored in the memory data structures described in chapter 3. Although this would represent duplicated information (every conceptual inference would be encoded in both an easy-to-execute and an easy-to-inspect and reference form), it would afford the best of two worlds: fast program-based matching, yet access at the data level to the conceptual content of rules of inference. This idea is illustrated in Fig. 7-12 as it might apply to the ⊗NEGCHANGE3 inference atom of Fig. 7-10.

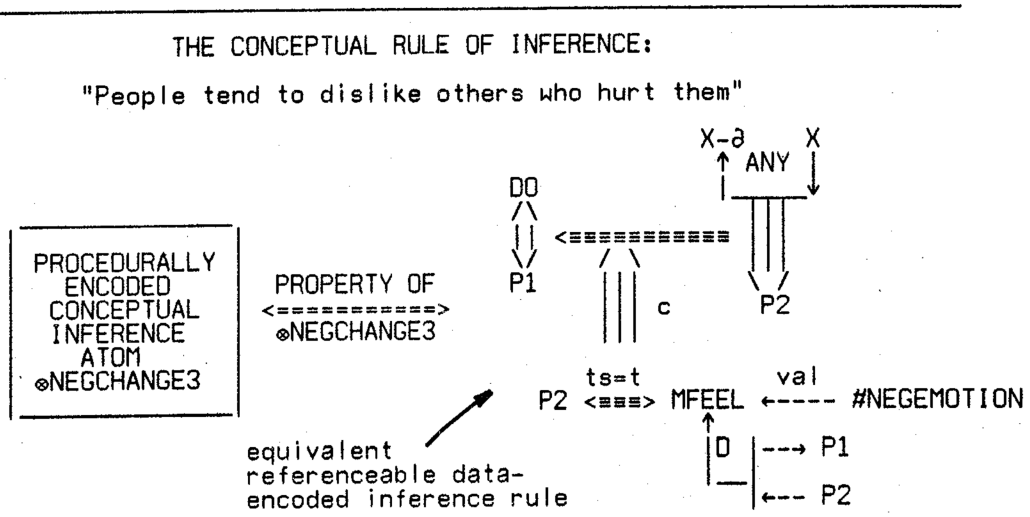



**Figure 7-12. Making program-based conceptual rules of inference referenceable as data structures.**

## 7.4 MULTIPLEXING INFERENCES BY THEORETICAL TYPES

The expansion of an input into many probablistic inferences represents the memory's reflex attempt to relate every language stimulus to the context in which it occurs, and to world knowledge in general. For this purpose, *all* types of inference discussed in chapters 5 and 6 are of extreme potential interest. Every aspect of the input potentially merits examination and elaboration via conceptual inference. Without this, significant relations may remain undiscovered.

However, there are times at which it would be useful to restrict the generation of inferences to those *of certain theoretical types only*. These are times when some specific task or process requires a much narrower analysis of a situation for its goal-specific needs. This "narrowing of analysis" corresponds with the ability of the inference monitor to allow the application of only certain types of inference to some proposition. Hence, the term "multiplexing", or the selecting-out of one or several "signals" from a larger group of potential ones.

### 7.4.1 WHY MULTIPLEXING IS NECESSARY

In the memory, there are several such goal-specific processes which require of the inference monitor the ability to "multiplex" or filter out all but certain kinds of inference. A motivational inference, for example, is triggered by the pattern of a person wanting an action to occur in the world. The goal of this type of inference is to determine why the person might possibly want the action to occur. As we have seen, a good assumption is that people want actions to occur because of the states those actions could produce in a particular environment. The knowledge of what changes to the world an action might be capable of effecting is contained in inference molecules, which, when executed on some proposition in some environment, can yield a set of resultative inferences in addition to many other inferences of other types. Therefore, to carry forth a motivational inference from an action and the possible states that action could cause, there must be some way of generating *only resultative inferences* from a WANT-ACTION pattern. This ability to multiplex inferences by type enables processes such as this to contribute to the larger, unmultiplexed, expansion of a structure in inference space.

Another example concerns the task of making action prediction inferences. Inferences in this class start from a person's internal WANT states and predict what actions in the world he might reasonably be expected to carry out as a result of these states. Thus, from a WANT state of a person, several WANT-ACTION patterns involving the person may arise. However, rather than stop at that point, the predictive inference generator seeks out the extrinsic event-enabling preconditions for each of these predicted actions. For those preconditions which are not already known to be satisfied, other WANT-STATE patterns arise, and these in turn can lead to more action predictions. Hence, in the process of generating action prediction inferences, points exist which require that only extrinsic event-enabling inferences for certain actions in certain environments be generated by (allowed to pass through) the inference monitor.

Still another process which relies upon this multiplexability is that of enablement prediction. This process implements the notion that particular states of the world are often desired because of the actions they enable. That is, it is often possible to work forward from a WANT-STATE pattern to WANT-ACTION patterns, where the state which is WANTed is a common extrinsic enabling state for the actions. Clearly, in order to accomplish this, a point comes where only enablement prediction inferences are desired from some state.

There are many other potential uses of multiplexing, such as conducting very narrow searches backward from an event to determine only its original causes, or performing an extensive but narrow resultative-inference analysis of some situation, and so on.

### 7.4.2 THE MULTIPLEXOR

The method by which multiplexing is achieved is straightforward. Every recursive entrance into the inference monitor has associated with it an *inference filtering* (IF) vector. This vector specifies, by mnemonics standing for the various inference types, *which* inference types are to be passed by the structure generator. For example, the IF vector which allows only action prediction, extrinsic enabling, and result inferences through would look like

( A EE R )

There is a pre-defined vector which is simply a list of *all* type mnemonics, and this is the default filter vector: when the filter vector consists of all types, this is the "global" mode of operation, where the expanding sphere of inferences about points in inference space "to see what might be seen" is the only goal.

The IF vector is transparent to inference molecules. This means that an inference molecule can execute as though it were generating inferences of all types. Undesired inferences are intercepted and suppressed by the structure generator which is called by all inference molecules to generate their various inferences. The postscanner is also sensitive to this vector.

Fig. 7-13 illustrates the multiplexing process.

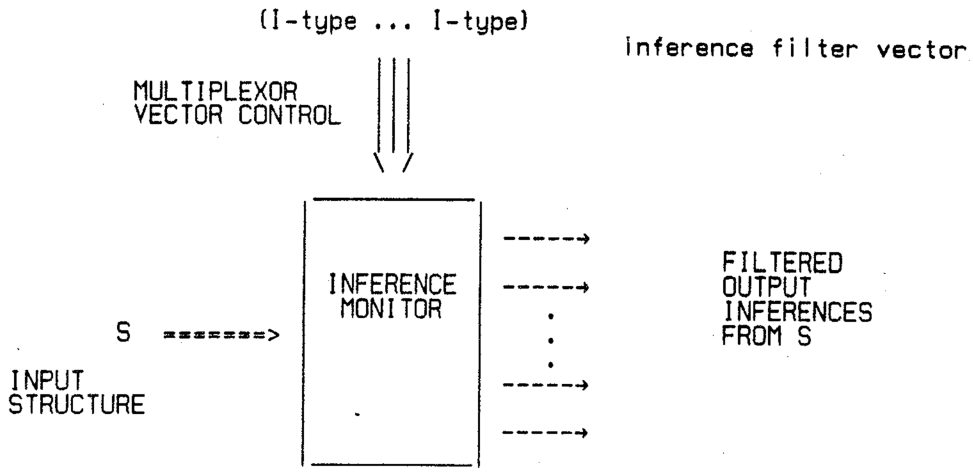



**Figure 7-13. Multiplexing inferences by theoretical type.**

### 7.4.3 AVOIDING INFERENCE BACKFLOW IN THE NETWORK

There are several pairs of conceptual inference classes which perform functions that are inverses of each other. That is, where one class may look "forward", from some point in inference space, another companion class may simultaneously be looking "backward" from the same point. Examples of this are the resultative/causative pair, the enabling/enablement-prediction pair, the missing enablement/intervention pair, and the motivational/action-prediction pair. In each of these pairs, inferences in one class have the potential for inferring, say, structure S1 from S, while inferences in the other class of the pair have the potential for inferring S2 from S. The problem is that the same kind of inference which is capable of carrying S to S2 is precisely the kind which could carry S1 back to S. Similarly, the kind of inference which carried S to S1 is precisely the kind which could carry S2 back to S. This undesirable situation, which I will call *inference backflow* is illustrated in Fig. 7-14.

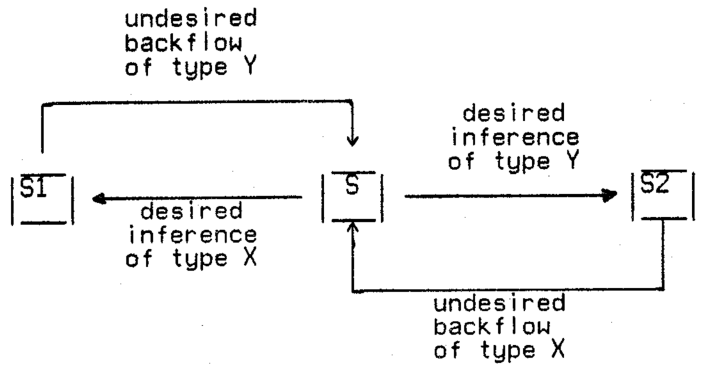



**Figure 7-14. Inference backflow.**

In words, the desire to avoid backflow means, for instance, that if we have just generated a *resultative* inference, S1, from S, we do not then want to worry about generating any *causative inferences* from S1: we *know* what the cause was; it was S, because S1 just arose from S by a resultative inference! This is admittedly rather a mundane issue, but without a stop to prevent this backflow, many lines of inference would either be duplicating effort or would lead to recursive cycles in the expansion in inference space.

The solution to the backflow problem relates to the multiplexor. As described, the inference monitor has the ability to filter out all inferences but those types which have been requested by some subprocess in the memory. Recall also that, as each inference of any is generated, a mnemonic representing its theoretical type is also indicated by the inference molecule which requests that it be generated. In this way, the monitor can filter it out if it is not of a desired type, simply by checking whether this mnemonic is a member of the filter vector.

We can make use of this filtering to avoid backflow. As each inference is generated and added to the ever-growing inference queue to be expanded later itself, its type mnemonic is recorded on the queue along with it. For instance, if S1 is a resultative inference from S, not just S1 is placed on the queue, but rather ("R" S1) to indicate that S1 arose via a resultative inference. When S1 subsequently comes up for inference, the monitor will examine S1's associated mnemonic. If the mnemonic is one of these pairs of inverses, the mnemonic of its

inverse is *removed* from the filter vector if it is currently on it, then inferences from S1 are generated by applying the appropriate inference molecule. Thus, if S1's type is "R" (resultative), then "C" (causative) inferences will be disabled by the filter vector. When the inference molecule returns control to the monitor, the original filter vector is restored, and the monitor proceeds.

## 7.5 RECOGNIZING POINTS OF CONTACT: THE INFERENCE EVALUATOR

**What happens to an inference after it is generated? How is it related to existing knowledge? What happens when it confirms or contradicts some other information in the memory?**

The process of *inference evaluation* represents the fruit of all the memory's labors, since it is the means by which new points of contact are recognized in inference space.

The problem is this: a new inference is generated, and something must be done to relate it to other structures. On obvious thing is to integrate it into the memory, but this is automatically done by the structure generator as part of the process of inference. We are more concerned with how this new piece of information relates to *other knowledge* in the memory at that point. Were the memory not able to do this, it would never connect lines of inferencing, and hence would never really "understand" the connection between the information in one line of a story or dialog and others in that story or dialog. Equally catastrophic, the inference monitor could blithely duplicate the same information over and over, not realizing that all related to the same situation in potentially interesting ways.

### 7.5.1 POSSIBLE INTERACTIONS: CONFIRMATION, CONTRADICTION, AUGMENTATION

What, specifically, does it mean to relate a new piece of information, S, to existing world knowledge? That is, what are the possible interactions of S with other information structures in a conceptual memory? There seem to be five very general ways the new structure S can relate to world knowledge:

1. S *matches* some existing memory structure. That is, the new structure references the same action or state as some other existing memory structure.
2. S *contradicts* some existing structure
3. S conforms to the memory's knowledge of what is normal in the world

4. S deviates from the memory's knowledge of what is normal in the world

5. S is "neutral" (none of 1-4 applies)

As we will see, each of these conditions has a different effect on the inference mechanism, and all are usually "fuzzy" events. For the purposes of classification, I will refer to case (1) as *confirmation*, to case (2) as *contradiction*, and to cases (3), (4) and (5) as forms of *augmentation*.

### 7.5.2 REACTIONS TO COMFIRMATION, CONTRADICTION AND AUGMENTATION: INTUITIVELY

Briefly, confirmations indicate that some point of contact has been established between two different lines of inferencing. Contradictions indicate that something peculiar has been discovered, or that a prediction has turned out to be wrong, or that some incorrect reference decision has occurred in the memory. Unlike formal systems, *contradictions are healthy occurrences* in the conceptual memory, because they offer a form of feedback to a process which is concerned with generating many *probabilistic* inferences: the memory is not in search of just one truth! Augmentation is empirically the most common result of evaluation and is very important because it represents the addition of new information to the memory. But it is uninteresting from the evaluation function's standpoint.

I will describe intuitively what each of these five cases signifies. In the next sections, I will explain how each of the cases is recognized by the inference evaluator.

#### 7.5.2.1 DIRECT CONFIRMATIONS

When some new information can be found to confirm some other piece of existing information directly, it can mean one of two things: if the information it confirms has a high enough STRENGTH, the new information is simply reaffirming something the memory is already "pretty certain of". If, however, the information confirms something which has a fairly low strength, it is a far more significant event. In general, this will be an indication that new evidence has appeared for some "guess" (probabilistic inference) the memory has made in the past. Typical of this is the case where the memory has predicted some future state or action which subsequently turns out to be true. For example, hearing (1) below, action prediction inferences will be called into play to determine what Mary is likely to do, given her current WANTs. One line of predictions is that she will go to the store, doing all the necessary actions. These predictions made, (2) is perceived as confirming one of these action predictions:

1. Mary needed some eggs.
2. She got in the car.

Another very common source of direct confirmations arises from the process of causal chain expansion.

#### 7.5.2.2 DIRECT CONTRADICTIONS

When the new information can be found to contradict some old factual information, if the STRENGTHs of the two pieces are *approximately equal*, a conflict exists. This is a hint that the line of inference has gone far enough: either it has uncovered a genuine contradiction, some probabilistic inference has turned out not to have been correct, some incorrect reference decision has been made, or even an incorrect meaning graph has been given to the memory. In any event, inferencing should be discontinued on this line of inferencing, and the conflict noted.

If, on the other hand, the *new* information has a clearly higher STRENGTH than the old, it would seem correct to retain it on the inference queue as a potentially interesting line of inference, and lower the strength of the old. If the strength of the *old* is clearly higher than the new, it would again seem appropriate to discontinue the new line of inference, and perhaps lower the strength of the new inference.

What would be a reasonable thing to do to a structure which has clearly been overridden by another contradictory inference? We are on the limits of the theory at this point; this is simply a difficult question. Clearly, the memory should not simply erase the overridden structure: this is intuitively incorrect from a psychological point of view. Rather, it would seem most appropriate to "rule it out of the picture" in a way which would still retain the structure for future reference. In the memory, the way to do this is by severely decreasing the overridden information's STRENGTH. But by how much? And should this demotion in strength apply only to the overridden structure, or to other structures which arose from it and from which it arose (OFFSPRING and REASONS)? I have made some tentative decisions which I will describe, but they are highly speculative.

### 7.5.2.3 "NORMALITY" OF THE NEW STRUCTURE

Intuitively, when the new information neither directly confirms not contradicts other existing explicit information, but nevertheless *conforms* to the memory's knowledge of what is normal in the world (which, recall, is principally encoded in N-molecules), the new information is likely to be "uninteresting". The idea I want to capture is, roughly speaking, that if *any* inferences at all must be *cut off* from the inference queue, those which strongly confirm the memory's knowledge of what is normal should be the first to go:

> **Less cognitive processing should be devoted to information which is highly normal according to the memory's model of normality.**

"Normal" here is used in the sense described in the discussion of N-molecules. In the evaluation sequence, checks against normality are performed only after attempts to discover confirmations and contradictions with *explicit* knowledge have not yielded results.

