Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

AN INTELLIGENT TUTORING SYSTEM FOR GRAPHIC MAPPING STRATEGIES

Richard Gordon Feifer

November 1990
CSD-900039

An Intelligent Tutoring System for Graphic Mapping Strategies

Richard Gordon Feifer

February 1989

Technical Report UCLA-AI-89-04

UNIVERSITY OF CALIFORNIA

Los Angeles

An Intelligent Tutoring System for Graphic Mapping Strategies

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Education

by

Richard Gordon Feifer

1989

© Copyright by

Richard Gordon Feifer

1989

TO

MY PARENTS

TABLE OF CONTENTS

Acknowledgements vii
Vita ix
Abstract xi
1. Overview 1
    1.0. Introduction 1
    1.1. Graphic Mapping 3
    1.2. Sherlock Environment 6
    1.3. Sherlock's Input and Output 10
    1.4. Sherlock's Tutoring Strategy 20
    1.5. Sherlock's Architecture 21
    1.6. Motivation for an Intelligent Tutoring System 23
    1.7. Organization of the Thesis 23
2. Analysis of the Graphic Mapping Task 25
    2.0. Introduction 25
    2.1. First Pilot Study 25
    2.2. Second Pilot Study 34
    2.3. Summary 39
3. Representing Text Knowledge 41
    3.0. Introduction 41
    3.1. Declarative Knowledge in Sherlock 42
    3.2. Representing the Consideration Text 47
    3.3. Representing Icon Interpretations 53
    3.4. Summary 57
4. Representing Strategy 59
    4.0. Introduction 59
    4.1. Strategic Knowledge 60
    4.2. Knowledge About Strategy 62
    4.3. Sherlock's Strategy for Constructing a Graphic Map 64
    4.4. Sherlock's Strategy for Tutoring 71
    4.5. Summary 75
5. Spreading Activation 77
    5.0. Introduction 77
    5.1. Activation 78
    5.2. Markers 79
    5.3. Inhibition 80
    5.4. Classification 81
    5.5. Finding a Relationship 85
    5.6. Summary 89
6. Using Rules 91
    6.0. Introduction 91
    6.1. Modelling the use of Strategy 91
    6.2. Modelling Thinking About Strategy 96
    6.3. Summary 104
7. Trace and Implementation of Sherlock 107
    7.0. Introduction 107
    7.1. Determining the Relationship Between Icons 108
    7.2. Determining What Sherlock Would Have Done 111
    7.3. Getting More Information 113
    7.4. Classifying The Learner's Plan 114

7.5. Evaluating The Learner's Beliefs....122
7.6. Providing Feedback....122
7.7. Second Attempt....128
7.8. Implementation Overview....130
7.9. Summary....131
8. Evaluation of Sherlock....133
8.0. Introduction....133
8.1. Methodology....136
8.2. Results....137
8.3. Summary....145
9. Overview of Intelligent Tutoring System Research....147
9.0 Introduction....147
9.1 SCHOLAR....147
9.4 WEST....152
9.5 Anderson's Tutors....155
9.6 Proust....158
9.7 Summary....161
10. Future Work and Conclusions....163
10.1. Limitations and Future Work....163
10.2. Summary and Implications....168
References....171
Appendix A: Post Test from First Pilot Study....177
Appendix B: Data from Second Pilot Study....181
Appendix C: Annotated Protocols....191
Subject: Becky....193
Subject: Janey....213
Subject: Jennifer....227
Subject: Stan....251
Appendix D: Sherlock: the Program....275

ACKNOWLEDGMENTS

I was extremely fortunate to find co-advisors from different departments who were willing to work together and with me. It was Eva Baker's reputation in technology and education that originally motivated me to seek her out as an advisor. Eva had enough faith in me to offer the support necessary for me to return to school full-time. She also had the vision to suggest that I "look into Artificial Intelligence." She has continued to provide me with support and professional guidance through the painfully long process that led to this dissertation.

In looking into Artificial Intelligence, I found Michael Dyer. Michael had the courage to take a chance on an outsider. He welcomed me into the AI lab and the computer science department and offered me the resources, and more importantly the guidance and knowledge, necessary to master a new field. He helped me acquire financial support for a portion of this research and edited every word of several incarnations of this dissertation.

I thank my committee for being patient and caring: Earl Rand, Sandra Graham, and Leigh Burstein. I also thank Richard Shavelson, an original member of the committee, who has since moved on to become the dean of the School of Education at UC Santa Barbara. I owe a special debt to Phil Ender, the computer expert in the Graduate School of Education. He has been my teacher and confidant, and most importantly, a true friend.

It has been my privilege to work with the faculty, staff, and students in the Graduate School of Education. Through them I was exposed to the emerging field of cognitive science, intelligent tutoring systems, and the challenge of evaluating new technologies. Although it is impossible to name everyone, I would like to mention Merlin Wittrock, Aimee Dorr, W. James Popham, Wellford Wilms, Charles Healy, Marvin Alkin, Joan Herman, Nancy Atwood, Pam Aschbacher, Adrianne Bank, Elaine Lindheim, Wanetta Jones, Ruth Paysen, John Novak, Janine Woodward, Anath Jacoby, George Marcolides, and the staff in the Office of Student Services and in the Business Office.

My thesis research was conducted in the UCLA AI Laboratory, the creation of Michael Dyer and Margot Flowers. Much of my progress came through discussion with other lab members. In particular, I owe a great debt to Michael Gasser, now at the University of Indiana. Michael had come to the lab from outside of computer science, like myself, but with the advantage of being brilliant. In addition to writing the network package that is used in my work, we spent hours together discussing issues in knowledge representation and life in general. These discussions were extremely important in shaping the direction of my research.

Another lab member who unselfishly shared his code and expertise was Seth Goldman. His T Flavors Graphics package made possible the interface used in my tutoring system. Other lab members include Sergio Alvarado (now at UC Davis), Erik Mueller (now at Morgan Stanley), Stephanie August, Charlie Dolan, Maria Fuenmayor, Jack Hodges, Valeriy Nenov, Mike Pazzani (now at UC Irvine), Alex Quilici, Walter Read, John Reeves, Ron Sumida, Scott Turner, and visiting scholar Edward Hoenkamp.

Financial support was received for portions of this research from the Lockheed Software Technology Center, Austin, Texas and from the Defense Advanced Research Projects Agency (DARPA), contract number: ONR N00014-86-K-0395

I would like to thank friends and colleagues who read and commented on early drafts of this dissertation: Emanuel Maidenberg, Zach Wechsler, and Katharine Fry. Katharine, in particular, labored over every detail of every draft of this dissertation, and I could not have finished it without her.

I thank the parade of subjects who have used Sherlock and contributed to its growth. A special thanks to Stan, Jennifer, Becky, and Janey.

I have to thank Dr. Joanne Dawson and the staff at the UCLA Student Health Service who kept my body going while I completed this research, Claudia Mazzetti and the American Association for Artificial Intelligence for the computer this dissertation was written on, and Rick Rose and the staff at the ASUCLA Computer Store who kept the computer going.

I thank my family. My parents who made me who I am, and my brother and sister. And my godmother, may she rest in peace, who insisted that I be the best that I could be.

VITA

July 16, 1950	Born, Los Angeles, California
B.A. 1973	Bachelors in Fine Art California State University Northridge, California
M.A. 1976	Masters in Religious Education Hebrew Union College Los Angeles, California
1976-1980	Director of Education Shaare Emeth Religious School St. Louis, Missouri
1980-1983	Director of Education Beth Hillel Religious School North Hollywood, California
1983-1989	Research Associate UCLA Graduate School of Education Center for the Study of Evaluation Los Angeles, California
1984-1987	Research Associate UCLA Computer Science Department Artificial Intelligence Laboratory Los Angeles, California

PUBLICATIONS AND PRESENTATIONS

Feifer, R.G., Dyer, M.G., Baker, E.L. (1988). Learning procedural and declarative knowledge. *Proceedings of the Intelligent Tutoring Systems Conference - 88*, Montreal, Canada.

Read, W., Dyer, M.G. & Feifer, R.G. (1988). What's So Hard About Understanding Language? Paper presented to the American Educational Research Association, 1988 Annual Meeting, New Orleans.

Feifer, R.G., Dyer, M.G., Baker, E.L., & Flowers, M. (1987). SHERLOCK: Identifying tasks in graphic mapping Paper presented at the Third International Conference on Artificial Intelligence and Education, Pittsburgh, PA.

Feifer, R.G., Dyer, M.G., Baker, E.L., Flowers, M., & Read, W. (1986). SHERLOCK: A tutoring system for graphic learning strategies (Technical Report UCLA-AI-86-5). University of California, Los Angeles, Computer Science Department, Artificial Intelligence Laboratory.

Feifer, R.G., Dyer, M.G., Baker, E.L., Flowers, M., & Read, W. (1985). *Teaching inferencing strategies* (Technical Report UCLA-AI-85-5). University of California, Los Angeles, Computer Science Department, Artificial Intelligence Laboratory.

Baker, E.L., Aschbacher, P., Feifer, R.G., Bradley, C., & Herman, J. (1985). *Intelligent computer-assisted instruction study* (JPL contract #956881). University of California, Los Angeles, Graduate School of Education, Center for the Study of Evaluation.

ABSTRACT OF THE DISSERTATION

An Intelligent Tutoring System for Graphic Mapping Strategies

by

Richard Gordon Feifer
Doctor of Philosophy in Education
University of California, Los Angeles, 1989
Professor Eva L. Baker, Co-Chair
Professor Michael G. Dyer, Co-Chair

ABSTRACT

This dissertation presents a cognitive model for the task of graphic mapping. Graphic mapping is a learning strategy that has been shown to improve a learner's comprehension and retention of new material. The cognitive model is implemented in an intelligent tutoring system, Sherlock, intended to train people to use graphic mapping strategies.

A graphic map is a simplified knowledge representation that the learner constructs in order to learn new material. Constructing a graphic map involves making links, which represent relationships, between icons, which represent concepts. Each link the learner makes is evaluated according to Sherlock's knowledge of graphic mapping and of the material being represented. If the link is wrong, then Sherlock attempts to diagnose the source of the learner's misunderstanding.

Three knowledge sources are modelled in order to understand a learner's constructions in a graphic map:

1. The learner's strategy for constructing a graphic map.
2. The learner's current understanding of the content of the text.
3. The learner's interpretation of icons on the screen.

The learner's knowledge of the text is represented in a localist spreading activation network. The strategy for constructing a graphic map is represented in a production system that uses activation patterns in the network to determine what type of graphic map link is appropriate between any two icons.

In a limited evaluation of Sherlock's diagnostic model, Sherlock was able to correctly determine if a learner's links was right or wrong 96% of the time. When the learner was wrong Sherlock was able to diagnose the cause of the misunderstanding 56% of the time.

Chapter 1

Overview

1.0. INTRODUCTION

In this dissertation I examine the problem of teaching people to use a learning strategy called **graphic mapping.** In graphic mapping the learner is taught a simplified knowledge representation scheme. The learner then uses this scheme to pictorially represent textual material. Figure 1.1 contains a sample graphic map construction.

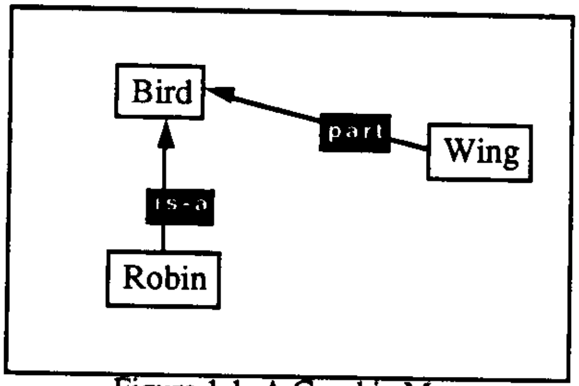


Figure 1.1: A Graphic Map

Researchers have shown that comprehension of text can be enhanced by having the learner construct a graphic map representing the text (Dansereau, 1978; Dansereau, Collins, McDonald, Holley, Garland, Diekhoff & Evans, 1979a,b; Anderson, 1979; Novack, Gowin & Johnson, 1983).

I present a cognitive model for the process of building a graphic map. This model is implemented in Sherlock, an intelligent tutoring system which is intended to teach people how to build graphic maps.

Learning to construct a graphic map requires individual feedback. Automated instruction is the most reasonable means of providing this feedback. However, conventional computer-assisted approaches to providing feedback for the graphic mapping task have been unsuccessful[1]. An intelligent tutoring system can provide automatic individual feedback by inferring the underlying cognitive structures that lead to a graphic map.

Intelligent tutoring systems have been developed which attempt to model a learner's declarative[2] knowledge or a learner's procedural knowledge, but not both. There are two reasons previous systems have not modeled both kinds of knowledge:

1. Previous tutoring systems have been developed in domains which are either predominantly declarative such as geography (Carbonell, 1970) or predominantly procedural such as programming (Anderson, Boyle & Reiser, 1984; Johnson & Soloway, 1983);
2. Tracking the interaction of both types of knowledge is a complicated task.

Sherlock models the acquisition of both procedural and conceptual knowledge in an integrated manner (Feifer, Dyer, & Baker; 1988). There are two motivations for modelling both kinds of knowledge in Sherlock:

1. It is essential for the domain of graphic mapping;
2. It would be beneficial in any domain.

Neither kind of knowledge seems to predominate in graphic mapping. The decisions that a learner makes in constructing a map appear to be equally dependent on both the learner's strategy for building a graphic map (procedural) and the learner's understanding of the text (declarative).

Even domains which seem to rely predominantly on one kind of knowledge involve, to at least some extent, both kinds of knowledge. Programming, for example, requires a knowledge of the syntax and commands of a computer language (declarative) in addition to a knowledge of how to build a program in that language (procedural).

The weakness inherent in modeling only one kind of knowledge can best be seen in an example. Imagine a tutoring system that models a learner's goals and plans. At a given moment this system has determined that the learner should be using the following plan:

---

[1] The problems involved in teaching someone to build a graphic map are discussed in 1.1.4.