On the other hand, when the new information *deviates* from the memory's knowledge of what is normal (that is, it is assessed with low compatibility by an N-molecule), the potentials for making interesting discoveries is intuitively greater. Since language's centralmost function is to communicate new or unusual relationships, the memory should have some sort of awareness about what is unusual, and use that awareness to heighten the amount of cognitive processing it devotes to the information. In terms of the inference control structure, this means that if *any* inferences are cut off from the inference queue, those which deviate from what the memory believes to be normal should be the *last* to go.

### 7.5.2.4 AUGMENTATION

When the new information neither confirms nor contradicts explicit knowledge, and its normality cannot be assessed, the new information should simply remain as a new structure in the memory, and exert no particular influence on the inference control structure. The memory has simply heard something new. Empirically (in the program), more new information currently falls into this category than is ultimately desirable, because, as we will see, the evaluator's powers are not yet very highly developed.

Let us turn now to a discussion of the problems involved with detecting confirmations and contradictions, and to the problems of judging how "normal" a new structure is.

### 7.5.3 DETECTING CONFIRMATIONS AND CONTRADICTIONS

**How does the evaluator detect when newly inferred information directly confirms or contradicts some other information structure in the memory?**

#### 7.5.3.1 THE FIRST PROBLEM: COMPATIBILITY OF OCCURRENCE SETS

The simplest form of direct confirmation is of course to discover another structure which stores a bond identical to the bond of the new structure. Therefore, the first step in searching for confirmations is to locate any other structures with the same bond. Similarly, the simplest form of contradiction of structure S is to find another structure (NOT S). But there are two rather complex potential mishaps for these simple first steps. *First*, a failure to locate an identical bond via the low level memory search function does not necessarily imply that the new structure does not directly confirm or contradict some existing structure. We will get to this in the next sections. *Second*, even though some confirming or contradictory bond can be located, it will be rare indeed for them to have identical occurrence sets. Because of this, we must consider the problem of *compatibility of two occurrence sets*: if the occurrence sets are incompatible, it is unlikely that the two structures could be referentially identical. That is, even though the structures might have identical bonds, there may be irreconcilable features on their occurrence sets, and these would preclude a meaningful confirmation or contradiction.

The general problem of determining with certainty when two structures reference the same action or state is a complex one, and is not yet very well understood. To know whether or not some member or combination of members of one is incompatible with some member or combination of members of the other will eventually require many heuristics and a better measure of fuzzy compatibility than currently exists. To illustrate the potential problems of occurrence set compatibilities, suppose (Fig. 7-15) the new structure, SX, is "John gave Bill a cigar at the fair", and the bond, X, of this new structure is identical to Y which is the bond of another existing structure, SY. Suppose in addition that SX and SY both have TIME and LOCation features on their occurrence sets, and that these features are compatible with one another. But suppose that

1.*Mary* saw SX occur

2.*Pete* saw SY occur

That is, each of SX and SY has an MTRANS feature on its occurrence set, but one involves Mary, the other Pete. Here, SX and SY obviously have different features, but does this make SX incompatible with SY? Probably not. But what if X and Y represent some relatively short-lived action (this ATRANS is such an example), and Mary is known to have been at the fair and left before Pete ever arrived. Clearly, SX and SY could not represent the same ATRANS event.

From this simple but typical example, we observe (a) that the compatibility of features of occurrence sets can involve features of other entities arbitrarily distant from the two structures under examination, and (b) that a large number of special case heurisics would be required to recognize such "subtle" interactions as these. To be completely certain of the referential identity of the two structures would involve tremendous quantities of computation. This is the type of problem which also can pose difficulty even to human

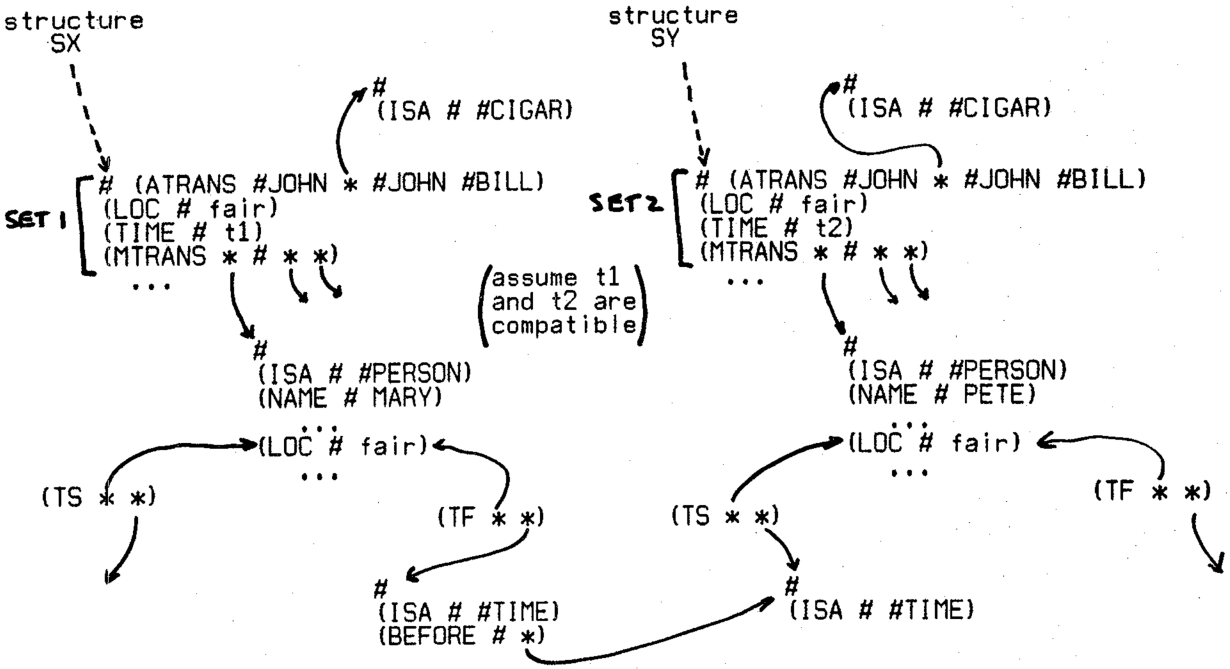



**Figure 7-15. When are two occurrence sets compatible with each other?**

language users. For the most part, tests on a very few common features of action and state structures will suffice to determine the compatibility or lack thereof of two structures. In the current implementation, the only two classes of occurrence set features checked for compatibility

are *time* and *location*. That is, if another structure can be found whose bond matches the bond of the new structure in one of the ways to be described, and the locations and times are compatible, the current memory considers the two structures identical and merges them into one.

The ways in which the location features can be compatible are the following:

1. one or both structures have no explicit LOCation feature
2. both have LOCation features, and their values are identical
3. both have LOCation features, but one or both are UNSPECIFIED
4. both have LOCation features X and Y, and (LOC X Y) or (LOC Y X) (that is, one is a more general location which is compatible with the more specific one)

For time features, if the two times or time intervals could possibly have been the same, the times are considered compatible. That is, if the structures are actions, then only if the time of one is known to be *strictly before* or *strictly after* the time of the other are the times considered incompatible. If the structures are states, then only if their time intervals can be shown *not* to overlap are the times considered incompatible. This is a crude heuristic and obviously needs considerable refinement; fuzzy matching of times poses a major topic of research all its own. The basic problem is one of durations, and how close in time two structures must be in order to stand a chance of being referentially identical. In a typical case the times, T1 and T2, of the two structures will have only the very loose relation shown in Fig. 8-16: they are both after some particular point. But the after relationship could represent microseconds or centuries the way things are handled by the evaluator currently.

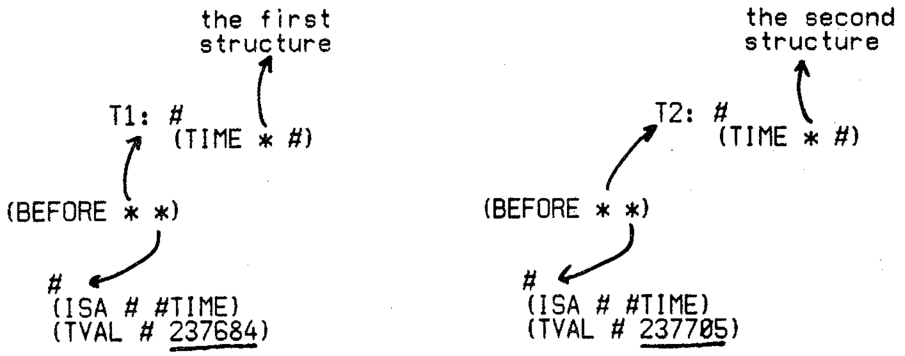



**Figure 7-16. Typically, time relations in the memory are sparse.**

The current plans are to apply the state-duration inference capability to the problem of deciding when two states have occurred at times close enough together to be considered time-wise compatible. Also, since one very common source of confirmation is the matching of a *prediction* about the future to what *actually* occurs in the future, another valuable extension would be to impose approximate time limits on the applicability of all predictions. Thus, if John needs some nails, an action prediction is that he might go to a store, and perform all the intermediate actions to do this. But these predictions only apply to the very near future (say the rest of the day), and should not match up several weeks later.

The compatibility of occurrence sets is a problem of detecting both confirmations and contradictions. Let us now turn to the problems involved with detecting when the *bond* of the new structure matches the *bond* of another structure closely enough for the two structures to be potentially referentially identical structures.

### 7.5.3.2 DETECTING CONFIRMATIONS: THE PROBLEMS OF MATCHING BONDS

As mentioned, even if a lookup fails to locate another bond which is identical to the bond of the new structure, this does not imply that no confirmation exists. There are several reasons for this, and each relates to the need for a certain amount of tolerance -- fuzziness -- in the process which recognizes confirmations. The particular forms of fuzziness the memory is on the lookout for are the following:

1. Some conceptual predicates are symmetric. This can easily prevent a successful retrieval of some bond by low level-lookup functions which are not sensitive to symmetry. For example, (PHYSCONT X Y) directly confirms (PHYSCONT Y X), even though there is not a strict *structural* match between these two structures. The evaluator must be sensitive to this simple kind of "fuzziness".

2. Often, the structures exist at different levels of *specificity*, but mean essentially the same thing. For example, if an inference is made that Mary feels a *positive emotion* toward John around time T1, and another structure exists which represents that Mary feels *love* toward John around T1, the two structures stand a good chance of referencing the same state, and should be recognized as a confirmation:

(MFEEL #MARY #POSEMOTION #JOHN)
(MFEEL #MARY #LOVE #JOHN)

Another example of this is when a very general action prediction of the form (DO P X) is made. It would be desirable that the evaluator recognize this general prediction as matching any appropriate *specific* action which P might subsequently perform, provided the occurrence sets of the general and specific action structures are compatible.

3. If one or both of the structures contain an UNSPECIFIED entity, but match in other respects, and if the features (occurrence set) of the unspecified entity are compatible with the features of the corresponding entity in the other structure, then the chances are also good that the two structures reference the same action or state. For example, if the action prediction inference "John will get some nails from *someone*" arises, and subsequently another structure "John is getting some nails from *Pete*" comes about, it would be desirable to recognize the probable direct confirmation:

| | |
|---|---|
| (ATRANS #JOHN C2247 C2396 #JOHN)<br>(ATRANS #JOHN C2247 #PETE #JOHN) | C2247 represents some nails, C2396 some unspecified person. |

4. It is quite common that two distinct tokens which in reality represent the same entity will appear in two otherwise identical structures. The above example about nails illustrates this: the prediction is that John will acquire *some* nails (no particular ones). But the same indefiniteness is present in the confirming structure "John is acquiring *some* nails from Pete. Although it makes little sense to ask whether the nails which John *wanted* are the "same" nails he is *getting* from Pete, it would be desirable to recognize that the two distinct tokens for these two sets of nails reference basically the same entity. Otherwise, the confirmation would be missed:

| | |
|---|---|
| (ATRANS #JOHN C2247 #PETE #JOHN)<br>(ATRANS #JOHN C6511 #PETE #JOHN) | where C2247 represents the nails John WANTed, and C6511 represents the nails John is getting from Pete. |

5. It is possible in the memory for two completely different structures to represent essentially the same information. Although the use of conceptual primitives has reduced this problem to manageable levels, one gains the feeling that it will never be completely solvable. An illustration of this will occur in an inference-reference interaction example in section 8.1. In that example, the two information units

(TS #ANDY1 #1JUN48)
(TS #ANDY2 #7MAR72)

exist in the memory. That is, one Andy started to exist (was born) at time #1JUN48, and another at time #7MAR72. In the example, the problem is to discover which Andy might be the referent of an ambiguous reference. During inferencing, a crucial discovery is that whichever one it is, his age obeys:

(AGE X #ORDERMONTHS)

Clearly, this *relative* information tends to agree with with the *absolute* TS information about #ANDY2 more than with the TS information about #ANDY1, who is much older. The problem is that the AGE inference has nothing to say about *absolute* times, but is essentially the same as the TS information about #ANDY2 *if it is made during the years 1972, 73 or 74.* It is highly desirable that the AGE inference be recognizeable as representing essentially the same information as the TS for #ANDY2.

### 7.5.3.3 DETECTING CONTRADICTORY BONDS

What are the effective procedures for determining when the bond of a newly-inferred structure contradicts the bond of some other structure in the memory? I am *not* concerned with an elaborate probing ahead to determine whether X would *eventually* contradict some Y which already exists, since the new structure will eventually lead to that point in its expansion by inference anyway. Rather, all the problems of detecting contradictions concern whether two bonds are contradictory *in themselves*, not in what they imply.

Currently, the memory can detect a direct contradiction between new structure, S, and some other memory structure, X, in any of the following forms:

1. S and X are identical propositions, except one is true and the negation of the other is true. That is, both an S and a (NOT S) structure exist.

2. S and X both involve a predicate, P, and, for this P, its conceptual arguments in S contradict its conceptual arguments in X. An example is where (LOC #JOHN #USA) contradicts (LOC #JOHN #FRANCE), given that the times of the two structures are the same or "very close". This kind of test is clearly specific to each conceptual predicate: whereas it is impossible to be in two different locations at the same time, it is entirely possible to POSSess two different objects at the same time: (POSS #JOHN #BALL3), (POSS #JOHN #CAR17).

3. X and Y both involve predicate P, and have arguments which are conceptually opposites (to some degree) of one-another. An example is: (MFEEL #JOHN #LOVE #MARY) vs. (MFEEL #JOHN #HATE #MARY), where (OPPOSITE #LOVE #HATE).

4. X and Y are simply different structures which directly contradict each other. In section 5.7, the computer example ("Mary said she killed herself") showed how an intrinsic enabling condition of Mary's speaking action contradicted an implication of what she said. The contradictory structures had the forms:

```
(TIME #MARY T1)  (that is, Mary was existing at T1)
(TF #MARY T2)    (Mary ceased to exist at T2)
```

where T1 occurred after T2 (Mary still existed after she ceased to exist)! The heuristics which detect this class of contradictions seem to be both predicate-dependent and highly sensitive to the natures of the entities the predicate relates.

### 7.5.4 N-MOLECULES AND THE EVALUATION PROCESS

The need for special predicate-specific heuristics to perform all these relatively involved tests for fuzzy confirmations and contradictions seems to be great. That is, the heuristics which POSS uses to locate contradictory structures will be quite a bit different from those LOC uses, both these will be quite a bit different from those MFEEL uses, and so on. Similarly, the process of detecting confirmations involves heuristics which seem to be quite specific from predicate to predicate. It would be desirable, therefore, to have access to predicate-specific knowledge during these confirmation and contradiction-seeking processes.

The main question is: where should such predicate-specific heuristics exist in the memory? The nature of this task is reminiscent of N-molecules. Recall that the basic purpose of an N-molecule is to encode patterns and normative information in *processes* rather than in complex passive data structures. In other words, when applied to a memory structure, it is the N-molecule's job to rate the structure according to "how reasonable it sounds" in the absence of specific information one way or the other. As described in section 6.7, the N-molecule was sensitive mainly to features of the structure being rated, and to its occurrence set, and only minimally sensitive to other *specific* world knowledge, since it operated under the assumption that unfruitful attempts had already been made to locate specific knowledge which would answer the question directly. But, as we are beginning to see, this is not a good assumption, because the low level retrieval functions search for information on the basis of structural similarity only, disregarding the *meaning* of what they are trying to locate.