[2] When the terms procedural and declarative are used in this dissertation they will not refer to how the knowledge is represented in the computer. Rather they will refer to the distinction made by education researchers between the types of knowledge a learner is trying to acquire. Declarative will refer to the learning of facts and relationships. Procedural will refer to the learning of how to perform a task.

IF X is true

THEN do Y

If the learner does not do Y, then the plan-based tutoring system assumes that the learner does not know this plan. But there are other possible reasons. One possibility is that the learner does not know that X is true.

Thus, a tutoring system able to model the acquisition of both kinds of knowledge can more completely model the acquisition of knowledge in any domain[3].

1.1. GRAPHIC MAPPING

1.1.1. LEARNING STRATEGY RESEARCH

Learning strategy research begins with the question, "Why are successful learners successful?" According to learning strategy research, one of the things that separates successful learners from unsuccessful learners, is the possession and execution of specific strategies to help in the acquisition and application of new knowledge. Successful learners are posited to have these strategies and use them; unsuccessful learners are thought to lack them. One problem is that these strategies are automatic for successful learners, and thus it is difficult to identify and verify them.

The goal of learning strategy research is to define these strategies and then teach them to less successful learners. Graphic mapping is one such strategy; it is based on a strategy developed at Texas Christian University called **Networking** (Holley & Dansereau, 1984a).

1.1.2. THEORETICAL FOUNDATIONS

There has been some previous work in describing the cognitive processes used in graphic mapping (Breuker, 1984; Goetz, 1984). However, these efforts discuss general attributes, and provide insufficient detail to provide a cognitive model that can be implemented in a computer. The work of these researchers does, however, provide a theoretical basis for the use of graphic mapping to enhance learning. Based on this research there are six reasons that the construction of a graphic map may facilitate the learning of textual material:

1. Graphic mapping forces the learner to make relationships explicit. There is a great deal of information in any text which is only implicitly

---

[3] A complete tutoring system must be able to both model knowledge acquisition and provide remediation based on that model. This dissertation will deal mainly with the problem of knowledge acquisition leaving remediation for future work.

stated. In order to represent the text, the learner must make this information explicit.

2. Graphic mapping forces the learner to re-organize the concepts, facilitating integration into his existing knowledge structures. According to the generative model of learning (Wittrock, Marks & Doctorow, 1975), learning takes place only when the learner actively processes information to fit his existing knowledge.
3. Graphic mapping adds a pictorial learning component to textual learning. Research has shown that learning is enhanced when material is presented in both a visual and a verbal component (Paivio, 1971).
4. Graphic mapping provides external memory to reduce cognitive overhead. Thus graphic mapping can simplify text understanding as a pencil and paper can simplify arithmetic (Breuker, 1984).
5. Graphic mapping provides a mnemonic aid. The graphic map provides an image which can later be used to facilitate retrieval of information (Bower, 1970).

1.1.3. EMPIRICAL EVIDENCE

Most studies designed to measure the effectiveness of a learning strategy take the form illustrated in Figure 1.2.

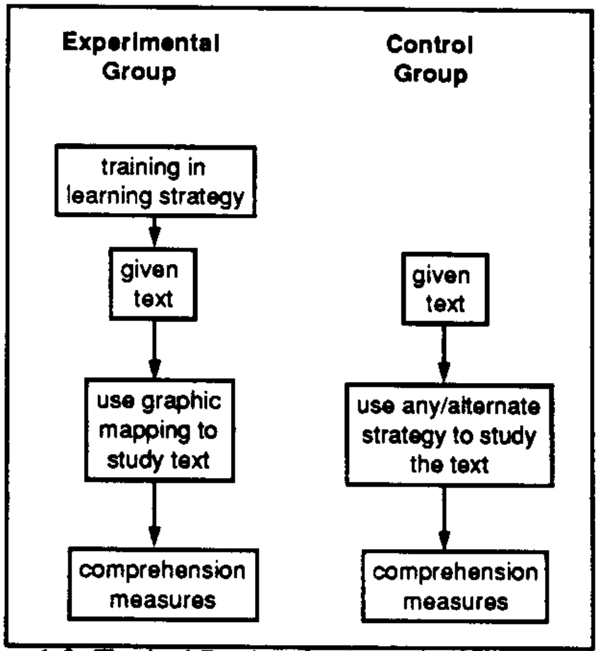

Experimental Group
Control Group
training in learning strategy
given text
given text
use graphic mapping to study text
use any/alternate strategy to study the text
comprehension measures
comprehension measures

Figure 1.2: Typical Design for Learning Strategy Study

The experimental group is trained to use the particular learning strategy. The experimental and control groups are then given a text to read and study. The experimental group is told to use the learning strategy which they have just studied. Depending on the study, the control group is either told to use their own study technique, or some alternate technique which they have been taught. Both groups are given equal time for reading and studying.

Tests are then used to measure the comprehension and retention of the text material in each of the groups.

A number of studies have demonstrated that graphic mapping learning strategies enhance a learner's ability to remember and understand new material (Dansereau, 1978; Dansereau, Collins, McDonald, Holley, Garland, Diekhoff & Evans, 1979a; Anderson, 1979; Novak, Gowin & Johnson, 1983).

Some studies, however, have shown no significant improvement for groups using graphic mapping learning strategies (Long, 1976; Camstra, 1979). The lack of improvement has usually been attributed to either the representation strategy used or the quality of training. Long (1976) found that having too few (3) link types or too many (13) link types resulted in no improvement. Long also attributed poor results to training problems.

Camstra (1979) compared two different training approaches for the same graphic mapping type learning strategy. One group (12 subjects) learned the strategy in a supervised training program. The second group (11 subjects) taught themselves using specially prepared training materials. Eleven of the 12 subjects in the supervised group successfully learned to apply the strategy. Only 3 of the 11 subjects in the unsupervised group were able to use the strategy. After several studies with similar results, Camstra concluded that sufficient practice with feedback is essential to effectively train someone to use a graphic mapping type learning strategy (Camstra & Bruggen, 1984).

1.1.4. PROBLEMS

Training is an important issue in all learning strategy research. Even if a strategy is potentially effective, it is useless if a learner cannot be trained to use it. Graphic mapping is particularly difficult because it involves teaching the learner to represent knowledge. Learning to represent knowledge has proven difficult for graduate students in artificial intelligence. It is not surprising, then, that teaching unsuccessful learners to produce even simplified knowledge representations is difficult.

Practice with feedback is crucial to effective training in graphic mapping strategies. But it is impractical to provide supervised training with individual feedback in most educational settings. Researchers have attempted two approaches to providing feedback:

1. demonstrating correct strategy usage to groups (Holley & Dansereau, 1984b);
2. evaluating a learner's graphic map with an "automatic scoring procedure" (Bruggen, 1981; Camstra, 1979).

Demonstrating the correct strategy can be accomplished by providing the learner with a finished "expert" map to compare with his own. The problem with this approach is that it is difficult for the novice to decide whether differences between his map and the "expert" map are due to individual interpretations or errors.

An automatic scoring procedure for graphic maps would not only aid training, but would help establish whether subjects have mastered a learning strategy when testing the effectiveness of that learning strategy. No previous attempt at an automatic scoring procedure has been successful (Camstra & Bruggen, 1984). This is due to two factors:

1. Each graphic map is highly individual;
2. The notation used in most graphic map learning strategies is ambiguous.

An intelligent tutoring system designed to train people to construct graphic maps must incorporate some form of automatic scoring procedure. Thus Sherlock must account for these two factors. Due to the individual nature of graphic maps, constructions cannot be evaluated by comparison to some list of correct constructions. Sherlock uses a knowledge-based approach to the problem. Constructions are evaluated by determining whether a model of the processes involved in building a graphic map can account for the construction.

1.2. SHERLOCK ENVIRONMENT

The Sherlock tutoring environment provides the learner with three components: (a) a text to be represented pictorially, (b) a screen containing icons[4] representing concepts within the text, and (c) a set of gm-links[5] which the learner can use to connect the icons.

1.2.1. THE COMPONENTS

Sherlock has been programmed to facilitate learning of the text in Figure 1.3. This text is excerpted from an introductory business law text book and describes consideration and its place in a contract.

> A contract is usually an agreement in which, in effect, one party says to another, "If you do this for me, I shall do that for you." The thing which one person asks another to do in return for a promise is the consideration. Consideration may be the giving of another promise, or it may be an act. Consideration may also consist of forbearance, that is, refraining from doing what one has a right to do.
>
> Consideration must: (a) be lawful, and (b) require conduct that is not already required by law, or must prohibit conduct which is not already prohibited by law. If the consideration is a promise, the promise must also (c) be definite. (Fisk & Mietus, 1977)[6]

Figure 1.3 : Consideration Text

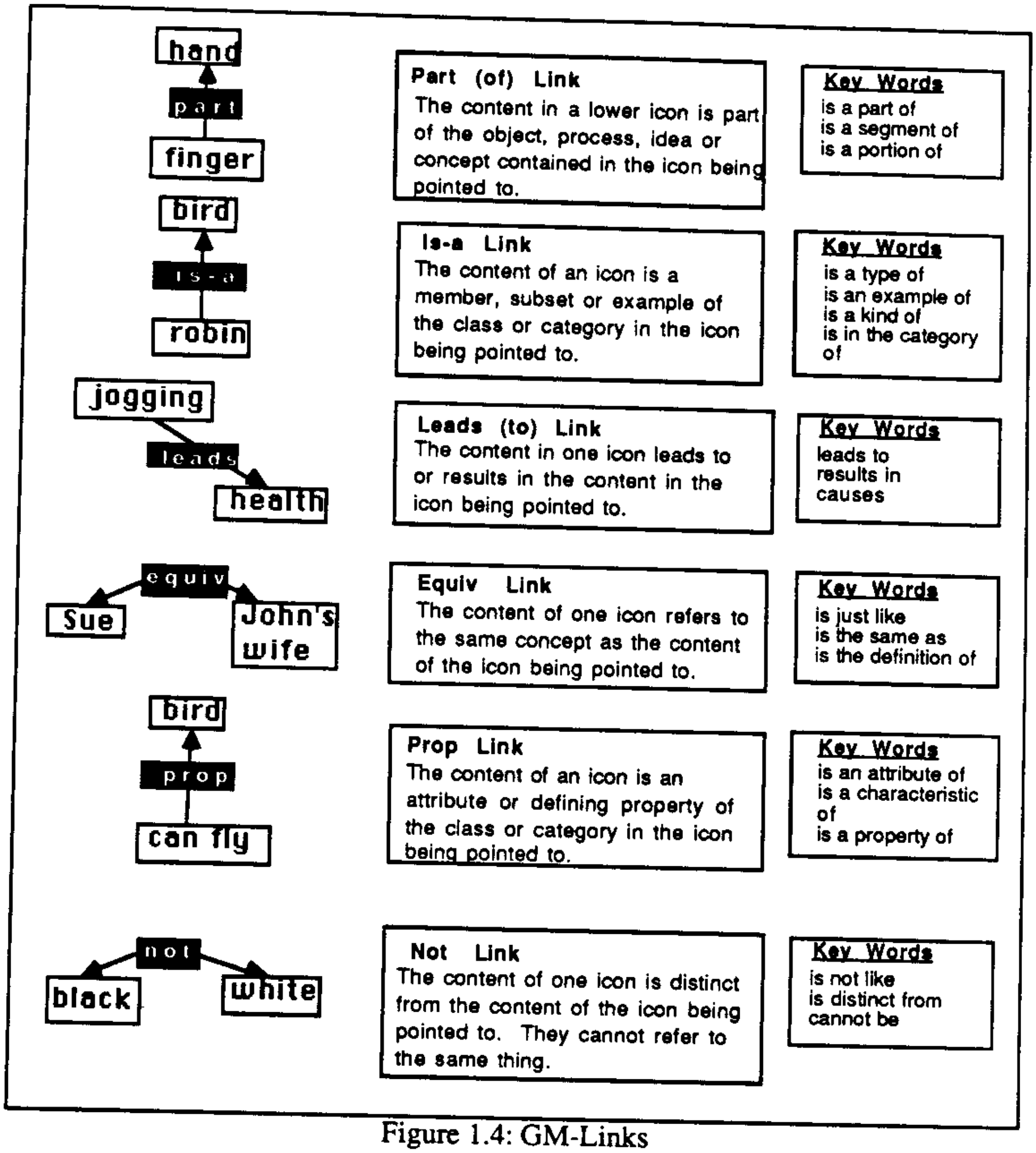
---

[4] I use the term icon to refer to a box containing text that the learner can use to represent a conceptual entity.

[5] To avoid confusion between the graphic map that the learner constructs and Sherlock's internal knowledge representation, links used in a graphic map will be referred to as gm-links.

[6] This is an edited version of text from an introductory business law textbook.

Currently Sherlock offers the learner six gm-links: PART, IS-A, LEADS, EQUIV, PROP and NOT (Figure 1.4). Learners are instructed to find icons that they believe are related and then choose the gm-link that best represents the relationship. For example, if the learner believes that one icon represents a concept that is a generalization of a concept represented by a second icon, he should make an IS-A gm-link between the two icons.



Part (of) Link The content in a lower icon is part of the object, process, idea or concept contained in the icon being pointed to.	**Key Words** is a part of is a segment of is a portion of
Is-a Link The content of an icon is a member, subset or example of the class or category in the icon being pointed to.	**Key Words** is a type of is an example of is a kind of is in the category of
Leads (to) Link The content in one icon leads to or results in the content in the icon being pointed to.	**Key Words** leads to results in causes
Equiv Link The content of one icon refers to the same concept as the content of the icon being pointed to.	**Key Words** is just like is the same as is the definition of
Prop Link The content of an icon is an attribute or defining property of the class or category in the icon being pointed to.	**Key Words** is an attribute of is a characteristic of is a property of
Not Link The content of one icon is distinct from the content of the icon being pointed to. They cannot refer to the same thing.	**Key Words** is not like is distinct from cannot be

Figure 1.4: GM-Links

A portion of the screen icons are pictured in Figure 1.5.