Putting enough predicate-specific knowledge to detect fuzzier confirmations and contradictions into N-molecules is more attractive than creating a new kind of process, because it seems to be a proper part of the general task of assessing a structure's compatibility with other knowledge. The generalization which seemed to be needed was to have N-molecules first attempt to relate the structure they are assessing to other specific structures, in search for direct contradictions or confirmations which might have been missed by the *simpler* memory retrieval functions. If a direct confirmation or contradiction could in fact be located, then it should affect the decision of the N-molecule.

#### 7.5.4.1 EVALUATOR/N-MOLECULE COMMUNICATION

In order to do this, we must extend the concept of an N-molecule to one which returns a three-part signal:

1. the judged compatibility of the structure
2. a list of reasons (pointers to other information) explaining why this compatibility was chosen. By convention, when the N-molecule returns a direct response, it returns a single REASON: the structure which was detected to confirm or contradict the structure it was given.
3. the type of discovery which this judgement was based upon: another structure in memory which (a) directly confirms or contradicts S in one of the "fuzzy" ways described in the preceding sections, or (b) a knowledge of what is normal in the world. The latter is the use of N-molecules as already discussed. If the N-molecule can provide neither a direct nor normative response, a failure signal is returned.

Fig. 7-17. illustrates the external appearance of an N-molecule designed to perform these tasks.

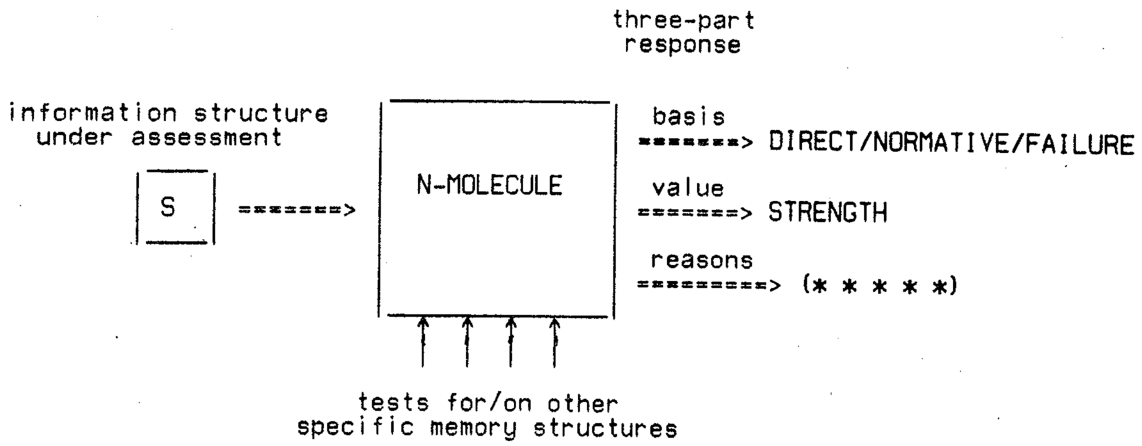



**Figure 7-17. The external logic of an N-molecule.**

This organization allows us to view an N-molecule as a black box which will tell us whether or not some structure S relates in a fuzzy way with some other specific information, and if not, how likely is S to be true, based on a knowledge of what is normal in the world.

### 7.5.5 THE EVALUATION SEQUENCE IN THE PROGRAM

The situations and processing actions which characterize these five cases are summarized in the flow diagram in Fig. 7-18. In order to decide which case is applicable, the evaluator asks the following sequence of questions and performs the associated actions for each newly-inferred structure, S:

1. does S directly match some other structure in the memory, and are their occurrence sets compatible? If so, call the structure merger to merge the two structures, increase the strengths of structures lying along both lines of inference which have been joined, and add the merged result to the inference queue. Also, record the merge on the list !CONFIRMATIONS.

2. if (1) fails, does S directly contradict (ie. S and (NOT S), or (NOT S) and S) some

other structure whose occurrence set is compatible with S's? If so, and if the strengths of the two structure are within 0.20 of each other, record the pair of contradictory structures on the list !CONTRADICTIONS, and do not place the new S on the inference queue (that is, discontinue the line of inference). Although pairs of !CONTRADICTIONS are simply expressed (passed to the conceptual generator) by the present program, they comprise the beginning point for another entire theory of what to do next. If the strengths are more than 0.20 apart, demote the strength of the structure with the lower strength, the strengths of its OFFSPRINGS, and the strengths of its REASONS which are also less than 0.20 below the strength of the higher.

3. if (1) and (2) fail, how compatible is S with the memory's knowledge of what is normal in the world? To answer this question, the evaluator applies the appropriate N-molecule to S. If the N-molecule can, by applying its special heuristics, locate another structure which directly confirms or contradicts S, the evaluator performs the appropriate step (1) or (2). Otherwise, if the N-molecule can assess the new structure's compatibility, C, based on how closely it matches the N-molecule's tests for normality, the quantity (1-C) becomes the inference's significance factor on the inference queue. This implements the heuristic: "the more normal some memory structure is, the less likely it will lead to interesting discoveries." Therefore, if any inferences must be cut off during expansion in inference space, the inference monitor should prefer those with the lowest significance factor.

4. if (1) and (2) have failed, and the N-molecule cannot assess the new structure, it is simply retained on the inference queue, using the default significance factor specified by the inference atom which generated it.

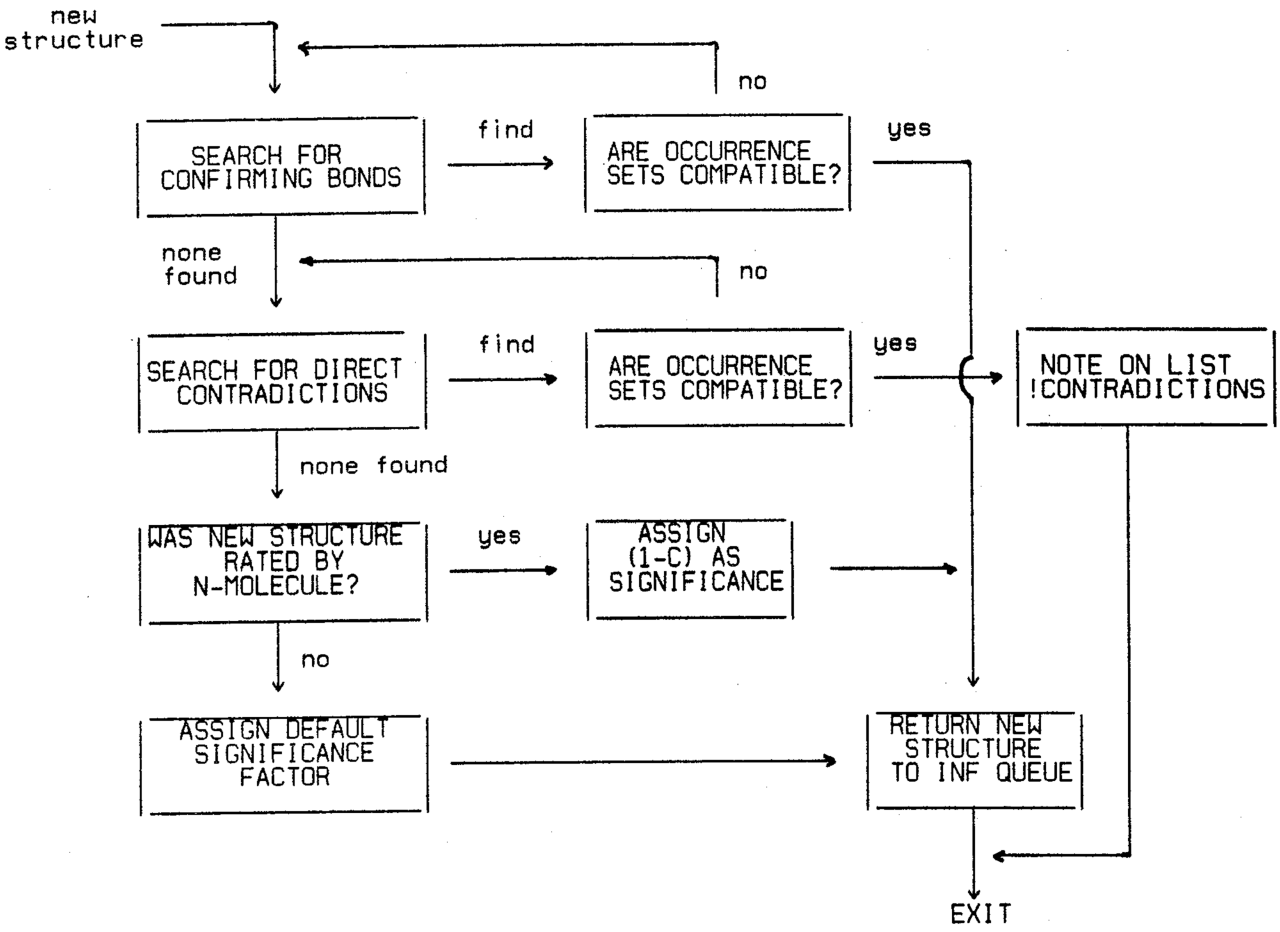



**Figure 7-18. Actions of the evaluator, based on how the new inference relates to other knowledge.**

The problem of detecting confirmations and contradictions, as I have shown, is not a simple one. In fact, future developments will likely prove that the problems I have addressed here are quite inadequate. Furthermore, the operation of the current evaluation function is stymied by the sparseness of N-molecules in the implementation. In the future, a massive N-molecule writing campaign will be necessary to upgrade the evaluator's performance.

### 7.5.6 PROBLEMS, PROBLEMS, PROBLEMS

The inference evaluator is, both in theory and in practice, the single most important component of the conceptual memory. At the same time, it is the most enormous problem I have yet encountered. Its successful operation is seemingly dependent upon hundreds of details which linger on the periphery of the current theory and remain to be identified and solved. For instance, can we reasonably expect a new structure to confirm more than one existing structure, and if so, what is involved in the merge of more than two structures? What should happen when the new structure both confirms existing structures and contradicts other ones? How concerned should we be with the logical consistency of the conceptual memory? How extensively should the occurrence sets of actions and states be examined for compatibility, and what are the features which are most salient ant reliable in this regard? How should confirmations affect the STRENGTHs of the REASONS and OFFSPRING of the confirming structures? The questions are endless, but most interesting.

Also, there are many inadequacies and inconsistencies of the evaluator and merger as they now exist. Many forms of interaction have been ignored in order to implement the basic mechanisms. For example, when a merge occurs between two structures, one or both of which contain UNSPECIFIED entities, and these entities are clarified by merge, they should be identified with their counterparts in the other structure, and merged into one (by an *identity merge*, section 8.1.2) *before* the merge of the two larger structures which are about to be merged. The absence of this interaction with the identity merger currently can lead to results which are simply incorrect. The entire evaluation-merge sequence needs extending and upgrading. All that exists now is a primitive capability for a very interesting and critical process.

### 7.5.7 INFERENCE CUTOFF: IS IT A REAL ISSUE?

Clearly, in a memory very richly endowed with conceptual inferences (for instance, a human language user), there must be some limiting influence on the spontaneous generation of inferences in reaction to each utterance. Otherwise, aside from wasting time on very unlikely or insignificant information, the process might never stop! The same problem exists, but is less severe, for the current modestly endowed memory: when can some line of inference be curtailed with reasonable assurance that "nothing of significance" will be overlooked? Of course, one can

never be absolutely certain; our only aspiration should be to be safe most of the time when a decision to discontinue a line of inference is made.

The question therefore is: what are the criteria for discontinuing a line of inference in such a way as to prevent an "overreaction" to an utterance by the inference process? Of all the potential criteria available, two seem to be the most relevant, and the most obvious: *significance* and *strength*. Intuitively, we would want cutoff to occur when the inferences become too "unlikely", or too "insignificant", or some combination of these abstract metrics. Those structures with high significance should be pursued to lower strengths than those structures of lower significance.

To approximate this idea, the following simple cutoff criterion has been used:

STRENGTH * SIGNIFICANCE < 0.25

where the strength is the STRENGTH property of the structure, and the significance is the quantity 1-C, where C is the compatibility returned by the N-molecule which the evaluator called to assess the new inference's compatibility with memory's knowledge of what is normal. In case no such C could be obtained, the significance is the default significance supplied by the inference atom which generated the inference.

I am trying to capture a very imprecise feeling about cognitive resource allocation by two admittedly crude measures, and somewhat arbitrary numbers. Time will tell how incorrect this simple measure is.

## 7.6 MERGING INFORMATION STRUCTURES: THE STRUCTURE MERGER

**When a direct confirmation is detected by the evaluator, the two memory structures representing this information must be merged into one. How is this achieved in a way which preserves all the surrounding information associated with each. How does this merge physically occur?**

As we have seen, one important goal of the evaluator is to identify some existing structure as representing the same information as each newly-inferred structure. That is, two structures

have been discovered to stand for the same entity in the real world -- concept, token, action or state. At that time, there is a need to merge the two structures into one new structure, because (1) structures in the memory should remain distinct if and only if the real-world notions they represent are distinct, and (2) by merging the two structures into one, new connections are made among other structures which involve them, and this frequently opens new useful paths which will enable more inferences to be made.

This need to merge also occurs when some reference to a token or concept is finally established and must replace the temporary token which was created to stand for it. Although most of what I am about to describe is applicable to this case as well, a later section deals specifically with other issues of merging two tokens -- in particular how the opening of new pathways can result in further inferencing when some missing referent is finally identified and merged with the temporary token which was used previous to identification. In every case, the discovery that the reference identity of two structures is equal is an important event, since it represents a quantum of understanding.

The *mechanical* goal of merging is rather straightforward: that one structure, S, result from two structures, S1 and S2, and that this structure contain all the information from S1 and S2. Equivalently, one structure, S2, is to be merged into the other structure, S1, which is to be preserved in its augmented form. Of primary concern is that all information from both sources be preserved. How does MEMORY effect a merge of S2 into S1?

### 7.6.1 THE MERGE SEQUENCE

The process is illustrated in Fig. 7-19. The first step involves examining each member of S2's occurrence set. For each occurrence set member, M, if the information represented by M about S2 is not also known for S1, M is *modified* so that it describes S1 instead of S2. The alternative to modification would be to create a new unit, M', to describe S1. However, M itself can in general be part of a much larger structure, so that generating a *new copy* of M would in general leave the new copy unrelated to the larger structure in which M participates. This modification involves substituting S1 in M's bond and adding M to S1's occurrence set. If the information M conveys about S2 is also known for S1, it is not copied.

At the end of this process, S1's occurrence set (its features) will represent the union of previous knowledge about S1 and S2. Members of S2's occurrence set which were not transfered to S1 are then purged.

Next, the RECENCYs (times of last memory access by the reference mechanism) are examined and the most recent is assigned as the recency of the merged proposition, S1. The merge process always performs recency and occurrence set merging. However, if the structures represent actions or states (that is, they are information-bearing structures rather than concepts or tokens), the merger continues. S2's REASONS set is appended to S1's REASONS set, and the OFFSPRING set of S2 is appended to the OFFSPRING set of S1. Both of these steps of course involve reflecting the modifications in the reverse links as well, since REASONS and OFFSPRING are inverses of each other. Next, S1's ISEEN set becomes the union of S1 and S2's ISEEN sets, preserving a record of which inference atoms have generated inferences from the information S1 and S2 represent. Finally, S1 receives the "logical-OR" of the TRUTHs of S1 and S2, and the higher of S1 and S2's STRENGTH. At this point, S1 represents the merged result. The merge is then recorded on !MERGELIST and the structural remains of S2 are purged.