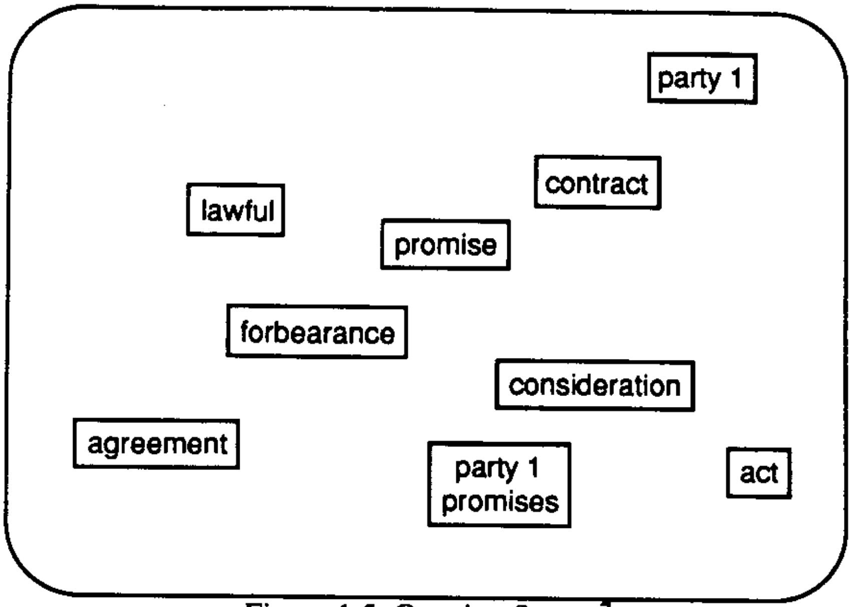


Figure 1.5: Opening Screen[7]

1.2.2. MAKING GM-LINKS

Sherlock provides the learner with the capacity to move icons, copy icons, link them together, and remove links. To link two icons together, the learner takes the following steps:

1. The learner uses the mouse to move the two icons to the same area of the screen (Figure 1.6). Moving the icons together is not mandatory, but helps keep the graphic map from getting tangled.

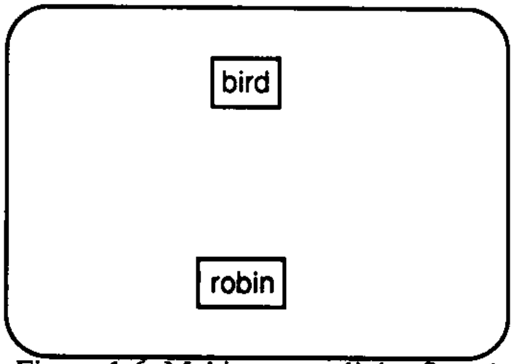


Figure 1.6: Making a gm-link - Step 1

---

[7] Whenever a figure illustrates elements of a graphic map, its frame will have rounded corners to indicate that it is something which the learner would see on the computer screen.

2. The learner uses the mouse to indicate, by clicking on it, the icon he wants the gm-link to come from (Figure 1.7).

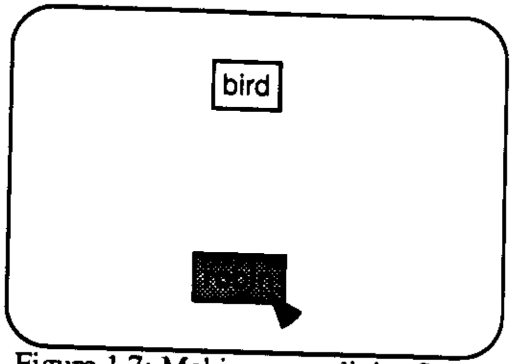


Figure 1.7: Making a gm-link - Step 2

3. The learner then clicks on the icon he wants the gm-link to go to (Figure 1.8).

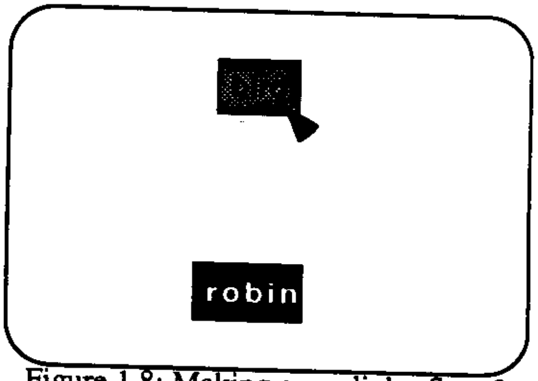


Figure 1.8: Making a gm-link - Step 3

4. Sherlock presents a prompt asking the learner to choose a gm-link type. The learner chooses a gm-link type by clicking on the desired type in a menu (Figure 1.9).

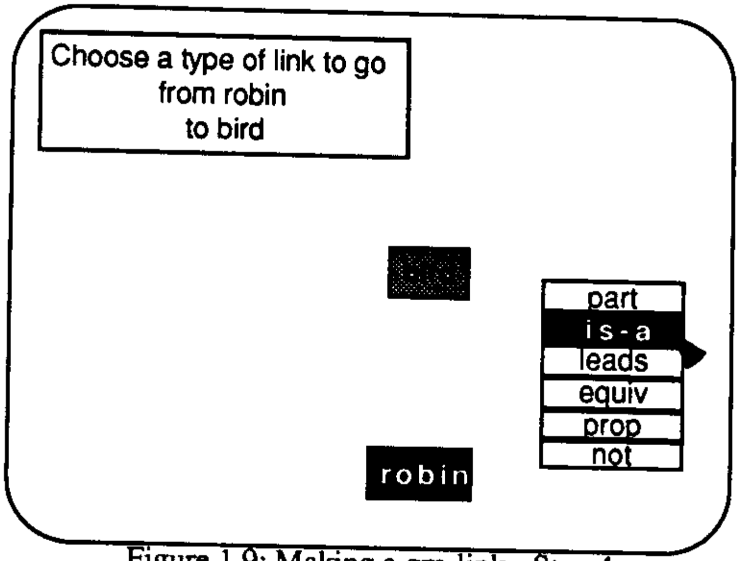

Choose a type of link to go
from robin
to bird

Figure 1.9: Making a gm-link - Step 4

5. Sherlock constructs the desired gm-link between the two icons (Figure 1.10).

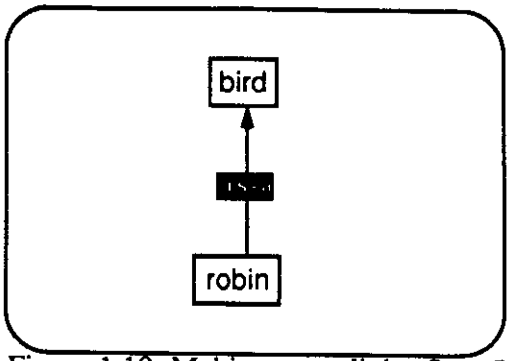


Figure 1.10: Making a gm-link - Step 5

1.3. SHERLOCK'S INPUT AND OUTPUT

Once the learner has made a gm-link, Sherlock tries to evaluate the gm-link and then provide meaningful feedback. What follows are excerpts of transcripts from actual interactions between learners and Sherlock (the complete transcripts are included in Appendix C). The text in parentheses are descriptions of the interactions and the figures that illustrate the content of the screen at each point in the interaction. All other text are comments made by the learner.

1.3.1. JANEY

(In the first excerpt a subject named Janey makes two attempts to link the icons "consideration act" and "legal act.")

1.3.1.1. FIRST ATTEMPT

I was first looking at these icons here (Figure 1.11); instead I am going to look at the links (the list of link types) to try and link something together from these different links I have to choose from.

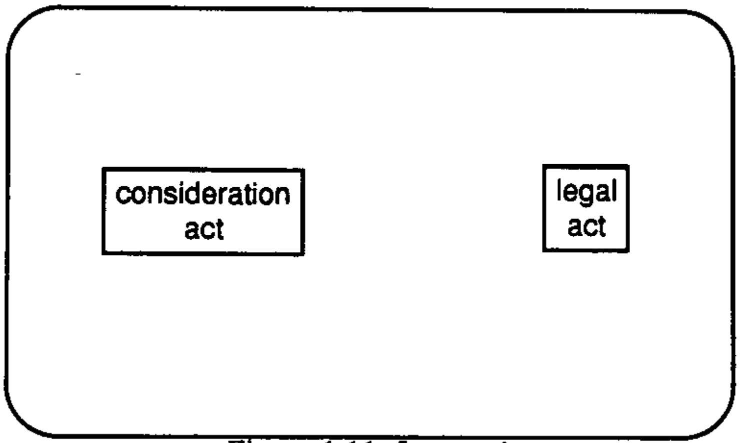


Figure 1.11: Janey - 1

Now they are equivalent according to this. So I am going to use an equivalent link (Figure 1.12).

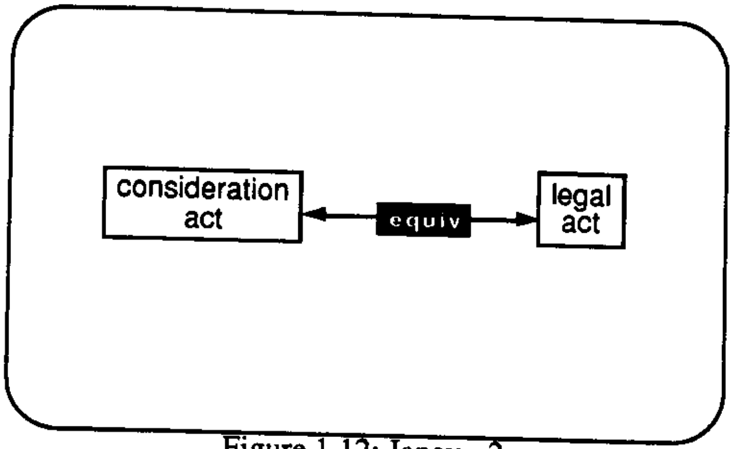
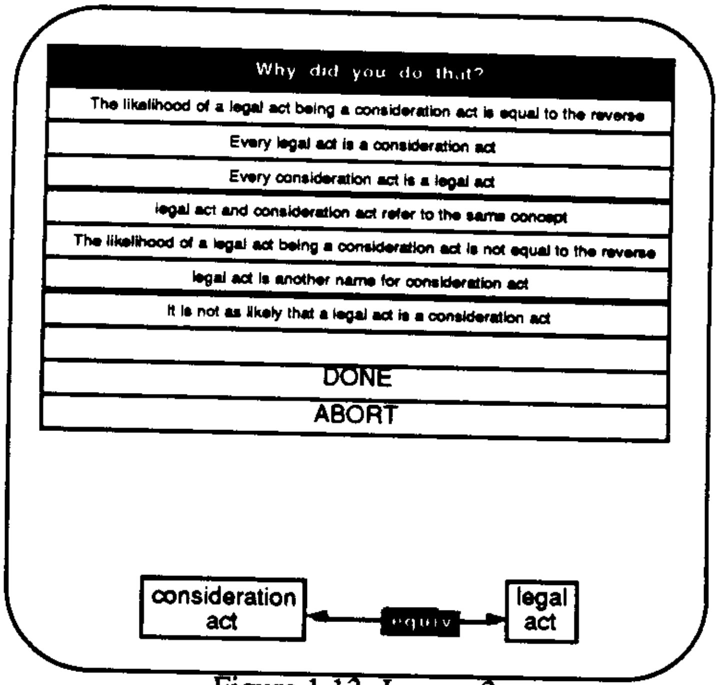


Figure 1.12: Janey - 2

According to this paragraph a consideration act is a lawful act. So a lawful act is a legal act. So I am going to link them as equivalents because they are both legal acts.

(Sherlock requests information regarding her reasons for making the link by presenting her with the menu shown in Figure 1.13)


Why did you do that?

The likelihood of a legal act being a consideration act is equal to the reverse

Every legal act is a consideration act

Every consideration act is a legal act

legal act and consideration act refer to the same concept

The likelihood of a legal act being a consideration act is not equal to the reverse

legal act is another name for consideration act

It is not as likely that a legal act is a consideration act

Figure 1.13: Janey - 3

In general I wouldn't say that a "consideration act" is a "legal act". In general. But this (the text) is telling me that it is. So I am answering these questions in regard to the material here? Yes.