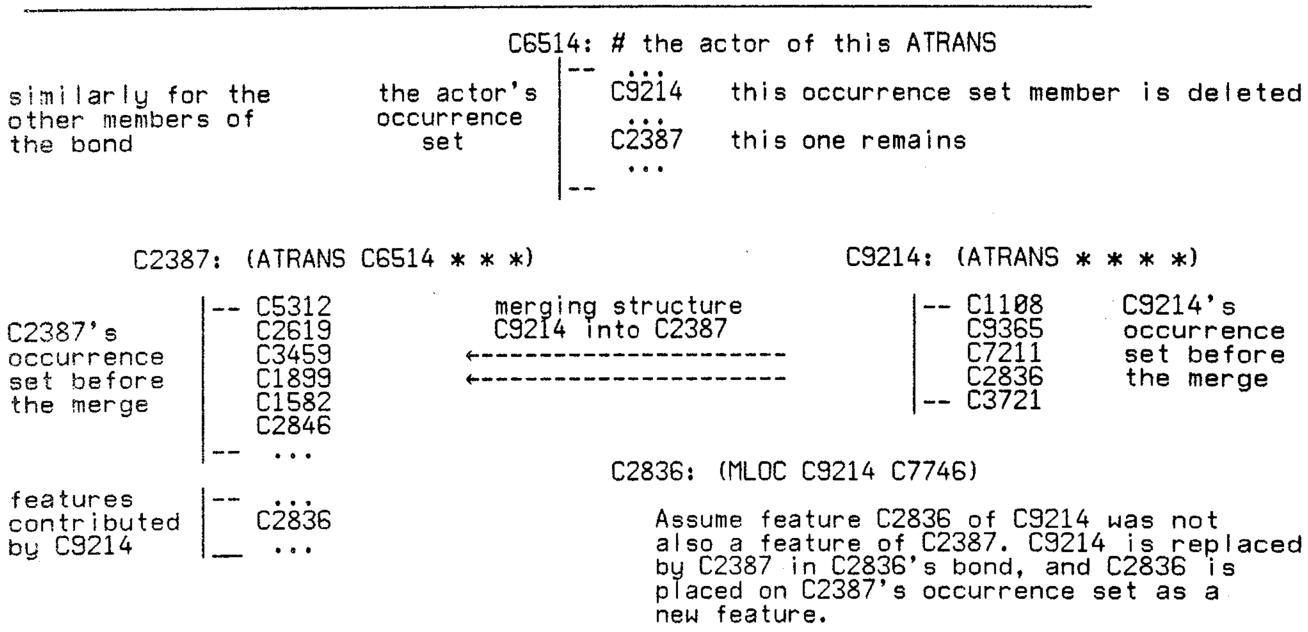



**Figure 7-19. Merging two memory structures.**

This sequence of processing represents the *standard* merge. However, when it is two *concepts* or *tokens*, rather than information-bearing structures, which are being merged, the merge process emcompasses some additional computation.

The following computer example demonstrates a complete standard merge.

## STANDARD MERGE COMPUTER EXAMPLE

In this example, we will observe the point of contact established which completes the causal chain expansion for the utterance "Mary kissed John because he hit Bill." The memory is able to discover the causal chain because this utterance occurs in an environment in which Mary has been inferred to feel a negative emotion toward Bill.

```
(JOHN, MARY AND BILL WERE AT THE PARK)
(BILL TOOK MARYS BOOK AWAY FROM HER)
```

MEMORY reads the first lines of the story. Among other inferences, one which will be generated from the second line is that Mary feels a negative emotion toward Bill because he took her book from her. This will be used during inferencing from the next line to infer that when Mary knows that Bill suffers some sort of negative change, she might become happy.

```
(MARY KISSED JOHN BECAUSE HE HIT BILL)

((CAUSE ((CAUSE ((*PROPEL* (#JOHN1)
(C0067) (#JOHN1) (#BILL1)) (TIME _
(C0070))) ((*PHYSCONT* (C0067)
(#BILL1)) (TIME _ (C0070))))) ((CAUSE
((*DO* (#MARY1)) (UNSPECIFIED _) (TIME
  (C0071))) ((*PHYSCONT* (C0075)
(#JOHN1)) (TIME _ (C0071))))))
(TIME _ (C0070)))

INTEGRATION RESULT: C0089

STARTING INFERENCE QUEUE:
((X 1.0 C0089))
```

MEMORY reads the third line. Its partially integrated result is shown, followed by its integrated result, C0089. C0089 is the main event, so it becomes the input to the inference mechanism. Other subpropositions have been suppressed for this example.

```
                .....

APPLYING INF MOLECULE *PHYSCONT*
TO C0086: (*PHYSCONT* C0075 #JOHN1)
ABOUT TO APPLY @PHYSCONT1 TO C0086
   C0086: (*PHYSCONT* C0075 #JOHN1)
INFERRING:
(*MFEEL* #MARY1 #POSEMOTION #JOHN1)
   ALSO GENERATING: (TIME C0099 C0071)
```

Numerous inferences result from these starting propositions. Eventually, from the fact that Mary kissed John, MEMORY infers that she feels a positive emotion toward him. That is, her feeling a positive emotion toward him caused her to kiss him.

```
                .....

APPLYING INF MOLECULE POSCHANGE
TO C0018: (POSCHANGE #MARY1 #JOY)
ABOUT TO APPLY *POSCHANGE1 TO C0118
   C0118: (POSCHANGE #MARY1 #JOY)
INFERRING:
(*MFEEL* #MARY1 #POSEMOTION #JOHN1)
   ALSO GENERATING: (TS C0125 C0070)
```

More inferences are generated. Eventually, because Mary underwent a positive change (she knew that someone she felt a negative emotion toward suffered a negative change), and since it was John who caused her positive change (by his action of hitting Bill), MEMORY infers that Mary feels a positive emotion toward John.

---

```
CAUSAL EXPANSION ACHIEVED:
(C0078 . C0086)
MERGING:
C0125: (*MFEEL* #MARY1 #POSEMOTION
          #JOHN1)
C0099: (*MFEEL* #MARY1 #POSEMOTION
          #JOHN1)

(*BREAK* . HELLO)
```

MEMORY realizes that the fact that Mary feels a positive emotion toward John has been generated from two distinct sources, and, further, that this fact links two causal chains together, thus explaining a causal relation in the input. At this point we will interrupt MEMORY to see what is about to be merged, and how things look after the merge.

```
*(VOMIT C0125)
-----------------------------------
C0125: (*MFEEL* #MARY1 #POSEMOTION
          #JOHN1)
ASET:
        C0127: (CAUSE C0118 #)
        C0126: (TS # C0070)
RECENCY: 54600
TRUTH: T, STRENGTH: 0.81225000
REASONS:
        C0118: (POSCHANGE #MARY1 #JOY)
        C0078: (*PROPEL* #JOHN1 C0105
                         #JOHN1 #BILL1)
ISEEN: NIL
```

This is one of the structures representing Mary's feeling a positive emotion toward John.

```
*(VOMIT C0099)
-----------------------------------
C0099: (*MFEEL* #MARY1 #POSEMOTION
          #JOHN1)
ASET:
        C0104: (CAUSE # C0086)
        C0101: (*MLOC* # C0055)
        C0100: (TIME # C0071)
RECENCY: 45250
TRUTH: T, STRENGTH: 0.95000000
REASONS:
        C0086: (*PHYSCONT* C0075 #JOHN1)
ISEEN: NIL
```

This is the other. Notice that its REASON is that Mary kissed John.

---

```
*(VOMIT C0125)
-----------------------------------
C0125: (*MFEEL* #MARY1 #POSEMOTION
          #JOHN1)
ASET:
        C0157: (WANT #JOHN1 #)
        C0100: (TIME # C0071)
        C0101: (*MLOC* # C0055)
        C0104: (CAUSE # C0086)
        C0127: (CAUSE C0118 #)
```

This is the merge result. The occurrence sets (ASETs) have been merged, as have the RECENCYs, TRUTHs, STRENGTHs and REASONs. In English, this structure is read: Mary felt a positive emotion toward John. This started at time C0070 and was also known to be true at time C0071. Furthermore, John probably wanted this to be the case. Mary's feeling was caused by C0118 (Bill's suffering a negative

```
        C0126: (TS # C0070)
RECENCY: 54600
TRUTH: T, STRENGTH: 0.95000000
REASONS:
        C0118: (POSCHANGE #MARY1 #JOY)
        C0078: (*PROPEL* #JOHN1 C0105
                         #JOHN1 #BILL1)
        C0086: (*PHYSCONT* C0075 #JOHN1)
ISEEN: NIL
```

change), and her feeling in turn caused C0086 (her kissing John).

## 7.7 A SWIPE AT THE NOTION OF "BACKUP": AN EDITORIAL

A fair question to ask is the following: "How does a conceptual memory cope with mistakes it makes?" I will briefly address the issue here.

"Backup", and sophisticated programming techniques and languages which implement it, have recently become fashionable in Artificial Intelligence. Backup is also called "backtracking" or "non-deterministic programming". These terms refer to the *undoing* of things which have been done by a process. Backup usually occurs in reaction to something which has gone awry, ostensibly because of a "bad decision" along the way. In this context, however, "decision" often refers to no more than a random or "first-option" choice of alternatives, where in fact no real decision criteria were ever applied. Programs which use backup techniques are based on the premise that they will frequently make incorrect decisions because of incomplete knowledge, and would like the final successful version of the processing not to reflect any incorrect decisions which occurred during the search for the final solution.

In many cases, backup is an attractive programming technique simply because it is often easier to undo bad decisions made by some myopic process than it is to give each decision-making process the extensive perspective required to make good decisions the first time. To implement a good backup system, decision points must be explicitly recorded and a record must be kept of which changes to the data were made after which decision points. If enough is recorded, previous states of the program and data can be restored as though nothing had happened.

The main swipe at this notion is this:

**Natural language processing is too complex to rely upon backup AS A PROGRAMMING TOOL. In the common programming sense of the term, there is no such thing as "backup" in the conceptual memory. Backup presupposes that precise decisions are being made along well defined goal paths. It is simply not the purpose of the memory to make all-or-none, precise decisions.**

Human language users rarely "back up" when comprehending language. To study them and discover *why* they don't back up seems to be a far more useful an endeavor than to construct more and more awesome programming languages under the banner of "attaining more power". In my opinion, the high-pressure language vendors have so far done little more than tempt language researchers into shortcut, "slam-bang" solutions with their new, improved programming techniques.

This is not to suggest that the memory assumes it will make no errors in processing the meaning graphs of natural language utterances, nor does it imply that there should be no recourse for altering bad decisions. Rather, it means that there should be no single source of decision points, and no "clean" way to restore the exact state of the program and data after something "bad" happens. These capabilities are simply not desirable for a conceptual memory. What *is* desirable is to make many probable inferences whose utility is ultimately a measure of how they connect with other information in the memory. Those which don't connect simply atrophy; they are not "bad" decisions, but rather conjectures which didn't pan out.

## 7.8 THE MEMORY AS A CONVERSATIONALIST

The memory cannot yet be properly called a "production" program which runs alone, and communicates with the outside world in great abundance; it is still in a fragile experimental stage. However, the current program *does* interface with Riesbeck's conceptual analyzer [R2], and with Goldman's conceptual generator [G1].

Two awesome questions remain to be addressed: what factors should determine when an external response of some sort is called for, and what factors determine the substance of responses? Both questions are beyond my present scope. Nevertheless, we may view the memory as black box with many queues and sources for external responses. By hooking up all these sources to the conceptual generator, we would essentially have an uninhibited low-brow intelligence which constantly babbled its stream of consciousness.

This is roughly what the memory is and does! I will simply list here the sources of information from which responses originate:

1. missing information which the specification process failed to make explicit (!MISSINGINFO)
2. unexplained causal chains (!CAUSALS)
3. comments on causal chains it has explained (!EXPANDED_CAUSALS)
4. contradictions (!CONTRADICTIONS)
5. confirmations (!CONFIRMATIONS)
6. unestablished and remaining ambiguous references (!REFNOTFOUND, !REFDECISION)
7. all inferences. Those which lie below the STRENGTH 0.75 are framed as "I wonder" type questions by including the modifier (MODE (*?*)) in the structure sent to the conceptual generator. Otherwise, a MODE *POSSIBLY*, *PROBABLY*, *CERTAINLY* is included, based on the inference's strength.

## 7.9 REORGANIZING THINGS A BIT

Up to this point in the research, the emphasis has been mainly on defining useful theoretical tasks for a memory, rather than focusing upon efficient ways of organizing large numbers of inferences. The decision was made early to perform as much inference pattern matching as possible *beyond* the point of the simple retrieval of the appropriate inference molecule. That is, it has been convenient to regard an inference molecule as some sort of benevolent black box.

However, as anticipated, experience is showing that it would be helpful to perform more matching in order to determine which subset of inferences organized around a conceptual predicate would be applicable. For example, rather than simply group all PTRANS inferences together under one inference molecule, it would be more efficient to subclassify them into, say, two groups: those involving the PTRANSing of a *person* in one group, those involving the PTRANSing of an *object* in another. This reduces the number of tests which must be made by the more specific inference atoms, and, although it doesn't buy any power, it is a convenient way of avoiding redundant and awkward testing in each molecule. This would also allow, for instance, the selection of relevant inference atoms on the basis of the time aspects of the structure: often, the nature of inferences the memory must make are quite dependent on whether S *has* occurred, *is* occurring, *will* occur, or is a timeless statement.

Plans for the immediate future are to evolve the organization into a more two-level system. Since the main advantage of embodying inferences in program form is that it is an easy way to implement arbitrarily detailed pattern matching, I want to preserve the idea of an *inference atom* as the last step before making each inference. Yet, there is much that can be filtered out by far simpler tests before the atom applies its detailed tests. I would like to associate these two levels of simpler and more complex tests in the way shown in Fig. 7-20. Instead of simply testing the conceptual predicate of structure S which is under inferencing, S would be filtered through a sequence of simple feature matchers. Associated with each simple test would be a collection of inference atoms containing all the specific tests for applicability beyond the first level of simple matching. In this way, as much special case attention to detail as necessary could still be exercised in the inference atom, but each inference atom could make more assumptions about the environment in which it is called, since it will have been called by a fairly selective general matcher.

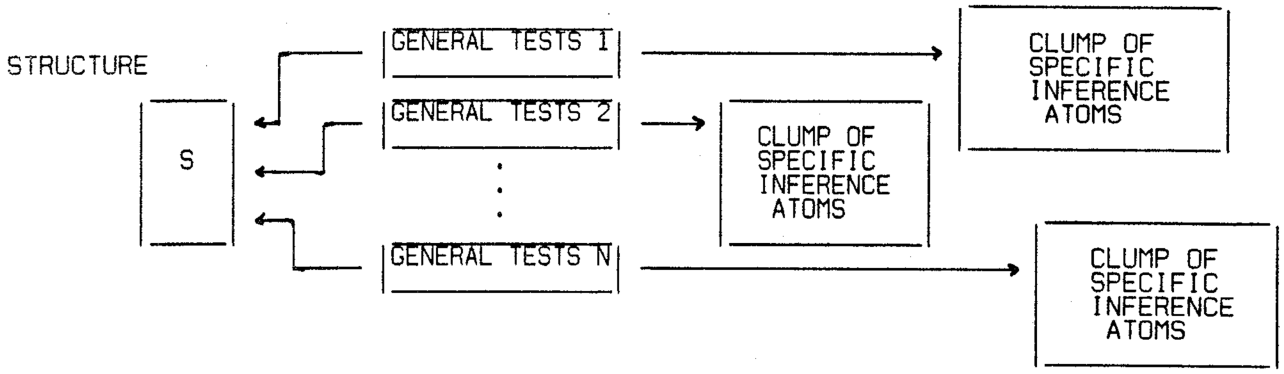



**Figure 7-20. Plans for a more two-level inference organization.**

In Fig. 7-20, all the general tests would be applied in sequence, so that more than one clump of inference atoms might be accessed: the structure might satisfy several of the more general selection tests.

Another extension of the current organization will be to implement an automatic REASONS collection. Currently, each inference atom has a "manually supplied" list of reasons which it

supplies when it generates its inference. But since these reasons are exactly a record of the successful and unsuccessful tests which lead up to the inference atom, it would be desirable for the inference monitor to keep track of the sequence of tests and their results which have been performed within the inference molecule up to the point at which each inference atom generates its inference. However, a completely automatic REASONS collection system will have to await a better defined set of tests than currently exists; for the time being, I want to exercise more control over the reasons list.

# CHAPTER 8

## INFERENCES APPLIED TO REFERENCE ESTABLISHMENT AND TIME RELATIONS

This chapter shows how inferences can be vital to the solution of two subsidiary problems of the conceptual understanding of natural language. Section 7.1 will show how some of the more general problems of reference and anaphora can be successfully attacked with the inference mechanism, and section 7.2 demonstrates the applicability of inferences to solving unspecified time relationships.