(She chooses "Every consideration act is a legal act" and "It is not as likely that a legal act is a consideration act," Figure 1.14)

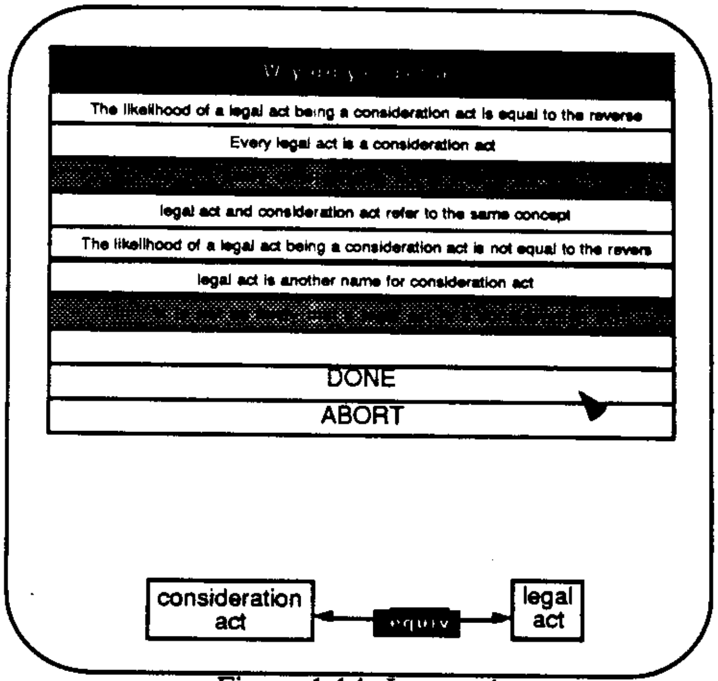

The likelihood of a legal act being a consideration act is equal to the reverse
Every legal act is a consideration act
legal act and consideration act refer to the same concept
The likelihood of a legal act being a consideration act is not equal to the revers
legal act is another name for consideration act

Figure 1.14: Janey - 4

(Sherlock provides her with the plan feedback shown in Figure 1.15)

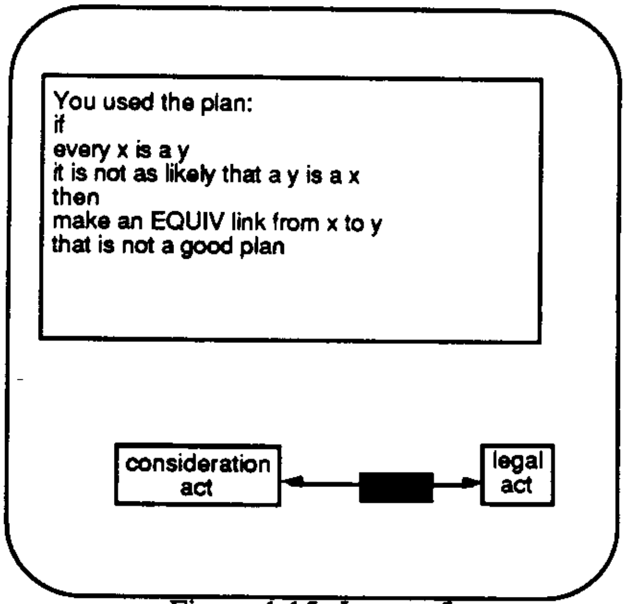

You used the plan:
if
every x is a y
it is not as likely that a y is a x
then
make an EQUIV link from x to y
that is not a good plan

Figure 1.15: Janey - 5

Sounds like algebra, and I am just terrible at algebra.

I guess that is what ... I used, but it doesn't really make good sense. That is the plan I used, but it doesn't seem like it is a very good plan or that it is going to be equivalent.

(Sherlock provides the comment shown in Figure 1.16)

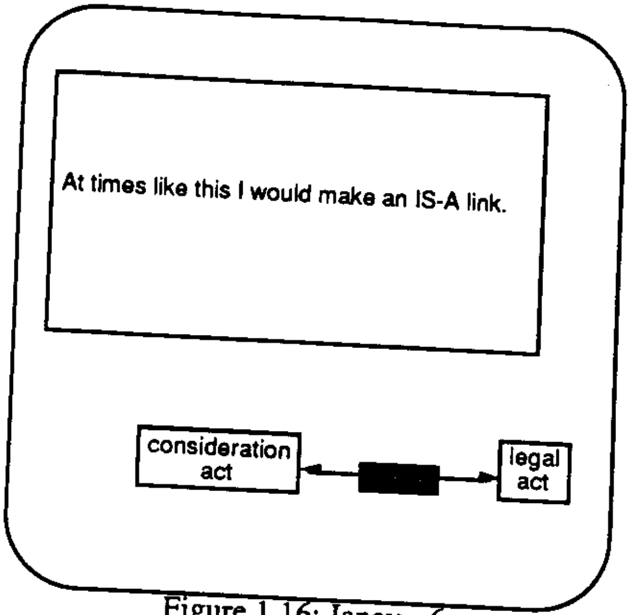

At times like this I would make an IS-A link.

Figure 1.16: Janey - 6

It means where I made the equivalent link? Huh. Let's check that out.

(In Figure 1.17 Sherlock asks her if she would like to remove the link. She answers "YES". In Figure 1.18 the link is removed.)

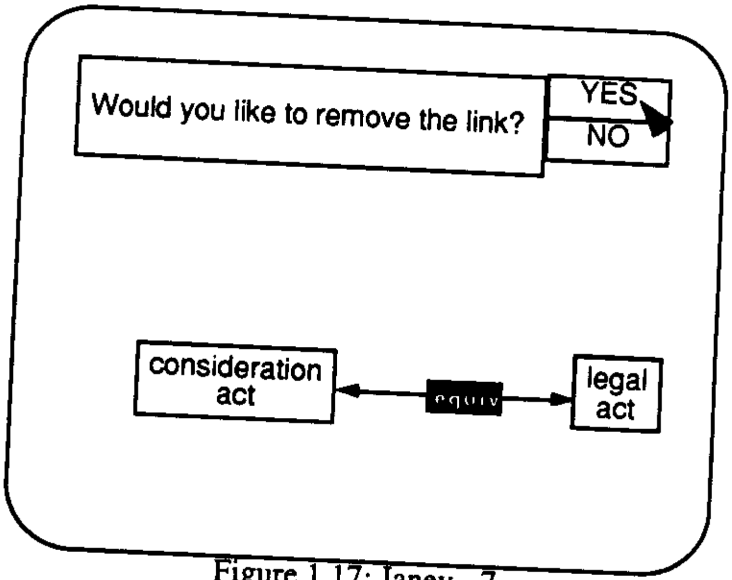

Would you like to remove the link?

Figure 1.17: Janey - 7

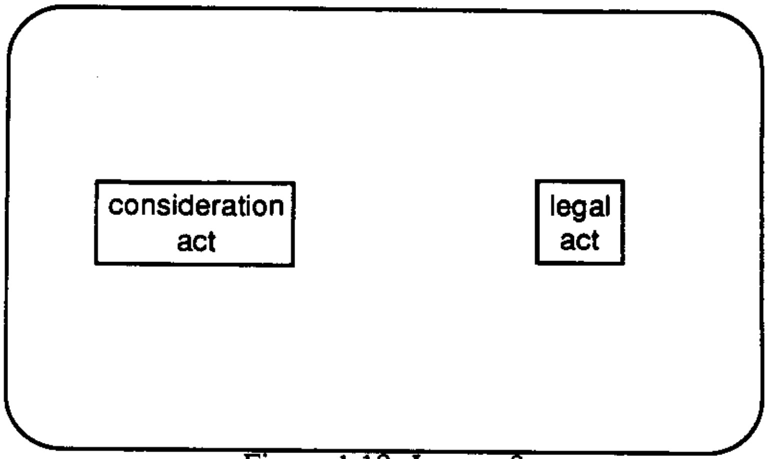


Figure 1.18: Janey - 8

1.3.1.2. SECOND ATTEMPT

So, "legal act" well... "consideration act" would be... would be a subset of a "legal act", or vice versa. Or do I have to decide that? I guess I have to decide that. Huh, OK. I would say that a... Well, according to this a "consideration act" is a type of a "legal act". So I am going to connect "consideration act" (Figure 1.19).

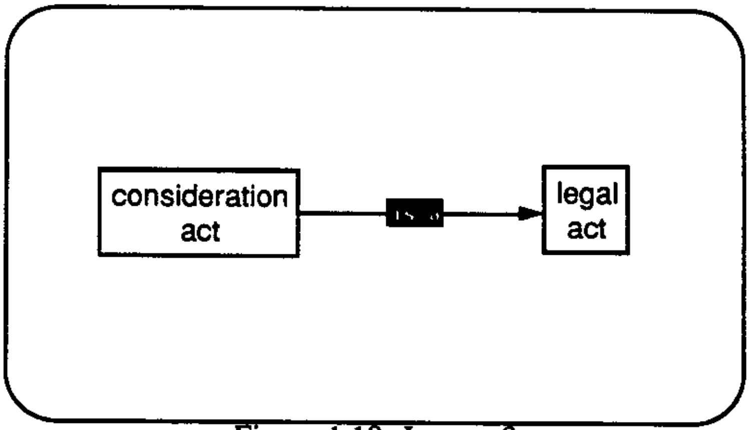


Figure 1.19: Janey - 9

Well according to this a "consideration act" is a "legal act". It is one type, it is a type of a "legal act". So it's got to be legal. So... it's some type, or kind of "legal act".

(Sherlock provides the feedback shown in Figure 1.20)

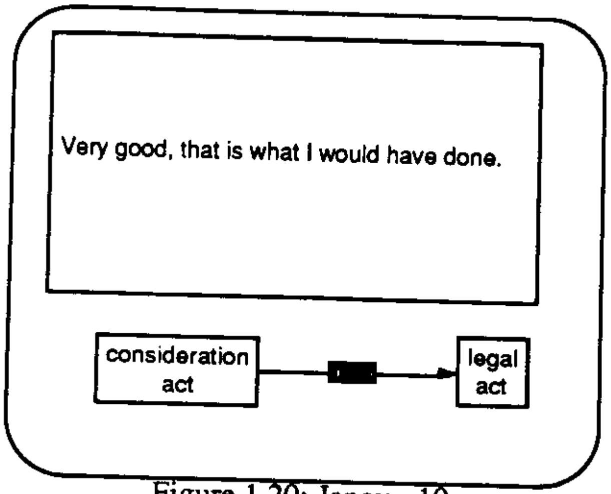

Very good, that is what I would have done.

Figure 1.20: Janey - 10

1.3.2. JENNIFER

(In the second excerpt a subject named Jennifer attempts to represent the relationship between "contract" and "party1 promises to do something for party2.")

1.3.2.1. FIRST ATTEMPT

This is an example of a contract (Figure 1.21).

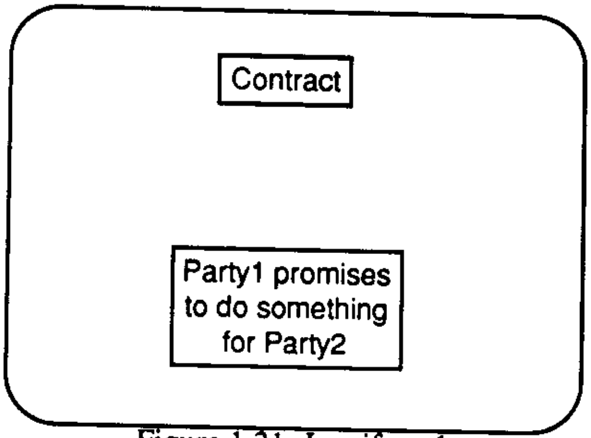


Figure 1.21: Jennifer - 1

(She makes an IS-A gm-link between them, Figure 1.22)

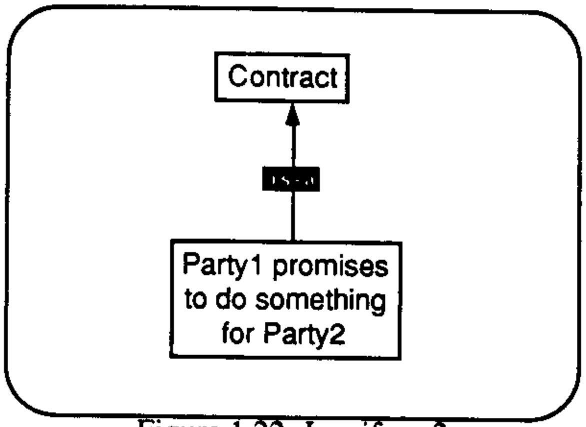
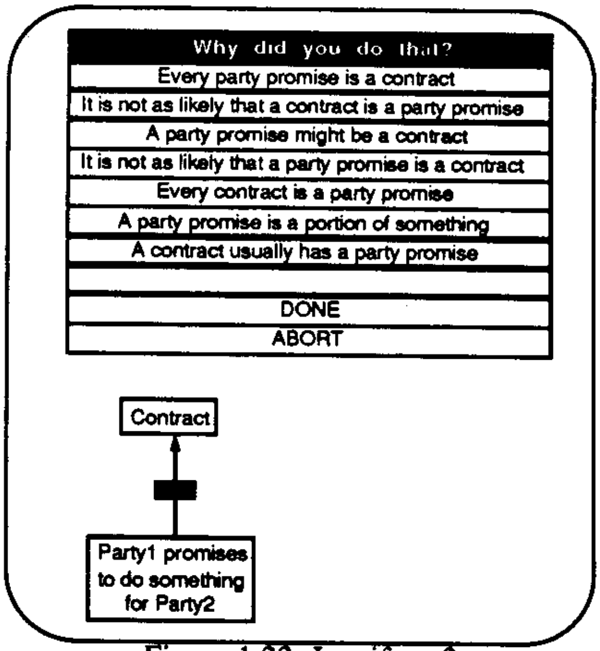


Figure 1.22: Jennifer - 2

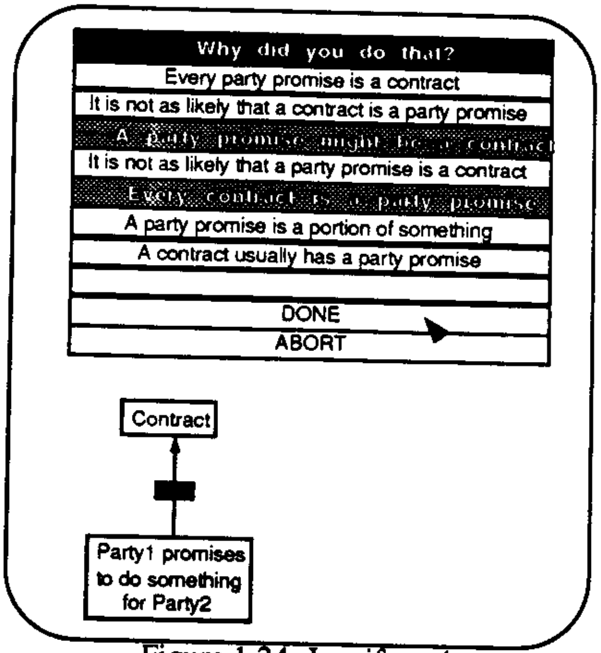
(Sherlock asks her to indicate her reasons for making the gm-link, Figure 1.23)



Every party promise is a contract
It is not as likely that a contract is a party promise
A party promise might be a contract
It is not as likely that a party promise is a contract
Every contract is a party promise
A party promise is a portion of something
A contract usually has a party promise

Figure 1.23: Jennifer - 3

(She chooses two of the possible reasons, Figure 1.24)



Why did you do that?

Every party promise is a contract

It is not as likely that a contract is a party promise

A party promise might be a contract

It is not as likely that a party promise is a contract

Every contract is a party promise

A party promise is a portion of something

A contract usually has a party promise

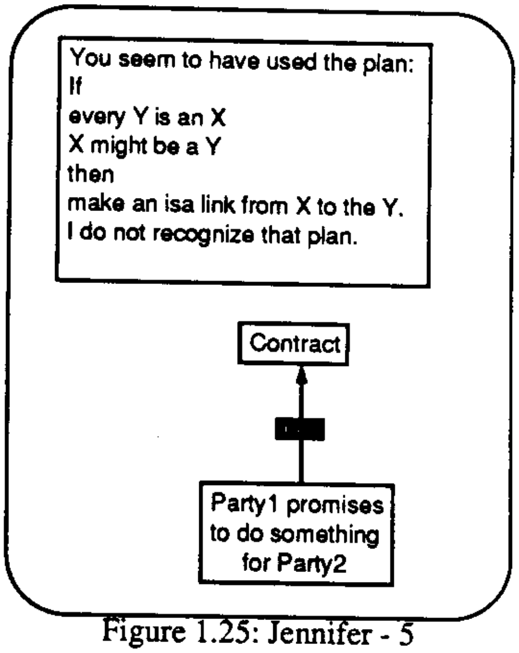
Figure 1.24: Jennifer - 4

It's probably going to tell me that it thinks it (the link) is wrong. It could be an equivalent too because of what the paper says. It could also be equivalent.

(Sherlock gives her feedback on the plan, Figure 1.25)

Yeah. It is the plan I used. Yeah, it should recognize this plan. No, I guess I don't think it's a good plan.

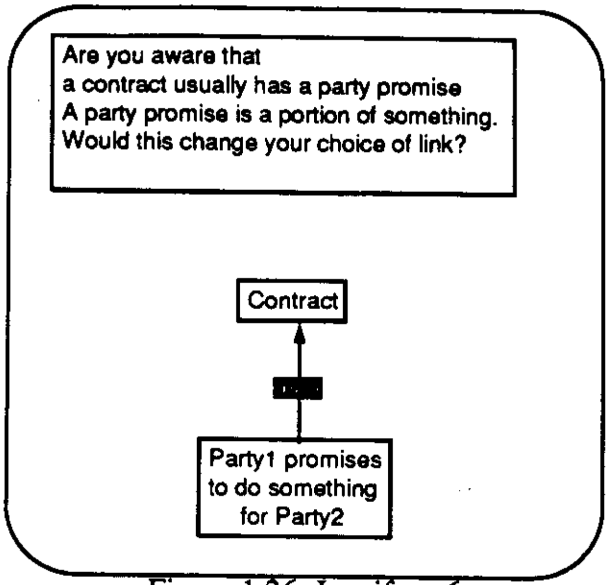
Because IS-A kinda means that A is B and B is A and if (they might be), then it's not always.



You seem to have used the plan:
If
every Y is an X
X might be a Y
then
make an isa link from X to the Y.
I do not recognize that plan.

Figure 1.25: Jennifer - 5

(Sherlock gives her feedback on the facts, Figure 1.26)

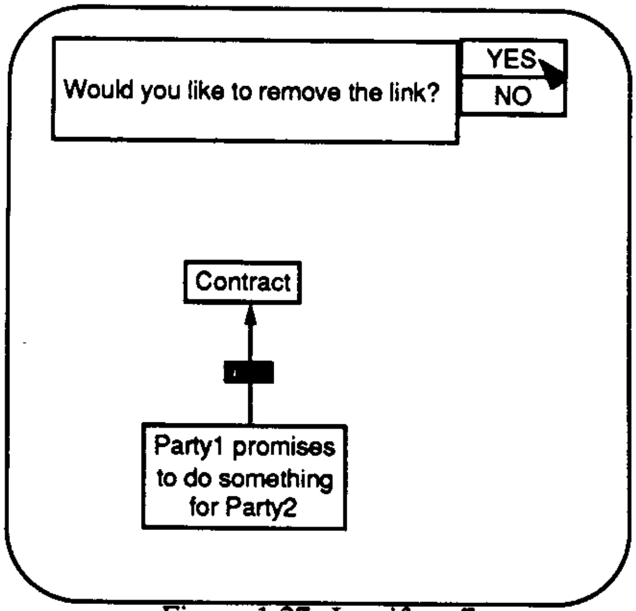


Are you aware that
a contract usually has a party promise
A party promise is a portion of something.
Would this change your choice of link?

Figure 1.26: Jennifer - 6

Yes I was aware of that. Oh, it's probably a portion of the contract. I didn't think like that.

(Sherlock asks if she would like to remove the gm-link, Figure 1.27)



Would you like to remove the link?

Figure 1.27: Jennifer - 7

Yes.

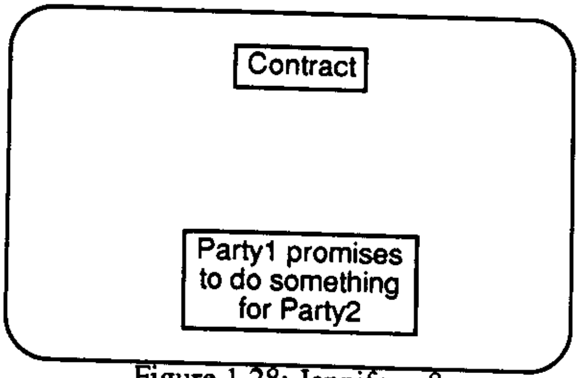


Figure 1.28: Jennifer - 8

1.3.2.2. SECOND ATTEMPT

I might try the "leadsto" (link). Party1 promising to do something for party2 leads to the contract (Figure 1.29).

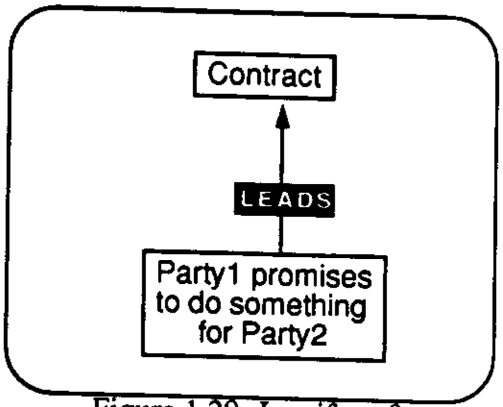
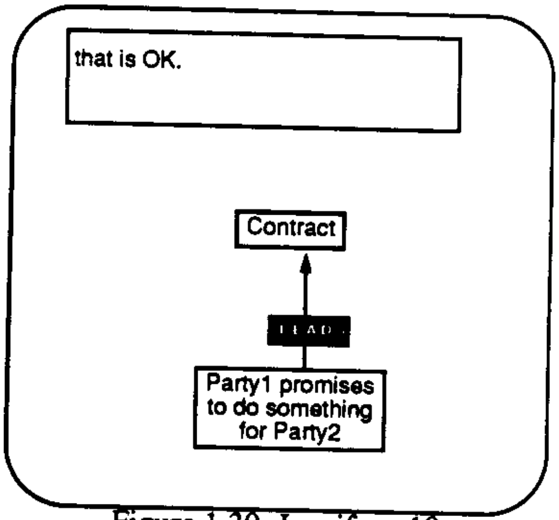


Figure 1.29: Jennifer - 9

(Sherlock provides feedback on the LEADS gm-link, Figure 1.30)

So I'm kind of right.



that is OK.

Figure 1.30: Jennifer - 10

1.4. SHERLOCK'S TUTORING STRATEGY

To provide the type of feedback shown in the preceding excerpts Sherlock follows the strategies illustrated by the flow chart in Figure 1.31.

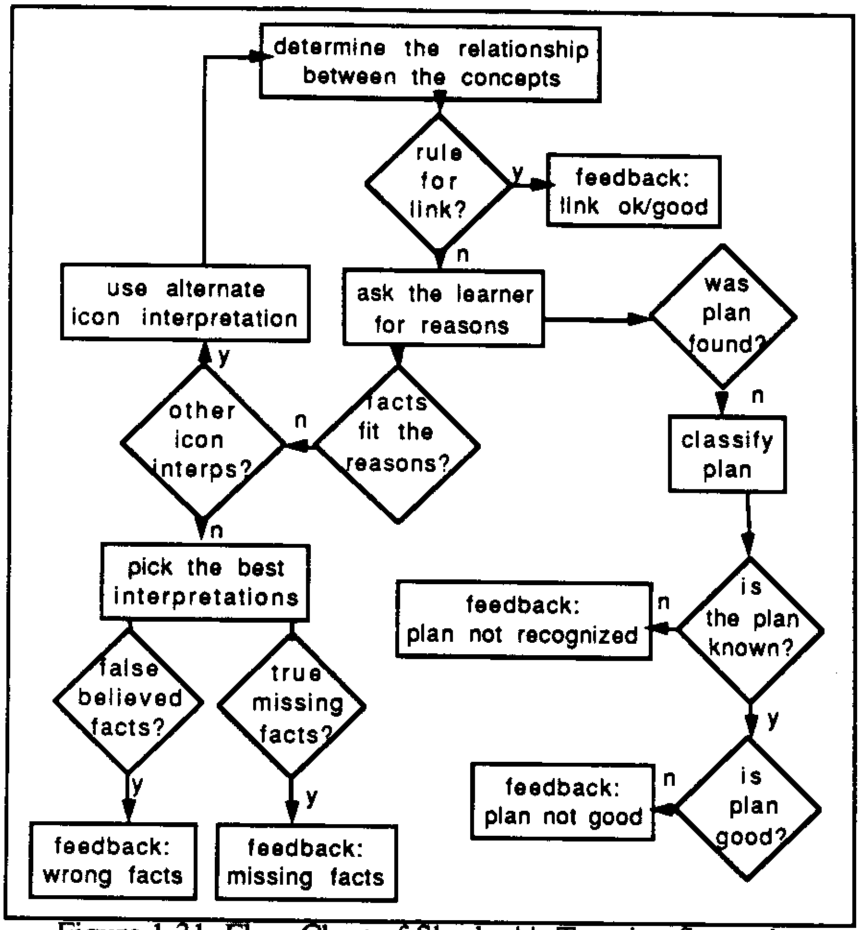


Figure 1.31: Flow Chart of Sherlock's Tutoring Strategies

Beginning at the top of Figure 1.31, Sherlock determines the relationship between the two icons that the learner just linked. Sherlock then determines whether it knows any plans that say in effect "If two icons have this relationship, then make the gm-link which the learner just made." If such a plan exists ("rule for link?"), Sherlock will tell the learner that the plan is either good or OK, depending on how strong Sherlock believes that plan to be. If a plan which would justify the gm-link (the learner just made) does not exist, Sherlock asks the learner to indicate the reasons the gm-link was made.

The flow chart splits at this point, using the learner's answer to separately evaluate the learner's plan and the facts that the learner believes.

If Sherlock has not already determined what plan the learner used, it will do so at this point. This determination is made by comparing all of the known plans with the learner's reasons for making the gm-link, and the gm-link the learner made. If

Sherlock cannot recognize the plan the learner used, it will inform the learner of this. If Sherlock does recognize the plan, but it is a bad plan, it will inform the learner that he has used a bad plan. If Sherlock recognizes the plan and it is a good plan, it will not give any feedback on the plan.

To evaluate the learner's fact beliefs, Sherlock compares the learner's answer to its own fact beliefs. If there is a good[8] match between the learner's fact beliefs and Sherlock's, no further action is taken on facts. If there is not a good match, Sherlock will consider the possibility that the learner is using an alternative interpretation for the icons. If any alternative interpretations for the icons exist, that have not already been tried, Sherlock will use one of these interpretations and start again from the top.

If all possible alternative interpretations have been tried, Sherlock will pick the interpretation that led to fact beliefs which were closest to the learner's fact beliefs. Using this interpretation Sherlock will determine two things:

1. Are there any false facts that the learner believes to be true, that led to the bad gm-link that the learner made? If so, Sherlock will bring them to the learner's attention.
2. Are there any true facts that the learner does not believe, which would have led to a better gm-link? If so, Sherlock will bring them to the learner's attention.

1.5. SHERLOCK'S ARCHITECTURE

To carry out the strategy described above Sherlock uses the eight components shown in Figure 1.32.

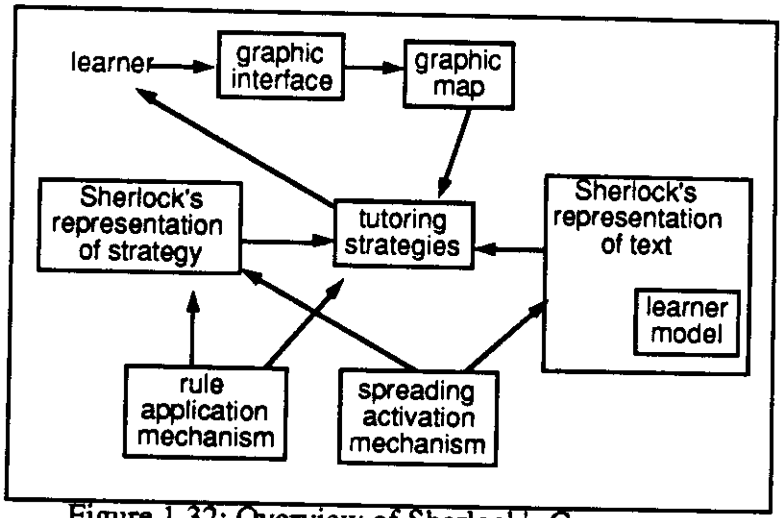

learner
graphic interface
graphic map
Sherlock's representation of strategy
tutoring strategies
Sherlock's representation of text
learner model
rule application mechanism
spreading activation mechanism

Figure 1.32: Overview of Sherlock's Components

---

[8] "Good" is currently defined as meaning that the learner and Sherlock agree on 80% of the facts.

The function of each of these components is briefly described below.