## 8.1 INFERENCE AND REFERENCE ESTABLISHMENT

**How does the process of conceptual inference interact with the process of reference establishment? What capabilities are required to make optimal use of this interaction?**

What does a human language user do when he can figure out the *lexical sense* of some word in a sentence (in conceptual dependency terms, he is able to identify a PP), yet isn't instantaneously certain of the real world token to which it refers, its *referent*? That is, if we hear "John made up with Mary yesterday" either out of the clear blue (in no particular context), or in some definite context, how do we decide which of the many #JOHNs we might know the "John" in this sentence references? (The case where *no* #JOHN can be located at all is distinct from, and less interesting than, this problem.) This is a very fundamental aspect of the *problem of reference.* Its solution should be sensitive to as much contextual information as is available. This section presents a theoretical solution to this problem and describes how it has been implemented in the memory. Hopefully, by *synthesis* of a solution, we can shed some light on functions which bear *analytical* reality from the psychologist's point of view.

The following observation has served to define my framework for the solution of references: people simply do not "back up" very often in processing natural language and its conceptual content. (They must be doing something right!) Avoidance of backup during the analysis of a sentence has always been a main tenet of Conceptual Dependency: conceptual world knowledge should be brought to bear upon analysis and integration of natural language utterances so that backup occurs for nothing less than truly conceptual problems. If, for

instance, the conceptual analyzer backs up when most humans do not, it should only be because of a deficiency in the conceptual world knowledge to which the analyzer and memory have access; it should never occur because of sloppy analysis or failures to make use of conceptual context. The same should be true of the process by which referents are identified.

What then do people do? The answer is simple at an abstracted level: they use everything available at the time to figure out the referent. This is not a very useful answer, so this section is devoted to some details of the problem. It should be clear that MEMORY is capable of effectively filling out the circumstances surrounding an utterance, and that it would be desirable to interface with this ability in solving problems of reference.

To this end, a reference capability has been developed which permits the *deferrment* of referent identification when necessary. This results in the ability to proceed with other aspects of conceptual processing even though all referents may not have been established before this processing begins. This scheme also provides for the eventual establishment of these referents as another goal of the inference process, and has the potential for recovering from incorrect identifications and doing so without loss of information or comprehension. The conclusion I will draw is that, in general, the solution of the reference problem for some concept involves arbitrarily intimate and detailed interaction with the inferential processes of the memory, and that these processes must be designed to function with concepts whose features may not be completely known at the time.

### 8.1.1 AN ILLUSTRATION

I will illustrate this inference-reference interaction by following a very simple, "clean" example. In this example, assume the memory knows of exactly two tokens, MC1, MC2 such that

```
X ∈ {MC1,MC2}:
        (D1)    (ISA X #PERSON)
        (D2)    (NAME X "ANDY")
```

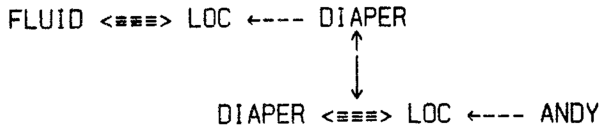
that is, there are two tokens in MEMORY each of whose occurrence set contains the information that the token stands for a person whose name is Andy. (Elswhere, these would have been illustrated as #ANDY1 and #ANDY2, but to avoid confusion in this section, these forms will not be used.)

However, probably in addition to much other information (many other structures on MC1's and MC2's occurrence sets), assume it is also known that:

```
    (TS  MC1  #7MAR72)
and (TS  MC2  #1JUN48)
```

that is, MC1 started to exist March 7, 1972, and MC2 started to exist June 1, 1948. The concepts here represented as #7MAR72 and #1JUN48 are actually time tokens having as TVALs points on MEMORY's "absolute" time scale. These descriptive symbols are used here for clarity only.

Now, suppose the sentence "Andy's diaper is wet."




("there is fluid located at the diaper which is located at Andy", a close-enough approximation for our current needs) is perceived. This is a typical reference dilemma: no human hearer would hesitate in the correct identification of "Andy" in this sentence using the available knowledge about these two Andys. (Let us assume no previous context for this example). Yet the obvious order of "establish references first, then infer", even though intuitively the correct order of processing, simply leads to an impasse in this case. In order to begin inferencing, the referent of "Andy" is required (ie. access to the features of MC1 -- its occurrence set-- in memory), but in order to establish the referen of "Andy" some level of deduction must take place. This would seem to be a paradox.

Actually, the "paradox" stems from the incorrect assumption that reference establishment and inferencing are distinct and sequential processes. The incorrectness of this assumption is a good example of the ubiquitous theme that *no* aspect of natural language processing, (from acoustic phonology to story comprehension), can be completely compartmentalized. In reality, referent identification and inferencing are in general heavily functionally dependent upon each other. Realization of this leads to an interesting sequence of processing capabilities which will untangle and solve this dilemma.

At the point the reference problem is undertaken, the state of the conceptualization "Andy's diaper is wet" (omitting times) is the following (represented in descriptive set notation):

```
(LOC C1: {(ISA C1 #FLUID)}
     C2: {(ISA C2 #DIAPER)
          (LOC C2 C3: {(ISA C3 #PERSON)
                       (NAME C3 "ANDY")})})
```

In other words, there is some fluid located at the diaper which is located at (worn by) a person whose name is Andy. Section 4.3.2.2 described how the correct LOC relation between "Andy" and "diaper" was inferred during the conceptual analysis.

Let us step back a moment and look several steps ahead by describing how the other references in this utterance will fall into place. Once the correct "#ANDY" has been identified, the referent of "diaper" can be established. That is, "the diaper", occurring out of context with no conceptual modification is referentially ambiguous (hence, we might be motivated to inquire "What diaper?"), whereas "the diaper *located at X*" is a signal to the referencer that the speaker has included what he feels is sufficient information either to identify or create the token of a diaper being referenced.

The reference to the concept #FLUID is simply solved: the concept #FLUID is drawn out by the analyzer as part of the definition of what it means conceptually to be wet, and MEMORY simply creates a token of this mass-noun concept. The referencer realizes that references to mass nouns frequently occur with no explicit conceptual modification, and does not bother to identify them further unless contradictory inferences result from them later on. The token of #FLUID created stands for the fluid which is currently in this person's diaper.

Back to the main problem! Using the reference search procedure described in section 4.2, MEMORY uses the descriptive set for C3 shown above to locate MC1 and MC2 as possible candidates for the referent of C3. When recency considerations fail to disambiguate, no more can be done to disambiguate at that point. MEMORY therefore *creates a new concept*, MC3, (which may or may not turn out to be temporary) whose starting occurrence set consists of the conceptual features which lie in the intersection of all candidate's occurrence sets. This, of course, includes at least the descriptive set which has served to locate the candidates. In general, the intersection will be large; however, in this simple example, we assume the intersection to be just this descriptive set, the resulting new structure being:

```
MC3:  #
      (ISA # #PERSON)
      (NAME # ANDY)
```

In addition, MEMORY notes that this concept has been created as the result of an ambiguous reference by adding MC3 to the global list !REFDECISION. This done, a token of a diaper which is located at MC3 can then be created. Ideally, This token too, by virtue of its referencing another possibly incorrectly identified concept in MEMORY (MC3), should be subjected to reference reevaluation, pending identification of MC3. For instance, if MEMORY were to hear "John's bike was broken." and makes the wrong identification of "John", the subsequent identification of "John's bike" will certainly be wrong and will have to be changed when "John"'s referent is. MEMORY currently does not attempt this.

At this point, MEMORY has an internal form of the conceptualization containing a tentative reference. Inferencing may therefore begin. Of interest to this example is the subproposition "a diaper is located at MC3." This situation is an example of where an explicit-peripheral subproposition which is incidentally communicated plays a major part in the understanding of the entire conceptualization: one feature inference MEMORY can make with a high degree of certainty from

```
(LOC X Y)
where (ISA X #DIAPER) and (ISA Y #PERSON)
```

is that the person at (on) which the diaper is located is an infant, namely:

```
(AGE Y #ORDERMONTHS)
```

#ORDERMONTHS is a fuzzy duration concept.

During inferencing therefore, the inferred structure (AGE MC3 #ORDERMONTHS) augments MC3's occurrence set, and other inferencing proceeds. Eventually, all inferencing from the utterance will cease.t or be cut off by At that point, !REFDECISION is consulted and MC3 is detected as having been unestablished, so the second (and subsequent)-pass reference solver, SOLVE_REF, is entered.

SOLVE_REF examines MC3's occurrence set, collecting those members which came into existence on the most recent pass of inferencing. These are detected by comparing RECENCYs of the new members with the value #NOW, stored before MEMORY began any processing of this conceptualization. The new members are first sorted according to the reference relevance heuristic mentioned in section 4.2.1. Then, for each new proposition, each candidate associated with MC3 is examined, checking for confirmation or contradiction. Candidates which have contradictory features on their occurrence set are immediately excluded. Confirmations count in a candidate's favor by the amount of the reference significance associated with the predicate of the confirming structure.

This done, SOLVE_REF selects the candidate with the highest score. When there are still ties, the surviving candidates are reassociated with MC3 on !REFDECISION, hopefully with fewer candidates than before. Subsequent lines of a story or a MEMORY-generated external question can solve these in the same manner. This means, therefore, that

> **The memory can defer identification of referents for as long as necessary without losing information.**

The worst that can happen is that MEMORY will fail to understand certain things by not having immediate access to some concept's occurrence set. This appears to approximate what people do: a human language user can listen to an entire sequence before who or what it is about dawns on him. At the point that happens, he nevertheless can reconstruct what he heard and re-apply it to the newly-discovered referent. Most of the time this is a "micro-process", in that the referent can be identified almost instantaneously -- usually during the analysis of the utterance into conceptual form. Nevertheless, the process is the same whether it is "micro" or protracted over a longer sequence.

I have yet to describe what occurs when the reference candidate set is narrowed to one, as in our example. After the first pass of inferencing in this example, information will be available which resolves the reference ambiguity: the AGE proposition is recognized by the confirmation process as matching the TS proposition stored on the occurrence set of MC1. This gives MC1 a higher score than MC2 in SOLVE_REF. MC3, the temporary concept, has thus been identified as MC1, one of the two candidates.

Upon identification, MEMORY must have the ability to *merge* the temporary concept into the identified concept. In this example, this means that MC3's occurrence set, which possibly has collected other *new* information (now known to reference MC1) which was not used in the identification (ie. it *augments* MC1), must be merged with that of MC1 to preserve any additional information communicated by the input or its inferences. After merging, the temporary concept, MC3, is purged and removed from !REFDECISION.

But now, MC1's (possibly augmented) occurrence set containing all old knowledge about MC1 is accessible since MC1 has been identified. This means that new inferences, which could have been successful had they had access to MC1's full feature set, may now be applicable. To illustrate that this in fact happens in people, consider the following scenerio: assume that we know of two people named "John", John Smith and John Doe, that we know that Bill and John Smith are arch-enemies and that Bill and John Doe are the best of friends. Further, assume we know that Smith owns a car, Doe does not. Suppose that in this situation we hear the following sentence, and from it make the response shown:

INPUT: Bill wrecked the car John loaned him yesterday.
RESPONSE: Oh, oh. I bet there's going to be trouble!

Here, the reference to "John" is undetermined when the first pass through the inference network begins. A temporary concept is therefore created to stand for this reference. After the inference network silences, one new piece of information (probably among many others) is that, whoever the referent of "John" is, he owns a car. This was generated by an enabling inference in which is encoded the knowledge that for someone to loan (underlied by ATRANS) someone else an object, the loaner must have ownership of that object. This inference serves to identify John Smith and rule out John Doe as the referent of "John" by the process we have just seen. However, now John Smith's occurrence set, and in particular the fact that Bill and John Smith are archenemies, is available. Because of this, a second pass through the inference network would enable the resultative inference:

> "if P1 has a negative relationship with P2, and P2 causes a NEGCHANGE of P1 on some scale (regardless of intent), then P1 is liable to do something that would cause a NEGCHANGE for P2 on some scale".

to be made. This is the inference that underlies the response above.

The point is that this inference turns up *only on the second inference pass* because only then was John Smith's full occurrence set available to inferences which could make use of it. The general principle is, therefore, that as long as references on !REFDECISION continue to be narrowed or solved, another pass of inferencing (on the original conceptual structures and all of their first-pass inferences) should be performed (without duplicating work done on the first pass). The hope is that new inferences will be generated which will in turn help to solve more references.

Notice that even the narrowing of a reference (decreasing the number of candidates) constitutes progress, since then the intersection of the remaining candidates' occurrence sets will in general increase, thus associating with the temporary referent *more common features* of the candidates. Theoretically, this process should be iterated until no new inferences arose.

This section has shown that solving references by waiting until the spontaneous generation of inferences on the structures involving those references can be quite fundamental to understanding. A good way to end is to suggest that this process is so automatic in day-to-day speech that we tend to overlook it. Last night Linda asked me: "Are you going to fix Andy's thing?" I must have made the (unconscious) inference that, whatever "thing" was, it was broken, because broken things need fixing. This enabled me to identify "thing" as Andy's chair, which I knew had been broken earlier that evening. There was no "conscious" deduction; the processes of inferencing and referencing simply "did their thing".

### 8.1.2 ADDITIONAL MERGE PROCESSING AT REFERENT IDENTIFICATION: IDENTITY MERGE

A conceptual memory should have the ability to preserve a record of each reference identification it makes. There are two reasons for this. First, it enables references to be made *to the process of identification*, something which the "fact that" test (section 3.2) indicates should be referenceable. For instance, we might hear "Before I realized it was John *Smith* you were talking about, ...". This utterance clearly makes reference to the time at which the process of reference identification occurred. Without recording the identification of a referent as an explicit event, there is no possibility of comprehending an utterance such as this.

The second reason is that there must be recourse for undoing reference blunders. This

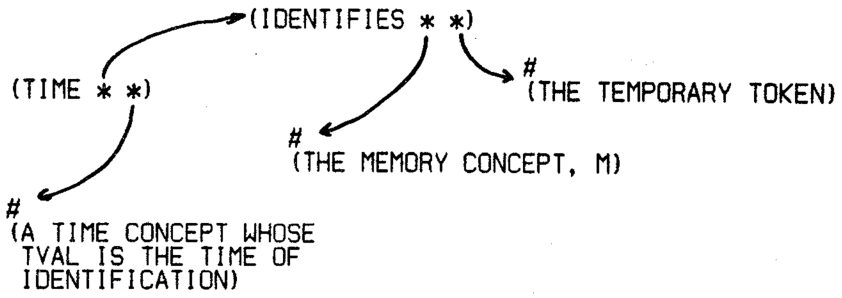
includes the ability to track down the effects of changing a reference after it has been involved in inferencing. Although the notion of backup has been deemphasized, and the current implementation of MEMORY does not have the ability to detect reference errors and back up, the embryo of this capability exists, and is embedded in the process of merging.

However, when the merge is being used to culminate the identification of a referent, the merge is called an "identity merge" and some additional things happen. First, the occurrence set of the temporary token, T, which is being merged into the identified memory token, M, is copied and saved (under the property SAVEDASET) as it exists at the time of merging. At that time, T's occurrence set consists of the original descriptive set and any inferences which were generated and which enabled the identification to be made.

The merge process will then combine T's occurrence set with M's, purge it, then delink all of T's other connections with MEMORY. However, T is not then purged. Rather, it receives a new occurrence set consisting of one feature:

(IDENTIFIES M T)

that is, the relationship which makes explicit that T has been identified by M in MEMORY. At that point, T has only this occurrence set and the feature SAVEDASET to define it. In addition, the time of the identification is recorded on this IDENTIFIES structure's occurrence set. The finished structure looks like:




By doing this, the *process* of reference identification can be referenced by a pointer to this IDENTIFIES structure, since it is just another structure in MEMORY. In addition, if the identification

is later found to be in error (for instance, it lies on a path leading to a contradictory inference), the SAVEDASET can be resurrected, and reused by some process which relocates and reexamines other reference candidates.

There are many issues I have only touched upon, and others I have completely ignored in this section. This scheme is just the beginning outline of a larger capability, and much research on these processes clearly remains to be done. This "identity merge" merely hints at one approach to a solution.

### 8.1.3 AN EXTENSION TO THE REFERENCE MECHANISM

The concept of an N-molecule came relatively late in the delevopment of the memory, as the need to perform "fuzzier" matching became more and more apparent. Gradually, the meaning-sensitive N-molecule has supplanted or augmented lower level memory retrieval calls to locate information on a structural similarity to the desired information.

The reference mechanism as I have described it provides yet another logical source of interaction with N-molecules. Rather than locate referents of descriptive sets by attempting to locate candidates with features which *directly* match features of the descriptive set, a more general approach would be to apply appropriate N-molecules to each feature in the descriptive set for each reference candidate. By doing this, the full power of fuzzy matching encoded in the N-molecules would be available to the referencer, and instead of a simple tally of the number of features on the descriptive set each candidate satisfied, the criterion for selecting one candidate over the rest would be based on a cumulative tally of *compatibilities* returned by N-molecules for each feature relative to each candidate.

For example, suppose some descriptive set references a male named John who owns a hammer, and a simple intersection search locates several Johns on the basis of the NAME, ISA and SEX features of the descriptive set. But none are explicitly known to own a hammer. The simple referencer I have described might fail to select one over the rest. But suppose one of the John's (John Smith) is a carpenter. If a human language user hears this reference, even though he also may not explicitly know that Smith owns a hammer, he might conclude that owning a hammer "simply sounds more appropriate" for John Smith than for any of the other candidate

Johns, because of other things he knows about Smith. He might therefore use this information to clear up the ambiguous reference.

But recall that this is just the sort of task N-molecules were designed for: in this example, the task posed to the OWN N-molecule would be the following (suppose C2317 is a token of a hammer):

RATE THE STRUCTURE (OWN C2317 Pi)
FOR EACH CANDIDATE, Pi, IN THE CANDIDATE SET OF "JOHNS"

Since the OWN N-molecule knows that P's profession might be relevant when trying to assess the likelihood that P possesses a certain kind of tool, this specific heuristic can have its effects in signaling Smith as the most likely candidate: although it is quite common that an adult male own a common shop tool such as a hammer, the likelihood is heightened if his profession involves the use of such a tool. In other words, "John Smith owns a hammer" is slightly more compatible with the N-molecule's knowledge of normality than "John Jones owns a hammer", etc.

By keeping a tally of compatibilities for each feature in the descriptive set as it applies to each candidate rather than a simple YES-NO tally of structurally matched features, the referencer could make far more sophisticated reference decisions, and do so in a way closer to the way I imagine a human language user does. This N-molecule/referencer interaction is one of the extensions planned for the immediate future.

# REFERENCE COMPUTER EXAMPLE 1

This example demonstrates reference-inference-reference interaction for the input "Andy's diaper is wet." MEMORY knows two people named Andy, one of them an infant, the other an adult. At the time the structure which represents this sentence is internalized, MEMORY does not know which Andy is being referenced. MEMORY thus creates a new token, X, to stand for this referent and copies all features which are common to both candidate Andys to it as its occurrence set. MEMORY can then initiate inferencing. One inference which is triggered by the information that there is a diaper located on X is that X is likely to be an infant. This, along with others, is therefore generated as an inference. After inferencing, MEMORY recalls that X's identity is pending solution, so it returns to the problem of establishing the reference. At that time, however, X's occurrence set contains the information that X is an infant. This solves the ambiguity, so MEMORY merges X into the token for the correct Andy. The normal process is to continue with a second pass of inferencing after such a successful reference attempt. The second computer example will illustrate a case where this results in the generation of information which was not possible on the first inference pass.

It might be reemphasized that this example illustrates the solution of a "micro-reference", rather than a missing reference which is protracted over several lines of a story. That is, it is only during the very first phase of understanding (first inference pass) that MEMORY lacks the referent of Andy. However, the process which permits deferring the choice of referent is general enough to handle protracted "macro-references".

---

```
ANDYS DIAPER IS WET

((ACTOR (FLUID REF (*A*)) <=> (*LOC*
VAL (DIAPER => ((ACTOR (DIAPER) <=>
(*LOC* VAL (ANDY))))))) TIME (TIM00))

(TIM00 ((VAL T-0)))
```

This example illustrates MEMORY's ability to defer the identification of some referent until more is known about it via inference. MEMORY previously knows about two people by the name of Andy: Andy Rieger, a baby, and Andy Smith, an adult. The representation of age has been simplified by using an AGE predicate with a fuzzy duration concept.

---

```
COPYING COMMON FEATURES
TO C0030 FROM (#ANDY2 #ANDY1)

(*BREAK* . HELLO)
```

The first step after discovering that "Andy" could refer to either #ANDY is to create a new token (C0030) to stand for whoever it is. This token then receives all features which are common to both candidates, #ANDY1 and #ANDY2. This new token and all candidates for its identity are then recorded on the list !REFDECISION. We interrupt processing here to examine this new token and its two candidates.

```
-----------------------------------
C0030: NIL

ASET:
   C0035: (*LOC* C0033 #)
   C0032: (ISA # #PERSON)
   C0031: (NAME # ANDY)
RECENCY: 6050
```

C0035 is the new information about a diaper being located on this person named Andy. C0031 and C0032 are the common features.

```
-----------------------------------
#ANDY1: NIL

ASET:
   I0012: (AGE # #ORDERMONTHS)
   I0011: (SURNAME # RIEGER)
   I0010: (ISA # #PERSON)
   I0009: (NAME # ANDY)
RECENCY: NIL
```

Here is Andy Rieger as MEMORY knows him.

```
-----------------------------------
#ANDY2: NIL

ASET:
   I0016: (AGE # #ORDERYEARS)
   I0015: (SURNAME # SMITH)
   I0014: (ISA # #PERSON)
   I0013: (NAME # ANDY)
RECENCY: NIL
```

Here is Andy Smith.

```
--------
((*LOC* (C0028) (C0033))
(TIME _ (C0036)))

C0037
```

This is the sentence's partially integrated result. C0028 is a token of some #FLUID, C0033 is a token for the diaper which is located on C0030, some person named Andy. C0037 is the integrated MEMORY structure for this input.

```
--------
STARTING INFERENCE QUEUE:
((X 1.0 C0037) (X 1.0 C0035)
(X 1.0 C0032) (X 1.0 C0031))

(*BREAK* . HELLO)

C0035: (*LOC* C0033 C0030)
C0032: (ISA C0030 #PERSON)
C0031: (NAME C0030 ANDY)

*PROCEED
```

The starting inference queue consists of this main structure, together with all other propositions MEMORY knows about this unidentified reference, C0030. These are subjected to inferencing in the hope that one or more of them, in addition to the main structure, will lead to inferred information which will clear up the reference. All other subpropositions have been suppressed for this example.

```
--------
      ......

APPLYING INF MOLECULE *LOC* TO
C0035:   (*LOC* C0033 C0030)

ABOUT TO APPLY @LOC2 TO C0035
C0035: (*LOC* C0033 C0030)

INFERRING: (AGE C0030 #ORDERMONTHS)
   ALSO GENERATING: (TIME C0040 C0039)

      ......
```

MEMORY generates inferences from these starting structures. C0035, that C0030 has a diaper on, leads to the inference that C0030 is a baby, namely, that his age is #ORDERMONTHS (at the current time, C0039).

Other inferences are generated. Finally,

```
(*BREAK* . HELLO)
```

inferencing ends. At this point, we examine The state of #ANDY1, #ANDY2 and C0030. #ANDY1 and #ANDY2 have of course thus far been unaffected by this input.

```
----------------------------------
C0030: NIL

ASET:
   C0040: (AGE # #ORDERMONTHS)
   C0035: (*LOC* C0033 #)
   C0032: (ISA # #PERSON)
   C0031: (NAME # ANDY)
RECENCY: 5300
```

Here is C0030. Notice the new inferred age information, C0040.

```
----------------------------------
C0035: (*LOC* C0033 C0030)

RECENCY: NIL
TRUTH: NIL, STRENGTH: NIL
OFFSPRING:
   C0041: (TIME C0040 C0039)
   C0040: (AGE C0030 #ORDERMONTHS)
ISEEN: (⊗LOC2)
```

This is the subproposition that a diaper is on this person named Andy. Notice its offspring set contains the inference about C0030's age.

---

```
*PROCEED

RETRYING REFERENCE:
        (C0030 #ANDY2 #ANDY1)


REFERENCE AMBIGUITY SOLVED.
        OLD: (C0030 #ANDY2 #ANDY1)
        NEW: #ANDY1
```

We allow MEMORY to continue. After inferencing has died out, by looking at the list !REFDECISION, MEMORY discovers that a reference identification is pending. This is shown at the left.

Scanning for new information about C0030, the new AGE information is discovered. It is further discovered that this settles the reference, since it matches I0012 (#ANDY1's age information) exactly. Such a clean match will not in general result, and the powers of the evaluation function and N-molecules must in general be called upon to determine whether two structures "match" each other.

```
MERGING:
        #ANDY1: #ANDY1
        C0030: C0030
```

Having identified C0030 as #ANDY1, MEMORY merges C0030 into #ANDY1.

```
(*BREAK* . HELLO)
```

Finally, we examine #ANDY1 and C0030 after the merge.

```
----------------------------------
#ANDY1: NIL

ASET:
   C0042: (IDENTIFIES # C0030)
   C0035: (*LOC* C0033 #)
   I0012: (AGE # #ORDERMONTHS)
   I0011: (SURNAME # RIEGER)
   I0010: (ISA # #PERSON)
   I0009: (NAME # ANDY)
RECENCY: 5300
```

Notice that the new information about a wet diaper has been associated with #ANDY1.

```
----------------------------------
C0030: NIL

ASET:
```

The merge process unlinked C0030 from MEMORY, recorded what was known about it, and identified it as #ANDY1.

```
   C0042: (IDENTIFIES #ANDY1 #)
SAVEDASET:
   (AGE # #ORDERMONTHS)
   (*LOC* C0033 #)
   (ISA # #PERSON)
   (NAME # ANDY)
RECENCY: 5300
```

---

The original structure which now involves #ANDY1 can be subjected to inferencing again, in hopes that new inferences will result by virtue of this new access to #ANDY1's occurrence set.

## REFERENCE COMPUTER EXAMPLE 2

This computer example illustrates reference-inference, reference-inference interaction (two inference passes). Hearing the input "Bill saw John kiss Jenny.", MEMORY is unable to decide upon the referent of "Jenny": it could be Jenny Jones or Jenny Smith. MEMORY therefore creates a temporary token having as features all the common features of Jenny Jones and Jenny Smith. By inference, MEMORY is able to decide upon Jenny Jones. At that point, the temporary token is merged into the concept for Jenny Jones, and a second pass of inferencing is initiated. However, on the second pass a new inference arises: because Bill loves Jenny Jones, and he saw John kiss her, he (probably) became angry at John. This inference was not triggered on the first inference pass because being loved by Bill was not a common feature of both Jennys, and hence not accessible then (ie. it had not been copied to the temporary token's occurrence set).

The example begins with a few lines to set the scene for MEMORY. Inferencing on these setup lines (which is normally spontaneous) has been suppressed for the sake of simplicity in this example.

---

```
--------------------
JOHN WAS IN PALO ALTO YESTERDAY
((*LOC* (#JOHN1) (#PALOALTO))
(TIME _ (C0001)))
C0002

--------------------
JENNY JONES WAS IN PALO ALTO YESTERDAY
((*LOC* (#JENNY2) (#PALOALTO))
(TIME _ (C0004)))
C0005

--------------------
```

This example illustrates reference-inference, reference-inference interaction. That is, MEMORY is unable to establish a reference, so it creates a temporary token, and proceeds with inference. Inferencing generates new information which solves the reference, so more inferencing can be undertaken. However, because features of the referent are accessible on the second inference pass, new inferences are possible.

To the left, MEMORY is reading in some

```
JENNY SMITH WAS IN FRANCE YESTERDAY
((*LOC* (#JENNY1) (#FRANCE))
(TIME _ (C0007)))
C0008


--------------------
BILL LOVES JENNY JONES
((*MFEEL* (#BILL1) (#LOVE) (#JENNY2)))
C0010

--------------------
BILL SAW JOHN KISS JENNY YESTERDAY

COPYING COMMON FEATURES TO C0015
FROM (#JENNY2 #JENNY1)

((*MTRANS* (#BILL1) ((CAUSE ((*DO*
(#JOHN1) (#UNSPECIFIED)) (TIME
(C0011))) ((*PHYSCONT* (C0012) (C0015))
(TIME _ (C0011))))) (C0018) (C0021))
(TIME _ (C0011)) (INST _ ((*LOOK_AT*
(#BILL1) (C0015 #JOHN1)) (TIME _
(C0011)))))

C0031
```

information which is relevant to this demonstration. Each of these inputs would normally produce inferences as it is processed, but inferencing has been suppressed for the first four sentences of this example. The four sentences are shown with their partial integrations and final structures, C0002, C0005, C0008, C0010.

The synopsis of this short plot is as follows: There are two Jennys: Jenny Jones and Jenny Smith. Bill loves Jenny Jones. John and Jenny Jones were in Palo Alto yesterday, Jenny Smith was in France yesterday. The climax comes when Bill sees John kiss Jenny. It is MEMORY's job to figure out which Jenny. MEMORY will decide upon Jenny Jones, then re-inference and infer that Bill probably got angry at John-- something which wouldn't have happened if Bill had seen John kiss Jenny Smith.

To the left, the climax line is in the process of being read and internalized. Its final structure is C0031. Notice that C0015 was created to stand for some Jenny, and that all common features of the two Jenny candidates were copied to it.

---

```
-----------------------------------
#JENNY1: NIL

ASET:
   I0019: (SURNAME # SMITH)
   I0018: (ISA # #PERSON)
   I0017: (NAME # JENNY)
RECENCY: NIL


-----------------------------------
#JENNY2: NIL

ASET:
   I0022: (SURNAME # JONES)
   I0021: (ISA # #PERSON)
   I0020: (NAME # JENNY)
RECENCY: NIL


-----------------------------------
C0015: NIL

ASET:
   C0029: (*LOOK_AT* #BILL1 #)
   C0026: (*PHYSCONT* C0012 #)
   C0017: (ISA # #PERSON)
   C0016: (NAME # JENNY)
RECENCY: 9866
```

We interrupt MEMORY at this point to have a look at the two Jennys and C0015, the token representing one of these Jennys.

This is the person named Jenny who Bill saw yesterday, and who John kissed. C0012 is the token representing John's lips, which were in *PHYSCONT* with this person named Jenny (C0015) at time C0011.

---

```
STARTING INFERENCE QUEUE:
((X 1.0 C0031) (X 1.0 C0017)
(X 1.0 C0016))

        ......
```

MEMORY begins inferencing from this input. The starting inference queue consists of the main structure for the sentence, together with all other facts known about C0015. In this case, these are simply that C0015 is

a person, and that its name is Jenny. These will not be of use in this example. All other subpropositions have been suppressed from the starting inference queue for this example.

```
APPLYING INF MOLECULE *MLOC* TO
C0037: (*MLOC* (CAUSE (*DO* #JOHN1
        #UNSPECIFIED) (*PHYSCONT* C0012
        C0015)) C0021)
```

One inference from Bill's seeing this event is that he knows that the event occurred. That is, the event went from his eyes to his conscious processor, C0021.
To the left, the inference that Bill knows about John's kissing Jenny is being generated: information in Bill's CP (C0021) will also enter his LTM, C0040. This fact will be of use during the second pass of inferencing (after MEMORY decides that C0015 is Jenny Jones).

```
ABOUT TO APPLY @MLOC1 TO C0037
INFERRING: (*MLOC* C0028 C0040)
   ALSO GENERATING: (TS C0043 C0011)

        ......
```

```
APPLYING INF MOLECULE *PHYSCONT* TO
C0026: (*PHYSCONT* C0012 C0015)
```

Another inference arises from John's lips being in PHYSCONT with C0015: that John feels a positive emotion toward C0015. The structure representing this inference is C0049.

```
ABOUT TO APPLY @PHYSCONT1 TO C0026
INFERRING: (*MFEEL* #JOHN1 #POSEMOTION
                C0015)
   ALSO GENERATING: (TIME C0049 C0011)
```

```
ABOUT TO APPLY @PHYSCONT2 TO C0026
INFERRING: (*MLOC* C0049 C0051)
   ALSO GENERATING: (TS C0054 C0011)
```

Another inference from John's kissing action is that C0015 knows that John feels a positive emotion toward C0015. C0051 is C0015's LTM. This inference will be of no direct consequence in this example.

```
ABOUT TO APPLY @PHYSCONT3 TO C0026
INFERRING: (*LOC* C0015 #PALOALTO)
   ALSO GENERATING: (TIME C0056 C0011)
```

MEMORY also infers from John's kissing C0015 that John and C0015 had the same location at the event time, C0011 (yesterday). Since MEMORY knows that John was in Palo Alto, and has no information concerning C0015's location yesterday, MEMORY infers that C0015 was also in Palo Alto yesterday. This information will solve the reference ambiguity.

```
        ......

APPLYING @POSTSCAN TO C0043:
(*MLOC* (CAUSE (*DO* #JOHN1
#UNSPECIFIED) (*PHYSCONT* C0012 C0015))
C0040)

        ......

INFERRING: (*MLOC* C0049 C0040)
COPYING TIMES FROM C0043 TO C0086

        ......
```

During the postscan inferencing, the fact that Bill saw John kiss C0015 leads to the inference that Bill knows that John feels a positive emotion toward C0015. This inference type implements the principle that if a person knows X, he also is likely also to know the inferences which can be drawn from X. That is, MEMORY assumes that other people possess the same inference powers as MEMORY does.