1. **The graphic interface** interprets the learner's clicks on the mouse buttons to build the graphic map. The interface also translates the graphic map that the learner is producing on the screen into an internal representation. The graphic interface was built on top of a window package developed by Seth Goldman in the UCLA AI lab.
2. The learner builds the **graphic map** using the icons provided by Sherlock. The icons are linked together using the six gm-links described above.
3. **Sherlock's representation of text** is a semantic network made up of nodes for concepts and links for the relationship between concepts. The semantic network is used to represent Sherlock's understanding of a text's content and background. The structure of the network is based largely on Fahlman's NETL (1979). The representation of the text is hand-coded into the network.
4. **Sherlock's representation of strategy** consists of rules or plans for building a graphic map. Each of the rules is of the form:

   IF the following things are true about the relationship between 2 icons
   THEN make this kind of gm-link between them

   These rules are hand-coded into Sherlock and are based on strategies described by subjects during initial pilot studies.

   The rules as a whole represent two things:

   1. the strategy Sherlock would use to build a graphic map;
   2. the strategy Sherlock initially assumes the learner will use to build a graphic map.
5. The **learner model** represents what Sherlock currently believes to be the learner's understanding of the text, graphic elements, and mapping skills. It is built by Sherlock by modifying elements in the hand-coded representations of the text content and graphic mapping strategy.

   The learner model is not fully implemented at this time. The only aspect of the learner model which Sherlock currently represents is the learner's interpretation of the screen icons.
6. **The tutoring strategies** are rules or plans in a form similar to that of the strategies for building a graphic map. The rules in the tutoring strategies are hand-coded to represent Sherlock's pedagogical knowledge. They implement the steps illustrated by the flow chart in Figure 1.31.
7. The **spreading activation mechanism** operates on the semantic network to generate inferences. It is based on a mechanism developed by Michael Gasser (1988) for classifying concepts. When a node is activated, the mechanism allows the activation to spread to surrounding nodes through that node's links. The activation next spreads from the surrounding nodes through their links, and so on.

Sherlock uses spreading activation for two purposes:

1. to classify a learner's plan as being a specialization of one of the plans in Sherlock's representation of strategy;

2. to determine the relationship between any two concepts.

8. The **rule application mechanism** operates on the rules in Sherlock's representation of strategy and the tutoring strategies to determine what action Sherlock should take, given those rules.

The mechanism finds all of the rules that match the current state of the system and have not already been used for the current goal. It then selects from this group the strongest rule, and carries out its actions.

1.6. MOTIVATION FOR AN INTELLIGENT TUTORING SYSTEM

Sherlock is intended for at least three potential uses:

1. Automatic training in graphic mapping learning strategy

Empirical research has shown that proper training in graphic mapping can help less successful learners become more successful. Sherlock provides a non-labor intensive solution to the training problem.

2. To facilitate mastery of material represented in tutoring system

As a by-product of teaching graphic mapping, Sherlock facilitates mastery of the material represented for the training session. At present it is a non-trivial task to adapt Sherlock to work with new textual material. It is possible, however, that in the near future Sherlock will be able to automatically parse a new text (Dyer, 1983). Sherlock could then adapt itself to any new text material that the learner needs to master.

3. Assessment of an individual's cognitive structures

Graphic mapping type techniques have been used to both measure and describe a learner's knowledge (West, 1985). These efforts, however, have been hampered by the problems just discussed: the difficulties involved in training someone to build a graphic map, and the difficulties involved in assessing a graphic map. Sherlock not only provides the training, but also can aid in the interpretation of a graphic map.

1.7. ORGANIZATION OF THE THESIS

Chapter 2 describes the two pilot studies which analyzed graphic mapping and provided the basis for the model implemented in Sherlock.

Chapter 3 describes the semantic network and its use in representing Sherlock's knowledge of the text.

Chapter 4 describes the representation of rules in Sherlock, and their use in representing the strategy for building a graphic map and the strategy for tutoring.

Chapter 5 describes the spreading activation mechanism and its use in classification and determining the relationship between two concepts.

Chapter 6 describes the processes that operate on rules in Sherlock.

Chapter 7 describes the implementation of Sherlock and its performance. Included are traces from the session which led to the I/O excerpts in this chapter.

Chapter 8 discusses an empirical evaluation of Sherlock's performance in tutoring four subjects.

Chapter 9 discusses previous work in intelligent tutoring systems and related work from educational research on modelling a learner's cognitive structures.

Chapter 10 summarizes the findings and implications of this dissertation and suggests future directions.

The appendices contain instruments and data from the pilot studies and complete protocols from the final study.

Chapter 2

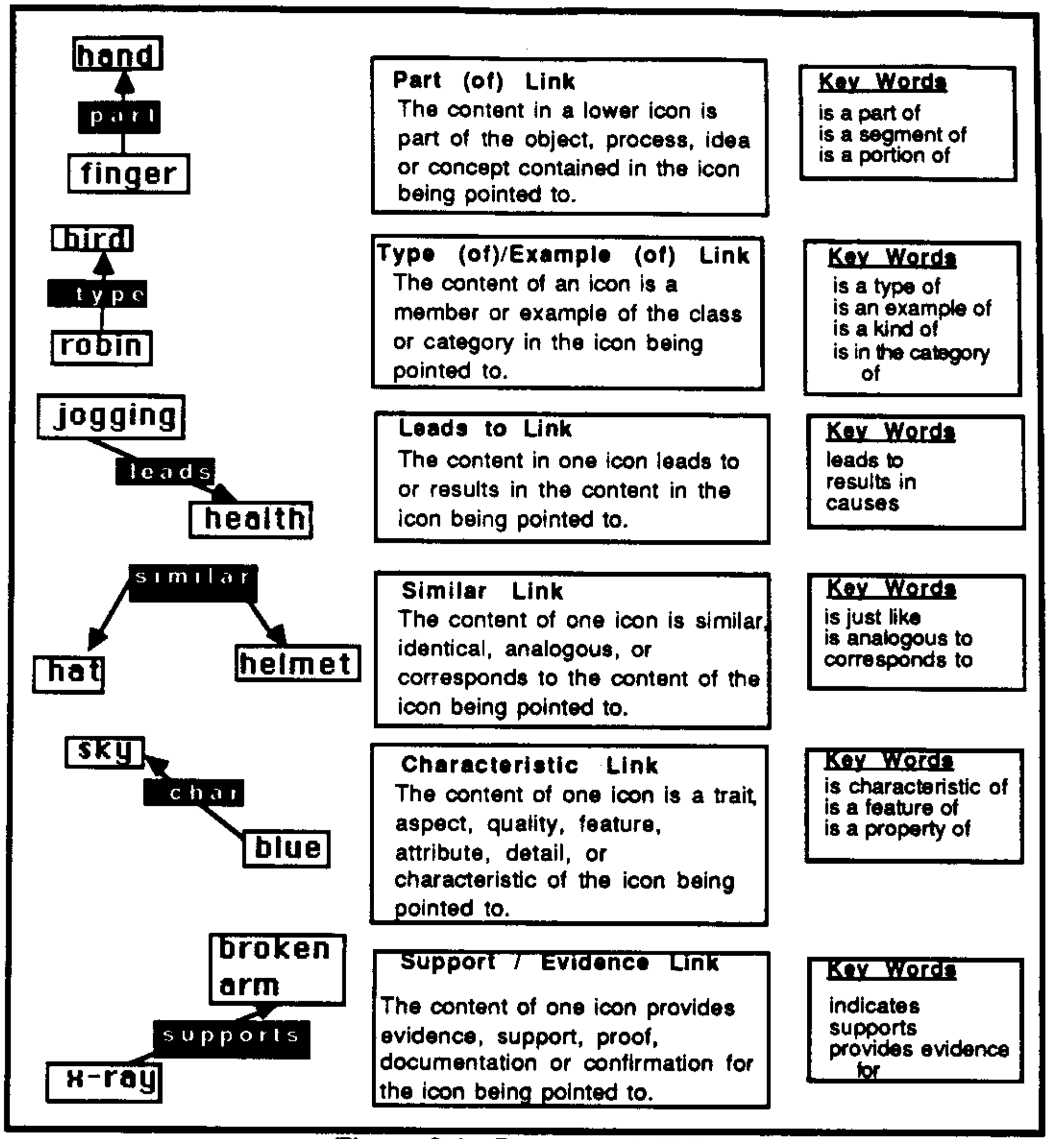
Analysis of the Graphic Mapping Task

2.0. INTRODUCTION

Two pilot studies were conducted to explore the process of building a graphic map. The first study was designed to look at the process from the point of view of a tutor watching a graphic map being constructed. The second study was designed to look at the process from the point of view of the learner constructing a graphic map.

2.1. FIRST PILOT STUDY

The first pilot study was conducted with 20 UCLA undergraduates in December of 1985. Subjects were asked to construct graphic maps for the consideration text (Figure 1.3) using a slightly modified version of the six links specified by Dansereau (1978). Figure 2.1 contains the description of these links as given to subjects. An attempt was made to infer what the subject was thinking based on his actions. These inferences were then validated using a posttest on relationships in the text.



Part (of) Link
The content in a lower icon is part of the object, process, idea or concept contained in the icon being pointed to.

Key Words
is a part of
is a segment of
is a portion of

Type (of)/Example (of) Link
The content of an icon is a member or example of the class or category in the icon being pointed to.

Key Words
is a type of
is an example of
is a kind of
is in the category of

Leads to Link
The content in one icon leads to or results in the content in the icon being pointed to.

Key Words
leads to
results in
causes

Similar Link
The content of one icon is similar, identical, analogous, or corresponds to the content of the icon being pointed to.

Key Words
is just like
is analogous to
corresponds to

Characteristic Link
The content of one icon is a trait, aspect, quality, feature, attribute, detail, or characteristic of the icon being pointed to.

Key Words
is characteristic of
is a feature of
is a property of

Support / Evidence Link
The content of one icon provides evidence, support, proof, documentation or confirmation for the icon being pointed to.

Key Words
indicates
supports
provides evidence for

Figure 2.1: Dansereau's links

2.1.1. OBJECTIVE

The goal for this study was to explore the cognitive processes a tutor would use to infer a learner's cognitive structure based on the graphic map the learner produced. A first step was to find correlations between constructions used in a map and the learner's understanding as demonstrated in a posttest.

One difficulty with graphic mapping is that the icons and links used may be ambiguous as to their interpretation. For example, in Figure 2.2 a subject has moved "Agreement" and "Contract" together and then has connected them graphically via a TYPE link. This construction is one possible construction for the phrase, "A contract is usually an agreement."

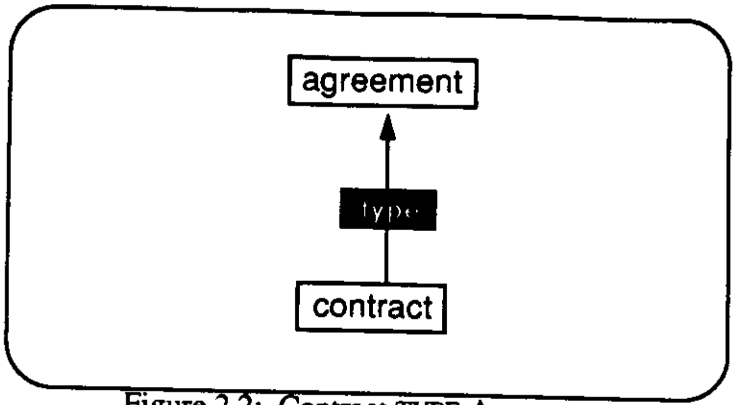


Figure 2.2: Contract TYPE Agreement

In contrast, another subject (Figure 2.3) has used the same two icons, but this time has chosen a SIMILAR link.

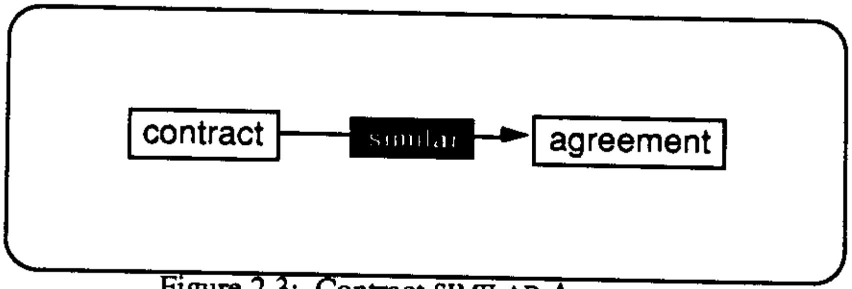


Figure 2.3: Contract SIMILAR Agreement

Does a distinct pictorial representation indicate a different understanding of the text, or a different interpretation of the links?

In Figure 2.4, the subject is apparently representing the phrases, "Consideration may be the giving of another promise," and "If the consideration is a promise, the promise must also be definite." At first glance this might seem an appropriate construction. Literally, however, it would indicate that all promises are definite and any promise is a type of consideration. Does the subject believe that all promises are definite, or is this some form of shorthand? How would we determine what the learner believes? Would tutoring be appropriate at this point?

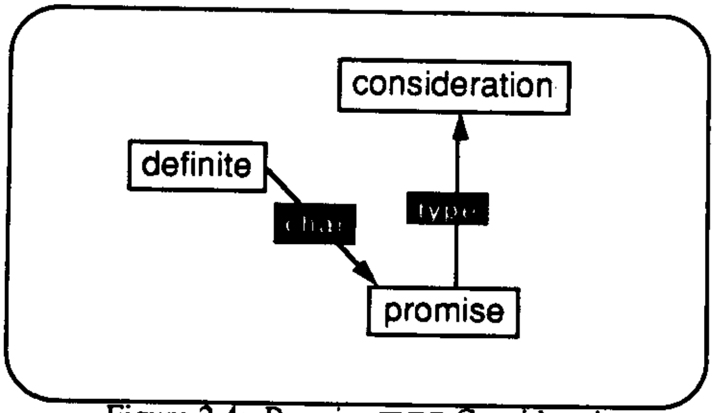


Figure 2.4: Promise TYPE Consideration

In Figure 2.5, a subject uses the SUPPORTS link to connect "Promise" and "Agreement." It is difficult to determine what aspect of the paragraph this construction was intended to represent. Did the subject misunderstand the paragraph, or was there confusion in the use of the links? What kind of information would be necessary to make sense of this construction?

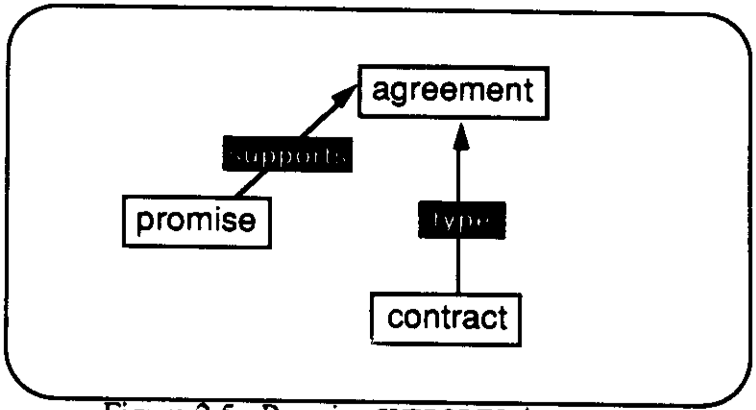


Figure 2.5: Promise SUPPORTS Agreement

In order to tutor, one must be able to determine whether an unexpected construction represents an innovative approach to graphic mapping or a fundamental misunderstanding of the text. If there is a misunderstanding, one must be able to determine whether the error resides in a reference to the text, the task of graphic mapping, interpretation of the icons, or some combination of these.

2.1.2. METHODOLOGY

Subjects built their graphic maps on the computer screen using the initial Sherlock interface (Feifer, Dyer, Baker, Flowers & Read, 1985; 1986). The interface allowed them to easily move and link icons. In addition, the interface generated a transcript of each move and link, and the time elapsed between each action.

Subjects began with a 10 minute introduction to graphic mapping and use of the interface. They were then asked to read the consideration text. Following this, the icons for the consideration text were put on the screen. Subjects were asked to spend at least 45 minutes constructing a map of the text using the links we had described. They were told that they could refer back to the text and the link descriptions at any time. They did not have to use every icon on the screen, but they were to represent every important idea in the text.

Subjects, while building their maps, asked the experimenter questions of three types;

1. Technical, i.e., "How do I get rid of this link?"
2. General, i.e., "What should I do now?"
3. Strategy, i.e., "How would I say that a promise has to be definite?"

Technical questions were always answered. In answer to general questions, portions of the general instructions were repeated. No strategy questions were answered.

After completing the graphic map, subjects answered 20 multiple choice questions on the content of the consideration text. A complete listing of the questions is included in Appendix A. The questions were chosen to ascertain the learner's knowledge about the relationships below illustrated in Figures 2.2 through 2.5:

contract and agreement;

promise and agreement;

definite, promise and consideration.

2.1.3. RESULTS

The first step was to attempt to find a correlation between graphic map constructions and posttest answers. This correlation could enable a tutor to infer, based on a finished graphic map, what the learner believed.

It had been assumed that there would be a variety of constructions for each of the relationships. There was, however, a much greater range of constructions than predicted. The majority of constructions were used only once, and very few were used more than twice. It was thus impossible to find any correlation between constructions and posttest answers.

The next step was to use both the learner's construction and his answers on the posttest to try and infer his cognitive structure. Cognitive structure here refers to both the learner's beliefs about the text and his strategies for building a graphic map.

I began by listing the beliefs a tutor would have for each of the relationships. An attempt was then made to infer what, if any, differences existed between this set of beliefs and the learner's beliefs.

Included below are analyses of three of the constructions used in the pilot study. They were chosen to represent the three possible relationships which can exist between a learner's cognitive structure and a tutor's cognitive structure:

1. Shared concept -- there are no differences between the two cognitive structures;
2. One difference -- some aspect of the learner's cognitive structure differs from the tutor's cognitive structure, leading to an unexpected construction;
3. Interacting differences -- two or more differences between the two cognitive structures interact to lead to an unexpected construction.

2.1.3.1. SHARED CONCEPT

Figure 2.6 is the construction shown in Figure 2.2. Figure 2.7a is the correct knowledge for the concepts in this construction. The correct knowledge is broken into three categories:

1. interpretations (the mapping between the icons and concepts);
2. facts (understanding of the content of the text);

3. strategy (rules for making gm-links between icons).

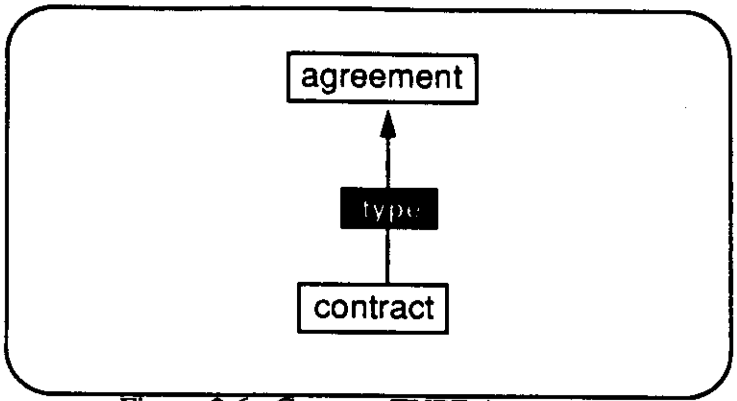


Figure 2.6: Contract TYPE Agreement

a. Tutor's Knowledge

Interpretations:

"agreement" refers to AGREEMENT

"contract" refers to CONTRACT

Facts:

CONTRACT is probably an AGREEMENT

there is less chance that an AGREEMENT is a CONTRACT

Strategy:

If an X is probably a Y

and there is less chance that a Y is an X

then make a TYPE link from X to Y

b. Differences between the tutor's knowledge and the learner's knowledge

Interpretations:

none

Facts:

none

Strategy:

none

Figure 2.7: Shared Concept

The first step in analyzing the subject's graphic map construction is to determine what gm-link the tutor would make. In this case, the tutor's interpretations, facts, and strategy, lead to a TYPE link. This is the same gm-link which the subject made.

Since the posttest indicated that this subject believed the same facts as the tutor, and the subject made the same link the tutor would have made, it is assumed that the all relevant knowledge is shared by the learner. Thus Figure 2.7b indicates that there are no differences. Thus, no tutoring is necessary here.

2.1.3.2. ONE DIFFERENCE

When the subject makes an unexpected move, some part of the learner's cognitive structure must differ from the tutor's cognitive structure. Figure 2.9a contains the tutor's beliefs for the concepts in learner's construction in Figure 2.8 (repeated from Figure 2.4). These beliefs do not lead to the learner's construction.

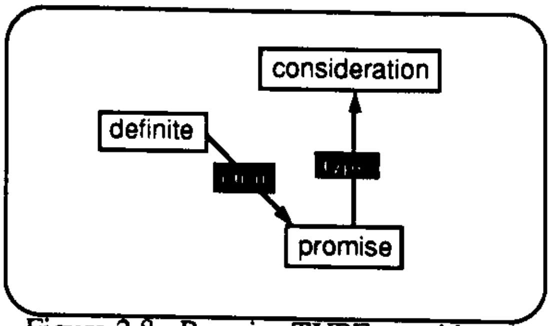


Figure 2.8: Promise TYPE consideration

a. Tutor's Knowledge

Interpretations:

"definite" refers to
DEFINITE THINGS

"promise" refers to
PROMISE

Facts:

CONS-PROM is probably
CONSIDERATION

CONS-PROM is probably
DEFINITE THING

Strategy:

If an X is probably a Y

and there is less chance that a Y is an X

then make a TYPE link from X to Y

b. Differences

Interpretations:

"promise" refers to
CONS-PROM

Facts:

none

Strategy:

none

Figure 2.9: One Difference

When the learner does not make the gm-links that are expected, the next step is to find possible differences between the tutor's beliefs and the learner's beliefs that explain the construction. The posttest indicated that the subject who made the construction in Figure 2.8 believed the same facts as the tutor. Therefore any differences must involve either interpretations or strategy. Figure 2.9b shows that one difference between the learner's interpretations and the tutor's interpretations can explain the learner's construction. That change involves the interpretation of the "promise" icon. The "promise" icon had been interpreted as referring to the concept promise. The subject apparently interprets "promise" as referring to CONS-PROM (consideration which is a promise).

2.1.3.3. INTERACTING DIFFERENCES

Once again, the tutor's knowledge in Figure 2.11a does not lead to the learner's construction in Figure 2.10 below (repeated from Figure 2.5). This subject's posttest does indicate one wrong belief, that every agreement has consideration. But this one difference still cannot explain the learner's construction. Therefore it appears that other differences in interpretations or strategy must be interacting with this difference in facts. Figure 2.11b contains three differences that together explain the learner's construction.

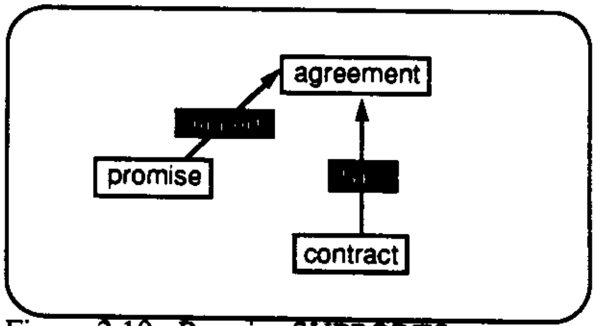


Figure 2.10: Promise SUPPORTS agreement

The three differences are:

1. The subject interpreted "promise" as referring to the promise which a party makes as part of a contract (CONS1) rather than the generic promise.
2. The subject over-generalized this event (party1 promising) as an attribute of AGREEMENT rather than of CONTRACT.
3. The subject used a rule which said to make a support gm-link if something is attribute, rather than a part gm-link.

Interacting differences compound the number of possible explanations that must be checked.

a. Tutor's Knowledge	**b. Differences between the tutor's knowledge and the learner's knowledge**
Interpretations:	***Interpretations:***
"agreement" refers to AGREEMENT	
"promise" refers to PROMISE	"promise" refers to CONS1
facts:	***facts:***
CONS1 is probably a PROMISE	CONS1 is an attribute of AGREEMENT
CONS1 is an attribute of CONTRACT	
Strategy:	***Strategy:***
If X is an attribute of Y	**If** X is an attribute of Y
then make a PART link from X to Y	**then** make a supports link from X to Y

Figure 2.11: Interacting Differences

2.1.4. SUMMARY OF FIRST PILOT STUDY FINDINGS

This first pilot study uncovered two problems that must be addressed: (1) ambiguity introduced by the link choices offered the learner, and (2) the difficulty in determining the learner's cognitive structures from actions alone.

It was well recognized, in developing a learning strategy based on AI knowledge representations, that compromises were necessary. A happy medium had to be arrived at between complexity and simplicity (Holley & Dansereau, 1984a). Unfortunately, as the simplicity of using a representation system increases, the ambiguity of the representations also increases.

In examining the ambiguity in graphic mapping using Dansereau's links, the following was observed:

- There is too much variability in graphic maps to infer, on the basis of the map alone, the user's cognitive structure.

- The only instances in which the map was any indication of the user's cognitive structure were those in which the link arrangement was what would have been expected.
- There are at least three possible reasons a user would not make the anticipated link arrangement: differences in understanding of the text domain, differences in strategies for building graphic maps, and differences in how the icons are mapped to concepts in the learner's cognitive structures.

2.2. SECOND PILOT STUDY

During December of 1986 a second pilot study was conducted (Feifer, Dyer, Baker, & Flowers, 1987). This study attempted to look at the cognitive processes of someone building a graphic map and the effects of various interventions. Subjects were again asked to construct graphic maps for the consideration text. They were asked to use the Sherlock links listed in Chapter 1 (Figure 1.4).

Subjects in this study were also asked to think out loud during the entire process (Ericson & Simon, 1980). Sessions were video taped for later analysis. In addition, when the subject made a link that the experimenter considered a mistake, the experimenter provided a remedial intervention.

This pilot study was designed to gather information for the following purposes:

1. definition of a formal model of the cognitive process involved in building a graphic map;
2. development and evaluation of remedial interventions which could be implemented in an automated system.

2.2.1. METHODOLOGY

Subjects were given the same training and instructions as in the first pilot study. In addition they were asked to verbalize everything they were thinking. During the first five minutes most subjects had to be reminded to think out-loud. After that they seemed to have little trouble keeping a monolog going. After each link, the subject was interrupted and asked to state as clearly as possible the reason he made the link.

This process led to a definition of:

1. the steps a subject took in planning a link;
2. the knowledge a link was based on;
3. gaps between intentions and actions.

The data from this study are included in Appendix B.

2.2.2. COGNITIVE TASKS IN GRAPHIC MAPPING

Using the data from the first study, three different aspects of a learner's cognitive structure were identified. These three aspects interact to determine the graphic constructions the learner will build:

1. the learner's strategy for constructing a graphic map;
2. the learner's current understanding of the content of the text;
3. the learner's interpretation of icons on the screen.

Using the protocols from this second study, cognitive subtasks were identified for each of these three aspects. The following is a list of these sub-tasks. This list is not intended as a sequential description of the process a learner uses in constructing a map. Rather, subjects in the study used some subset of these processes, not necessarily in this order, to construct their graphic maps.

2.2.2.1. THE LEARNER'S STRATEGY FOR CONSTRUCTING A GRAPHIC MAP

The learner's strategy can be modeled as a set of plans and goals. The learner sets a goal, must find a plan which is appropriate for that goal, and then attempt to implement that plan.

(1) Setting a goal

The learner is given a top level goal: to build a graphic map that represents the text. To accomplish this task, the learner must set appropriate sub-goals. It is by allowing the learner to set his own sub-goals that Sherlock provides a learner-directed environment. The learner decides where to start and how to proceed.

There are three major sub-goals learners set for themselves in constructing a graphic map: (a) link an icon to something else, (b) find the appropriate link for two icons on the screen, and (c) make an appropriate construction for a concept in the text.