---

```
C0015: NIL

ASET:
   C0056: (*LOC* # #PALOALTO)
   C0053: (PART C0051 #)
   C0049: (*MFEEL* #JOHN1 #POSEMOTION #)
   C0029: (*LOOK_AT* #BILL1 #)
   C0026: (*PHYSCONT* C0012 #)
   C0017: (ISA # #PERSON)
```

Inferencing eventually ceases. We interrupt processing at this point to examine C0015, the unknown Jenny. Notice the new information which has been built up about C0015.

C0051 is C0015's LTM.

C0012 is John's lips.

```
   C0016: (NAME # JENNY)
RECENCY: 9350
```

---

```
C0056: (*LOC* C0015 #PALOALTO)

ASET:
   C0078: (*MLOC* # C0040)
   C0057: (TIME # C0011)
RECENCY: 42533
TRUTH: T, STRENGTH: 0.90250000
REASONS:
   C0002: (*LOC* #JOHN1 #PALOALTO)
   C0026: (*PHYSCONT* C0012 C0015)
OFFSPRING:
   C0101: (*MLOC* C0024 C0051)
ISEEN: NIL
```

Since it will settle the reference ambiguity, we have a closer look at the structure which represents C0015's being in Palo Alto yesterday (C0011). C0078 represents Bill's knowledge of C0015's location yesterday (but has no direct relevance to this example).

Notice that the reasons for MEMORY believing that C0015 was in Palo Alto at time C0011 are twofold: that John was in Palo Alto at that time, and that a body part of John was in PHYSCONT with C0015 then.

---

```
C0086: (*MLOC* C0049 C0040)
(*MLOC* (*MFEEL* #JOHN1 #POSEMOTION
         C0015) C0040)
ASET:
   C0087: (TS # C0011)
RECENCY: 25750
TRUTH: T, STRENGTH: 0.95000000
REASONS:
   C0043: (*MLOC* C0028 C0040)
ISEEN: (*MLOC2)
```

We also examine the structure which represents the inference that Bill knows that John feels a positive emotion toward C0015. This information will come into play after C0015's identity is solved (on the second inference pass). C0087 indicates when Bill started knowing this fact (C0040 is his LTM).

---

```
RETRYING REFERENCE:
        (C0015 #JENNY2 #JENNY1)

REFERENCE AMBIGUITY SOLVED.

        OLD: (C0015 #JENNY2 #JENNY1)
        NEW: #JENNY2

MERGING:
        #JENNY2: #JENNY2
        C0015: C0015

PURGING: (*LOC* C0015 #PALOALTO)
PURGING: (*MLOC* (*LOC* C0015 #PALOALTO)
                 C0040)
PURGING: (TS (*MLOC* (*LOC* C0015
                 #PALOALTO) C0040) C0011)
PURGING: (TIME (*LOC* C0015 #PALOALTO)
                 C0011)
PURGING: (ISA C0015 #PERSON)
PURGING: (NAME C0015 JENNY)
```

The first pass of inferencing is now finished. We allow MEMORY to proceed. It notices that a reference decision is pending, and attempts to decide between #JENNY1 and #JENNY2 as the referent of C0015 by using newly-inferred information about C0015 (from the first pass). It succeeds, because #JENNY2 was known to be in Palo Alto yesterday, and this matches new C0015 information, C0056.

MEMORY merges C0015 into #JENNY2, purging old information which is not used to augment #JENNY2. Recall that the merge replaces occurrence set pointers, so that every MEMORY structure which referenced C0015 now references #JENNY2.

---

```
#JENNY2: NIL

ASET:
   C0117: (IDENTIFIES # C0015)
   C0026: (*PHYSCONT* C0012 #)
   C0029: (*LOOK_AT* #BILL1 #)
   C0049: (*MFEEL* #JOHN1 #POSEMOTION #)
   C0053: (PART C0051 #)
```

We have another look at #JENNY2 before the second inference pass begins.

```
   C0010: (*MFEEL* #BILL1 #LOVE #)
   C0005: (*LOC* # #PALOALTO)
   I0019: (SURNAME # JONES)
   I0018: (ISA # #PERSON)
   I0017: (NAME # JENNY)
RECENCY: 8950
```

MEMORY begins the second pass of inferencing. This consists of subjecting each inference which arose from the first pass to inference again. The ISEEN property prevents duplication of inferences during second and subsequent passes.

```
RE-INFERRING...

        ......
```

```
APPLYING INF MOLECULE *MLOC* TO
C0086: (*MLOC* (*MFEEL* #JOHN1
          #POSEMOTION #JENNY2) C0040)

ABOUT TO APPLY @MLOC3 TO C0086
INFERRING: (*MFEEL* #BILL1 #ANGER
                #JOHN1)
   ALSO GENERATING: (TS C0119 C0011)

        ......
```

One new inference which was not possible on the first pass is that Bill probably became angry at John. This inference arises from Bill's knowing that John feels a positive emotion toward #JENNY2, someone Bill loves. C0119 is the structure representing Bill's incipient anger toward John. The crucial point is that this inference became possible only after #JENNY2's features became available after a reference decision, which was in turn made possible through first-pass inferencing.

```
C0119: (*MFEEL* #BILL1 #ANGER #JOHN1)

ASET:
   C0121: (CAUSE C0086 #)
   C0120: (TS # C0011)
RECENCY: 107600
TRUTH: T, STRENGTH: 0.90250000
REASONS:
   C0086: (*MLOC* C0049 C0040)
   C0010: (*MFEEL* #BILL1 #LOVE #JENNY2)
ISEEN: NIL
```

Finally, we have a look at this second pass inference.

C0121 represents the cause of Bill's anger as being C0086, his knowing about the kissing event, C0049.

Notice the reasons MEMORY believes that Bill became angry at John: he knew John kissed #JENNY2 (this structure is C0049), and he loves #JENNY2.

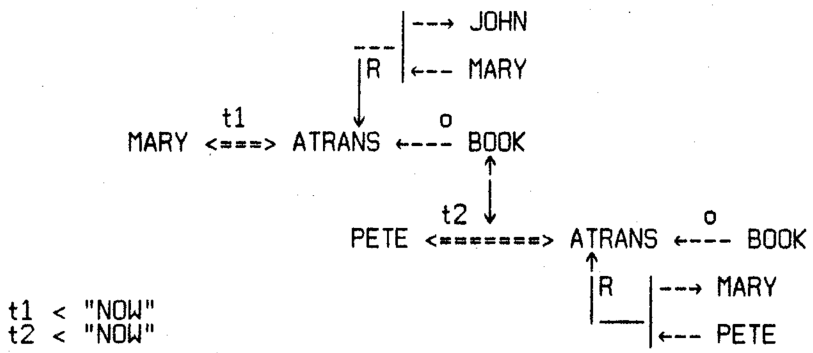
## 8.2 TIME AND INFERENCE

**Exact interrelations among the various times alluded to by an utterance are frequently unspecified. This is a form of missing information which the speaker of an utterance assumes the hearer can infer to obtain a complete meaning graph. How are inferences in conceptual memory relevant to this problem?**

One major theme has been that the notion of spontaneous conceptual inference is crucial to most aspects of understanding. That is, it has become clear that nearly every important aspect of

conceptual memory which involves probabilistic decision-making should be able to interface to one degree or another with the inference mechanism. In most cases this amounts to making the best of an indeterminate state and *deferring* decisions when at all possible until inferences have been called into play. This section discusses another aspect of this principle: inferences are useful for determining missing time relationships within one, or among several conceptual structures.

Consider the sentence "Mary gave John the book which Pete gave her.", for which the analyzer renders the following conceptual graph:




Here, the *analysis* yields only that the two times, t1 and t2, are both in the past of "NOW". The conceptual analyzer has no need here to ascertain the more detailed relation (that t2 must have been before t1) *for the purposes of analysis.* However, the *complete meaning graph* of this sentence cannot truly be said to exist until the relations among the various times in the graph have been determined as accurately as possible in the given context. In this case, a relatively precise relationship between t1 and t2 can be established through inferencing.

During the process of internalizing this graph from the analyzer, the following two subpropositional structures (among others) are extracted, and are put on the initial inference queue:

```
S1: (ATRANS #MARY C1 #MARY #JOHN) (at time t1)
S2: (ATRANS #PETE C1 #PETE #MARY) (at time t2)
```

where C1 is the book. Inferencing will thus occur on these two propositions simultaneously. We will assume for simplicity that all references have been successfully identified.

The solution to this time relation comes about via two resultative inferences which will lead from independent sources to the same proposition, causing two lines of inference from different sources to establish a point of contact in inference space.

In this example, a resultative inference which can be made with certainty from S1 is that Mary must have had a book immediately before the ATRANS action, and in particular, that she *ceased* to have it at t1:

I1: (TF (POSS C1 #MARY) T1)

(There are, once again, many other inferences which are not pertinent to this illustration.) Likewise, one resultative inference which arises from S2 is that Mary *began* having the book at t2:

I2: (TS (POSS C1 #MARY) T2)

Whichever of these two inferences, I1,I2, is generated second will cause the following event: the (POSS C1 MARY) generated by the second will be detected by the evaluation function as *confirming* an existing piece of knowledge (the first (POSS C1 MARY) generated as an inference). Hence, the evaluation function calls the merger to knit these two propositions together into the following single one:

```
# (POSS C1 #MARY)
(TS # T2)
(TF # T1)
   ...
```

that is, a structure representing a state with a TS and a TF, possibly in addition to much other information. Since t2 is now known to be the starting time of a state whose ending time is known to be t1, it is directly deduceable that t2<t1 by the time proof procedures. This relation was not ascertainable before inferencing.

# CHAPTER 9

## CONCLUSIONS, FUTURE WORK

### 9.1 NUTSHELL

Language, memory, and conceptual inference are inseparable notions. My thesis, briefly, is that there is a stratum of cognition in which large quantities of inference-based computations occur spontaneously from each thought to which the human brain attends. The existence of such a stratum would help explain much of the observable behavior we classify as "language comprehension", or simply "understanding".

### 9.2 SPECIFIC CONCLUSIONS

I have mapped out and implemented the beginning of what I hope to be a comprehensive theory of understanding by conceptual inference. Since the theory is "synthetic" rather than "analytic", can I justify drawing any conclusions from it about how human memory works? Probably not. Nevertheless, it is possible to summarize by way of the following "educated guesses" about memory and understanding.

1. Understanding is not possible without large quantities of conceptual inference. Humans must perform many spontaneous, meaning-based inferences on each thought they perceive in order to relate that thought to their models of the world and to other thoughts perceived in the same situation. "Understanding" can be defined in terms of conceptual inferences, and how the inference sphere around one meaning graph interacts with the spheres of inferences around other meaning graphs.

Inferences are *all* probabilistic, and must be made in seemingly wasteful quantity, even if only a very small number of them eventually interacts with other information in the memory. The existence of "weird contexts" for any given thought stands as evidence for this claim: what is a relevant or salient inference from thought T in one circumstance may be quite an irrelevant inference from T in another circumstance, and there is no a priori way of deciding without making exploratory inferences in hopes of discovering interesting interactions with other knowledge.

2. Based upon how it processes and understands language, human memory must be regarded as a highly "volatile" entity rather than a passive one. I submit that, *at a subconscious level*, hundreds of inferences of the natures proposed in chapters 5 and 6 are generated from each thought we perceive. Only when interactions between spheres in inference space occur do we become aware of the underlying processing. The notion of the conceptual expansion of meaning units in inference space bears a direct analogy to the notion of free-associative concept-concept activation in "concept space".

3. The "gestalt" of a thought (meaning graph) can only be captured by simultaneously exploring inferences from *all* subpropositions of the thought. Again, what might be a totally insignificant unit of information in one context might be extremely critical in another.

4. It is theoretically important to systematize conceptual inferences: to partition them into classes, based upon their utility in the understanding process. All classes of inference are always potentially applicable to all inputs. For certain sub-goals of the general expansion of the input in inference space, it is important that the memory be able to restrict inferences to one or several classes (to "multiplex" inferences). This can be imagined as a more directed type of reasoning which occurs during the general expansion.

The number of inference classes is manageably small -- perhaps no more than 30. Sixteen of the most important classes have been examined and incorporated in the computer model.

5. Prediction and specification (the filling-in of missing conceptual information) are powerful mechanisms of understanding. By making explicit probabilistic predictions about why each actor may have performed an action, what actions he might reasonably be expected to perform next, what the predictable results of an action were (will be), and so forth, the chances for discovering crucial implied relations are enhanced.

6. "Understanding", as the word is used, say to define how we process the information in a story, is simply the composite of many different kinds of inference: to understand is to uncover as many implied relations between an input and other information as possible. There is no black and white measure of understanding. For this reason, it is for the most part meaningless to talk about "backup" in the context of a conceptual memory. Probabilistic inferences which don't "pan out" are assumed simply to atrophy (disappear completely or become inaccessable) with time.

There are perhaps classes of special problems whose solutions rely on some sort of "backup thinking", but they do not provide insights into *general mechanisms* of cognition.

**7.** There is a strong interaction between the process which locates (deduces) the memory referents of language entities and the process of conceptual inference. This observation is by no means new (see [W5], [C1] for example), but has been dealt with here in a new way. This interaction is a form of relaxation processing, in which conceptual inference can infer features of an unidentified entity, thus clearing up its identity, or further restricting the set of candidates. On the other hand, having narrowed the candidate set, or having succeeded completely in identifying the referent, new features become available (the referent's occurrence set, or the then-larger intersection of all candidates' occurrence sets), and these new features can lead to further conceptual inference. This interaction occurs both as a "microprocess" which takes place as the thought containing the references is initially perceived (analyzed), and as a protracted process (say, over the duration of an entire novel).

Reference decisions are generally not made until there is little doubt about their correctness. The idea of a descriptive set allows the memory to use as much conceptual information from as many sources as possible in the identification process.

**8.** Inferences in a meaning-based theory of understanding can be conveniently structured around conceptual primitives. By using a system of meaning primitives rather than dealing directly with language, or with even a syntactic-semantic analysis of language, the memory (and human memory) can function in a pure meaning environment, without the additional burden of the syntactic and lexicographic variablility of each thought. Inferences can be organized in multiple-response discrimination-net-like structures beneath the conceptual primitives in a way which avoids time-consuming searching for relevant inferences. Inferences themselves are *active* (program) entities rather than passive patterns and templates.

**9.** The maintenance of time relations is crucial to the understanding process. The frame problem becomes a "non-issue" if low-level memory retrieval functions have the ability to access state duration inferences and update time-sensitive information *only as the need arises*. That is, my theory predicts that the majority of the information stored in the human brain is "out of date" until it is accessed again. The process of accessing it automatically updates the

information's temporal features and its strength of belief based on those features. The process of observing thus alters that which is observed.

**10.** The ability to assess the "normality" of a given meaning graph is important. It provides an essentially infinite amount of knowledge by allowing the memory to make educated guesses about the probable truth of information which is not explicitly stored. By encoding this knowledge of normality in active normality molecules, the assessment can be made quite sensitive to features of entities involved in the meaning graph being assessed. (That is, normality information stored this way can be tailored to accomodate situations which are known to be abnormal in specific respects.) A knowledge of what is normal also plays an important role in determining -- in a general way -- which inferences are likely to be the most fruitful to pursue from any given meaning graph.

## 9.3 GENERAL CONCLUSIONS

One general conclusion I have reached is that conceptual meaning primitives provide a very powerful and realistic approach to language. The memory truly does function in "pure meaning", as that phrase is defined by the nomenclature of chapter 2. Syntax and traditional semantics are implicated only at the outermost (input/output) level of perception, and are never seen at this deeper level of understanding.

I believe it is not unrealistic to assume that all of language (ie. all thoughts communicable by language) can be represented by a shockingly small number of conceptual primitives. Having done this research, it is my belief that the number of conceptual links will not exceed, say, 50, and that the number of meaning primitives, properly systematized, will not exceed several hundred. Furthermore, a system based upon these primitives could be natural and convenient to work with at this "compressed" level. At the beginning I expected to be depressed by the magnitude of the representation problem. Instead I have been encouraged.

## 9.4 DISAPPOINTMENT

One disappointment which occurred early in the research was my inability to encode process and data in the same "homogeneous" memory structures in a way that would make them indistinguishable, except for the way some other process happened to use them. I quickly discovered that the questions I wanted to ask were at too high a level to frame within the constraints of such a representation. I still believe that this should be done, and regard its absence as the system's major weakness, particularly since its absence precludes most forms of "learning". The development of a single, homogeneous data structure to accomodate all the ideas in this thesis will remain as one long-range goal.

## 9.5 IMPLICATIONS FOR PSYCHOLOGY AND AI

By this thesis I hope to have demonstrated (1) that the Artificial Intelligence framework is a valid one from which to conduct inquiries into questions of how humans use and understand language, and (2) that researchers -- psychologists in particular -- should not shy away from the AI point of view simply because it might lead to conclusions which are less tangible than those obtainable by direct laboratory experimentation. Modeling and experimentation must proceed hand-in-hand.

Some researchers have made this important commitment to approach problems of language from both the AI and psychology points of view. Of particular encouragement from the psychology side are such works as Anderson and Bower's research into human associative memory [A5], Rumelhart, Lindsay and Norman's proposals for a process model of long-term memory [R3], and Colby's computer simulation of artificial paranoia [C5]. In addition, Abelson's work with belief systems (scripts, superscripts and his structuring of notions of causality, motivation, enablement and purposes) [A1] are particularly refreshing, and served as inspiration for many of my views on language. From the AI side, Winograd's integration of syntax, semantics and world knowledge into a system for understanding language [W5], Quillian's semantic memory [Q2], Becker's model of intermediate level cognition [B1], and Charniak's model of the mechanisms of children's story comprehension [C1], have all demonstrated that AI is an effective framework from which to attack language.

Certainly these researchers are not in full agreement with all of the ideas of this thesis. Neither do I agree fully with them. It should be clear, for example, that I believe that Quillian's approach is too word-based, that the Rumelhart-Lindsay-Norman approach and Charniak's approach lack a much-needed formal system of *conceptual* representation, that Colby's non-inferential system relies too heavily upon stimulus-response theory, and that Winograd's system, although it represented a quantum advance in language processing, was overly syntactic and dealt with an overly restricted domain. However, the overall approach to language by all these researchers -- through detailed computer models -- is fundamentally correct. Language and memory -- indeed, *all of AI* and memory -- are inseparable, and this realization should be adopted as the underlying theme of all AI research in the years to come.

## 9.6 FUTURE QUESTIONS

One large lingering question concerning the conceptual memory is: what really happens when the memory has 50,000 inference atoms instead of 50? My intuition is that, as the system grows, *fewer* inferences will be recognized as applicable to any given meaning structure if they are well organized in their respective discrimination nets: the nets simply will become more discriminating! In other words, increasing the number of potential inferences will not lead to a combinatorial explosion. On the other hand, more and better heuristics for *cutting off* the expanding sphere of inferences around each meaning graph will have to be developed before the system can be called "practical" instead of "toy".

Finally, there is an irresistable analogy to be drawn between expanding spheres of inference in inference space, and expanding "wavefronts of cognition" in the human brain's neural network as suggested many years ago by researchers such as John C. Eccles:

> **"Thus we have envisaged the working of the brain as a patterned activity formed by the curving and looping of wavefronts through a multitude of neurons, now sprouting, now coalescing with other wavefronts, now reverberating through the same path - all with a speed deriving from the millisecond relay time of the individual neuron, the whole wavefront advancing through perhaps one million neurons in a second. In the words of Sir Charles Sherrington, the brain appears as an 'enchanted loom where millions of flashing shuttles weave a dissolving pattern, always a meaningful pattern, though never an abiding one: a shifting harmony of subpatterns'."**
>
> -- John C. Eccles,
> Scientific American, Sept. 1958

# REFERENCES


[A1] **Abelson, R. P.**, "The Structure of Belief Systems," in Schank and Colby (eds.), 1973

[A2] **Abelson, R. P.**, and **Carroll, J. D.**, "Computer Simulation of Individual Belief Systems," Amer. Behav. Sci. 8, pp. 24-30, 1965

[A3] **Adams, J. A.**, **Human Memory**, McGraw-Hill, New York, 1967

[A4] **Amarel, S.**, "On Representations of Problems of Reasoning about Actions," in **Machine Intelligence 3**, Mitchie, 1968

[A5] **Anderson, J.**, and **Bower, G.**, **Human Associative Memory**, Wiley & Sons, New York, 1973

[A6] **Atkinson, R.**, **Bower, G.**, and **Crothers, E.**, **An Introduction to Mathematical Learning Theory**, Wiley, New York, 1965

[B1] **Becker, J.**, "An Information-Processing Model of Intermediate-Level Cognition," Memo 119, Stanford A. I. Project, Computer Science Dept., Stanford Univ., 1970

[B2] **Becker, J.**, "A Model for the Encoding of Experiential Information," in Schank and Colby (eds.), 1973

[B3] **Bobrow, D.**, "Natural Language Input for a Computer Problem-Solving System," in Minsky (ed.), 1968

[B4] **Bower, G.**, "Organizational Factors in Memory," Cogn. Psy., 1(1), pp. 18-46, 1970

[B5] **Burks, A.** (ed.), **Essays on Cellular Automata**, Univ. of Illinois Press, Urbana, Ill., 1970

[C1] **Charniak, E.**, "Toward a Model of Children's Story Comprehension," Doctoral Dissertation, M.I.T., Dec., 1972

[C2] **Chomsky, N.**, **Aspects of the Theory of Syntax**, M.I.T. Press, Cambridge, Mass., 1965

[C3] **Colby, K. M.**, "Simulation of Belief Systems," in Schank and Colby (eds.), 1973

[C4] **Colby, K. M.**, and **Smith, D. C.**, "Dialogues between Humans and an Artificial Belief System," in Proceedings of the 1st Int. Joint Conf. on A.I., 1969

[C5] **Colby, K. M.**, **Weber, S.**, and **Hilf, F.**, "Artificial Paranoia," in Artificial Intelligence, 2, pp. 1-25, 1971

[C6] **Coles, L. S.**, "Talking with a Robot in English," in Proceedings 1st Int. Joint. Conf on A.I., 1969

[C7] **Collins, N.**, and **Mitchie, D.** (eds.), **Machine Intelligence 1**, American Elsevier, New York, 1967

[D1] **Didday, R.**, "The Simulation and Modeling of Distributed Information Processing in the Frog Visual System," Tech. Rpt. 6112-1, Stanford Univ. Center for Systems Research, 1970

[D2] **Dreyfus, H.**, **What Computers Can't Do - A Critique of Artificial Reasoning**, Harper and Row, 1972

[E1] **Eccles, J. C.**, "The Physiology of Imagination," Scientific American, Sept. 1958

[E2] **Ernst, G.**, and **Newell, A.**, **GPS: A Case Study in Generality and Problem Solving**, Academic Press, New York, 1969

[F1] **Feigenbaum, E.**, and **Feldman, J.** (eds.), **Computers and Thought**, McGraw-Hill, New York, 1963

[F2] **Fillmore, C.**, "The Case for Case," in Bach and Harms (eds.), **Universals in Linguistic Theory**, Holt, Rinehart and Winston, Inc., Chicago, 1968

[G1] **Goldman, N.**, "Computer Generation of Natural Language from a Deep Conceptual Base," Doctoral Dissertation, Stanford Univ., 1974

[G2] **Goldman, N.**, and **Riesbeck, C.**, "A Conceptually Based Sentence Paraphraser," Memo 196, Stanford A. I. Proj, Stanford Univ., 1973

[G3] **Green, C.**, "The Application of Theorem Proving to Question Answering Systems," Memo 96, Stanford A. I. Project, 1969

[G4] **Green, C.**, "Applications of Theorem Proving to Problem Solving," Proceedings 1st Int. Joint Conf. on A.I., 1969

[H1] **Hewitt, C.**, "Procedural Embedding of Knowledge in PLANNER," Proceedings 2nd Int. Joint Conf. of A.I., 1971

[H2] **Hunt, E.**, "The Memory We Must Have," in Schank and Colby (eds.), **Computer Models of Thought and Language**, 1973

[L1] **Lakoff, G.**, **Irregularity in Syntax**, Holt, Rinehart and Winston, Inc., New York, 1970

[L2] **Lamb, S.**, **Outline of Stratificational Grammar**, Georgetown Univ. Press, 1966

[L3] **Lindsay, R.**, "Inferential memory as the Basis of Machines which Understand Natural Language," in Feigenbaum and Feldman (eds.), **Computers and Thought**, 1963

[L4] **Lindsay, R.**, "In Defense of Ad Hoc Systems," in Schank and Colby (eds.), **Computer Models of Thought and Language**, 1973

[L5] **Luria, A. R.**, **The Mind of a Mnemonist**, Basic Books, New York, 1968

[M1] **McCarthy, J.**, and **Hayes, P.**, "Some Philosophical Problems from the Standpoint of Artificial Intelligence," in **Machine Intelligence 4**, Edinburgh Univ. Press, Edinburgh, 1969

[M2] **McCarthy, J.**, "Programs with Common Sense," in Minsky (ed.), **Semantic Information Processing**, 1968

[M3] **Miller, G.**, "The Magical Number Seven, Plus or Minus Two," Psych. Rev., 63, pp.81-97, 1956

[M4] **Miller, G.**, **Galanter, E.**, and **Pribram, K.**, **Plans and the Structure of Behavior**, Holt, Rinehart and Winston, Inc., New York, 1960

[N1] **Neisser, U.**, **Cognitive Psychology**, Appleton-Century-Crofts, New York, 1967

[N2] **Newell, A.** et al., **Final Report of a Study Group on Speech Understanding Systems," North Holland, 1973**

[N3] **Nilsson, N.**, **Problem Solving Methods in Artificial Intelligence**, McGraw-Hill, New York, 1971

[N4] **Norman, D.** (ed.), **Models of Human Memory**, Academic Press, New York, 1970

[N5] **Norman, D.**, **Memory and Attention**, Wiley and Sons, New York, 1969

[P1] **Penfield, W.**, and **Rasmussen, T.**, **The Cerebral Cortex of Man; A Clinical Study of Localization of Function**, Macmillan, New York, 1950

[P2] **Penfield, W.**, and **Jasper, H., Epilepsy and the Functional Anatomy of the Human Brain**, Little, Brown, Boston, 1954

[P3] **Pribram, K., Languages of the Brain**, Prentice-Hall, Englewood Cliffs, N.J., 1971

[Q1] **Quam, L.**, and **Diffie, W.**, Stanford LISP 1.6 Manual, Stanford A. I. Project Operating Note 28.7

[Q2] **Quillian, M. R.**, "Semantic Memory," in Minsky (ed.), **Semantic Information Processing**, M.I.T. Press, Cambridge, Mass., 1968

[Q3] **Quillian, M. R.**, "The Teachable Language Comprehender," CACM 12 (8), 1969

[R1] **Raphael, B.**, "SIR: A Computer Program for Semantic Information Retrieval," in Minsky (ed), **Semantic Information Processing**, 1968

[R2] **Riesbeck, C.**, "Computer Analysis of Natural Language in Context," Doctoral Dissertation, Stanford Univ., 1974

[R3] **Rumelhart, D., Lindsay, P.**, and **Norman, D.**, "A Process Model for Long-Term Memory," Tech. Report 17, Center for Human Information Processing, Dept. of Psy., Univ. of Cal., San Diego, 1971

[R4] **Rumelhart, D.**, and **Norman, D.**, "Active Semantic NetwoRks as a Model of Human Memory," in Proc. 3rd Joint Conf. on A.I., 1973

[R5] **Russell, S.**, "Semantic Categories of Nominals for Conceptual Dependency Analysis of Natural Language," Memo 172, Stanford A. I. Project, Stanford Univ., 1972

[S1] **Samet, J., Programming Languages**, Prentice-Hall, Englewood Cliffs, N.J., 1969

[S2] **Sandewall, E.**, "Representing Natural Language Information in Predicate Calculus," in **Machine Intelligence 6**, Edinburgh Univ. Press, 1971

[S3] **Schank, R.**, and **Colby, K.** (eds.), **Computer Models of Thought and Language**, W. H. Freeman, San Francisco, 1973

[S4] **Schank, R.**, "Conceptual Dependency: A Theory of Natural Language Understanding," Cogn. Psy. 3 (4), 1972

[S5] **Schank, R.**, "The Fourteen Primitive Acts and Their Inferences," Memo 183, Stanford A. I. Project, Stanford Univ, 1973

[S6] **Schank, R., Goldman, N., Rieger, C.**, and **Riesbeck, C.**, "Primitive Concepts Underlying Verbs of Thought," Memo 162, Stanford A. I. Project, Stanford University, 1972

[S7] **Schank, R., Goldman, N., Rieger, C.**, and **Riesbeck, C.**, "MARGIE: Memory, Analysis, Response Generation and Inference on English," in Proc. 3rd Int. Joint Conf. on A.I., 1973

[S8] **Schank, R.**, and **Rieger, C.**, "Inference and the Computer Understanding of Natural Language," Memo 197, Stanford A. I. Project, Stanford Univ., 1973

[S9] **Selfridge, O.**, "Pandemonium: A Paradigm for Learning," Proc. of the Symposium on Mechanisation of Thought Processes, Nat. Physical Lab., Teddington, Eng., 1959

[S10] **Simmons, R.**, "Natural Language Question Answering Systems: 1969," CACM 13 (1), 1970

[S11] **Simmons, R.**, "Semantic Networks: Their Computations and Use for Understanding English Sentences," in Schank and Colby (eds.), **Computer Models of Thought and Language**, 1973

[S12] **Slagle, J., Artificial Intelligence: The Heuristic Programming Approach**, McGraw-Hill, New York, 1971

[S13] **Smith, D. C.**, MLISP Reference Manual, Stanford Univ. CS-179, 1970

[S14] **Sussman, G.**, and **Winograd, T.** MICROPLANNER Reference Manual, Memo 203, M.I.T. A. I. Lab, 1970

[T1] **Tesler, L., Enea, H.**, and **Colby, K.**, "A Directed Graph Representation for Computer Simulation of Belief Systems," Math. Biosciences 2 (1/2), Feb., 1968

[T2] **Tesler, L., Enea, H.**, and **Smith, D.**, "The LISP70 Pattern Matching System," Proc. 3rd Joint Conf. on AI, 1973

[U1] **Uhr, L.** (ed.), **Pattern Recognition**, Wiley and Sons, New York, 1966

[V1] **Vicens, P.**, "Aspects of Speech Recognition by Computer," Memo 85, Stanford A. I. Project, 1969

[W1] **Weizenbaum, J.**, "Eliza," CACM 9 (1), pp. 36-45, 1966

[W2] **Wilks, Y., Grammar, Meaning and the Machine Analysis of Natural Language**, Routledge and Kegan Paul, Boston, 1971

[W3] **Wilks, Y.**, "Preference Semantics," Memo 206, Stanford A. I. Project, Stanford Univ., 1973

[W4] **Wilks, Y.**, "An Artificial Intelligence Approach to Machine Translation," in Schank and Colby (eds.), **Computer Models of Thought and Language**, 1973

[W5] **Winograd, T.**, "Procedures as a Representation for Data in a Computer Program for Understanding Natural Language," Doctoral Dissertation, M.I.T., 1971

[W6] **Winograd, T., Understanding Natural Language**, Academic Press, New York, 1972

[W7] **Winograd, T.**, "A Procedural Model of Language Understanding," in Schank and Colby (eds.), **Computer Models of Thought and Language**, 1973

[W8] **Woods, W.**, "Transition Network Grammars for Natural Language Analysis," CACM 13 (10), pp. 591-606, 1970

[Z1] **Zadeh, L. A.**, "The Concept of a Linguistic Variable and Its Application to Approximate Reasoning," working paper, Univ. of California, Berkeley, 1973