(2) Finding a plan

The mapping task involves discovering plans that will help to realize the above goals. Usually this will involve setting new subgoals. For example, a plan for linking an icon to something else might be:

1. Find another icon which is related to this icon;
2. determine how these two icons are related to each other;
3. determine what type of gm-link would best represent this relationship.

Below are examples of some of the plans subjects articulated for sub-goal #3, choosing a gm-link:

> In trying to decide if the appropriate link between two concepts is EQUIV or SPEC, I determine if they both cover the same area. If they do, I use an EQUIV link. If one covers more, then I use a SPEC link to the one that covers more.

> If two icons represent physical things, then I rule out LEAD as an appropriate link.
>
> If one icon is a description of another, then I make an EQUIV link between them.
>
> If X is a characteristic of Y then I put a PART link between them because X is part of the definition of Y.

2.2.2.2. THE LEARNER'S UNDERSTANDING OF THE CONTENT OF THE TEXT

In understanding text, the learner must do two things: map the words in the text onto concepts and determine the relationship between those concepts.

(1) Mapping Words to Concepts

The learner must determine how certain words are associated with particular concepts. The nature of this task is determined by whether the learner already knows the word and/or the concept, and whether the learner's existing knowledge is consistent with the information in the text. There are three situations that exist:

1. The word and the concept are known to the learner.

In this case the learner need only map any new information onto the existing concept. For example, most subjects know the word "contract" and associate it with a concept which is consistent with the information in the text. The purpose of the consideration text is to help the learner add new information to his concept of a contract.

2. The word and the concept are new to the learner.

In this case the learner must first build the concept from existing concepts. Then the learner must associate the new concept with the new term introduced in the text. For example, most subjects were not familiar with the word "forbearance." Although they could think of examples of people not doing something they could have done, they had never thought of these as belonging to one conceptual class. Thus, they also had no concept to go with the word "forbearance."

3. The word is known to the learner, but the text is using it to refer to a new concept.

The learner must again begin by building the concept. The difficulty in this case, however, is that the learner must first recognize that a new concept is necessary, rather than mapping any new information onto the existing concept. The learner must then associate the word with the new concept, in the present context.

For example, the subject making the link in Figure 2.12 explained, "When a contract is made, both parties have to consider it to see if it is a good contract."

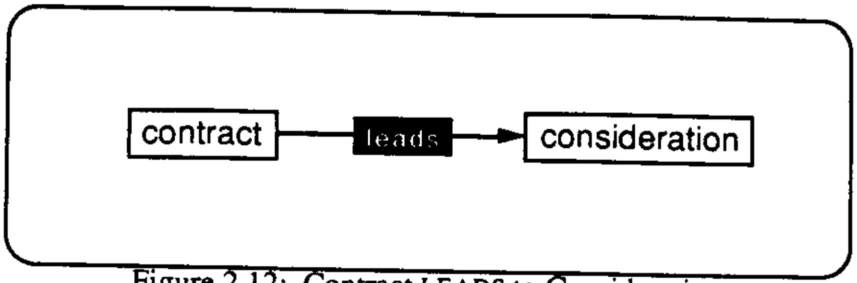


Figure 2.12: Contract LEADS to Consideration

Subjects interpreted the "Consideration" icon in three ways: (1) Something done in return for a promise; (2) Doing something nice for someone, as in "she showed him some consideration"; (3) To think about something, as in "He felt her suggestion deserved some consideration." In Figure 2.12 the subject is probably unaware that consideration is a technical term for a necessary condition of contract.

(2) Determining the Relationship Between Concepts

There are two different kinds of relationships which must be determined in order to understand textual material:

1. relationships which are a part of the text;

2. relationships between concepts in the text and the learner's existing knowledge.

Determining the relationship between two concepts, both of which are introduced in the text, involves making explicit what is implicitly contained in the text. The learner must parse the text into some internal meaning. The learner must recognize, for example, that a phrase of the form:

An X may take the form of an A, a B or a C

indicates that:

Concept X has three sub-types: A, B, and C.

In order for the learner to determine the relationship between a new concept and an old concept, two conditions must be true:

1. The learner must have sufficient background knowledge to understand the new material.

2. The learner's existing knowledge structure must be at least compatible with the new material.

(1) Supplying Necessary Background Knowledge

Any text is written with the assumption that the learner has some background knowledge. When the learner does not have the necessary background knowledge, he has no place to link new information. For example, learners often cannot link "forbearance" to anything because they do not know what forbearance really means. In this case the tutor must supply enough information about forbearance to allow the learner to relate that concept to other concepts in the text.

(2) Modifying Old Beliefs Which Conflict With New Knowledge

When textual knowledge conflicts with the learner's current knowledge, the learner's knowledge structures must be modified before any integration can take place. For example, we find that many learners begin with the mistaken belief that all contracts are written. This belief hinders their efforts to define the relationship between contract and agreement.

2.2.2.3. THE LEARNER'S INTERPRETATION OF THE ICONS ON THE SCREEN

We have found that we can seldom assume that the interpretation learners place on an icon is identical to that intended when we created the icon.

(1) Parsing The Text Which Appears Within an Icon

Natural language, at best, is an imprecise shorthand for expressing ideas. When single words or small groups of words are taken out of context, it only exacerbates the problem. The interface would be too cluttered if each concept were defined completely on the icon. The learner must, then, decide what conceptual entity an icon refers to.

As an example, a simple set of icons (Figure 2.13) was used to familiarize subjects with the interface and the links available. Icons were chosen to provide a clear example of each type of link. Subjects were not asked to read a text in this case, only to connect the icons using their previous knowledge. There was thus no mapping of words in a text to concepts. All variation, then, can be attributed to mapping icons to concepts.

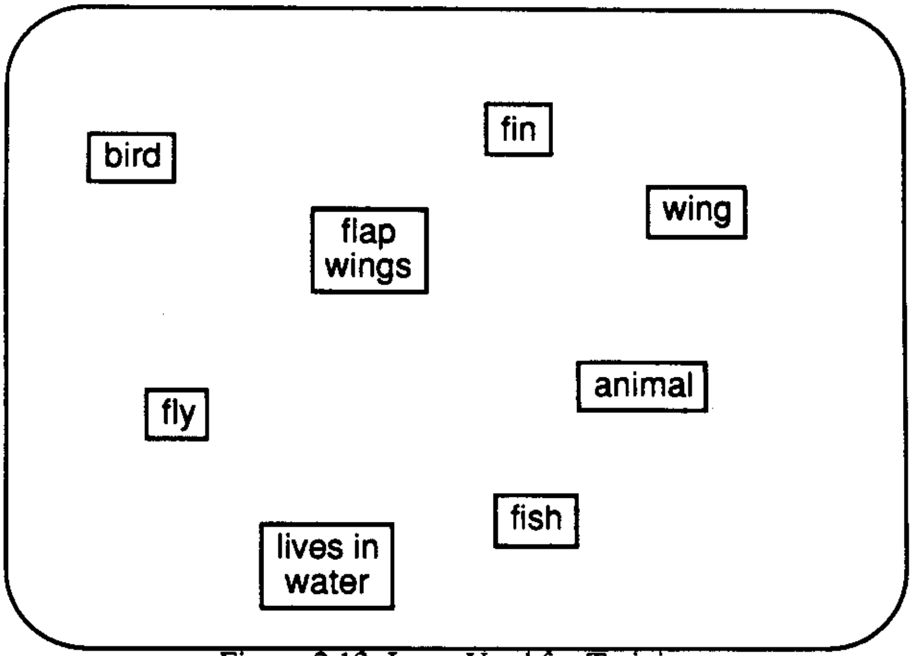


Figure 2.13: Icons Used for Training

The two icons for the LEADS link were "flap wings" and "fly." Most subjects suggested that "flap wings" LEADS to "fly." One subject, however, believed that "flap wings" referred to some part of a bird's wing with which she was not familiar and that "fly" referred to an insect.

Two things were interesting about this subject's interpretations. First, she was not using "flap wings" to refer to a concept she wanted to use, or even a concept she thought existed. She was creating an unknown concept to go with the icon. Second, it was not that she *chose* to interpret "fly" as a noun; it just did not occur to her that it could also be a verb.

(2) Mapping concepts to icons

When a learner wishes to represent graphically a concept or relationship he must choose an icon on the screen. This is complicated by the fact that there is not a one-to-one correspondence between icons and concepts any given learner will want to represent.

In Figure 2.8, for example, the subject used the icon "promise" to refer to the type of consideration which is also a promise. This does not seem to be a problem of parsing; most subjects interpret the word "promise" as referring the more generic concept of promise. Rather, it is more likely that the subject wanted to represent the concept of considerations which are promises and, finding the "promise" icon, decided that it was close enough.

2.3. SUMMARY

In order to build an intelligent tutoring system for graphic mapping we must be able to represent the elements of a learner's cognitive structure which determine the graphic map he will produce. In order to do this we must:

1. define the elements of a learner's cognitive structure that determine the graphic map he will produce;
2. develop a mechanism for automatically inferring those elements.

2.3.1. REPRESENTING A LEARNER'S COGNITIVE STRUCTURE

It was possible in the first pilot study to infer beliefs which might account for learner constructions. These beliefs were represented using three components:

1. the learner's interpretations for the icons;
2. the facts which the learner believes to be true;
3. a rule indicating the learner's strategy for translating these beliefs into a gm-link.

These three components are necessary and sufficient to predict the gm-link a learner will make. Thus Sherlock must have representations for these three components in order to understand a learner's graphic map.

2.3.2. AUTOMATICALLY INFERRING THE COGNITIVE STRUCTURE

Data from the first pilot study, however, indicated that it was impossible to infer what a learner was thinking by examining only his finished graphic map. In order to automatically infer the learner's cognitive structure, then, Sherlock must be able to follow the *process* of graphic mapping rather than just examine the final product.

The second pilot study was designed to provide the information needed to follow the process of graphic mapping. Based on the protocols from this study it was possible to define the sub-tasks necessary to build a graphic map. These tasks are:

A. Graphic mapping strategy
   1. setting a goal
   2. finding a plan to satisfy the goal

B. Text understanding
   1. mapping words to concepts
   2. determining the relationship between concepts

C. Icon interpretation
   1. parsing icon text
   2. mapping concepts to icons

These tasks serve as the basis for the model of graphic mapping used in Sherlock.

Chapter 3

Representing Text Knowledge

3.0. INTRODUCTION

The first step in developing a tutoring model is to define a means of describing or representing what someone knows. The type of representation we define must be capable of providing the information needed in the tutoring process.

In analyzing the data from the first pilot study, the tutor's knowledge was described as a set of natural language statements. This method of describing knowledge was adequate for defining the kind of differences which explain a learner's move. But there are two weaknesses inherent in using natural language to describe knowledge:

1. It is difficult to determine meaningful differences between two natural language descriptions. If two sets of statements are different, but will lead to the same inferences, then there are no meaningful differences.
2. Natural language statements are too ambiguous for a computer to use directly.

Sherlock's knowledge representation scheme must provide a non-ambiguous means of describing what someone knows. The knowledge representation must allow us to derive from the representation the information we could attain by asking questions of the person whose knowledge is represented. To allow this, the representation scheme must specify:

1. a primitive set of concepts;

2. a primitive set of relations;
3. processes which operate on concepts and relations.

Other concepts are built on top of the primitive set of concepts and relations. The processes determine the inferences which are possible given a specific knowledge representation.

Representing knowledge in this non-ambiguous manner allows us to put knowledge into an intelligent tutoring system that the computer can understand. As educators, even if we were not concerned with intelligent tutoring systems, a non-ambiguous representation also gives us a framework for discussing knowledge:

1. What does a learner know?
2. What do we want a learner to know?
3. How do we help the learner know it?

3.1. DECLARATIVE KNOWLEDGE IN SHERLOCK

Sherlock uses a localist, spreading activation network package, developed by Michael Gasser (1988). Gasser designed this package to model language generation. Part of designing Sherlock required specifying s-links[1] and nodes to represent Sherlock's knowledge of the consideration text. What follows is a description of Gasser's network package, described in detail in Gasser (1988).

The two basic elements in the semantic network are nodes and s-links.

3.1.1. NODES

A node represents the concept of a fact, entity, event or state. A node will be illustrated by English text within a rectangle (Figure 3.1). The English text has no meaning to the network; it is only intended to make the illustration more readable. The actual meaning of a node in the network is determined by its connections to other nodes. Thus, the node BIRD[2] might be defined by links which indicate that it belongs to the conceptual classes of LIVING-THING and FLYING-THING. This node is given the label BIRD to indicate that it represents the concept people think of when they hear the word "bird."

---

[1] Links in the semantic network will be referred to as s-links to distinguish them from gm-links in the graphic map.

[2] The name of a node will always appear in SMALL CAPS.

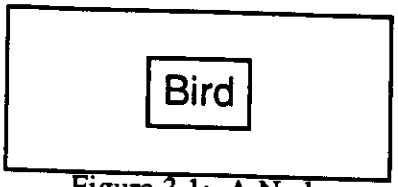


Figure 3.1: A Node

There are three types of nodes in Sherlock's semantic network:

1. heads;
2. roles;
3. inhibition.

A head node represents a main concept. A role node represents the concept of an attribute, characteristic, part or argument of a head node For example, WING might be a role node belonging to the head node BIRD. Any given node may be both a head node and a role node. WING, for example, might also be a head node for role nodes which represent concepts associated with wings.

An inhibition node represents a mutually exclusive category. For example, an inhibition node for the category animal represents the concept that each type of animal is mutually exclusive. This is not to be confused with the concept of animal, which is represented by an ANIMAL node.

A node's type is determined by how it is linked to other nodes. The function of each of these three types of nodes will described more completely in conjunction with the s-links which define it as being of that type.

3.1.2. S-LINKS

An s-link between two nodes indicates that there is some kind of relationship between those two nodes. Each s-link actually consists of two connections, one in each direction, each having a possible weight between 0 and 1.

All processing in the network is done through a form of spreading activation. The s-links determine how that activation spreads. Activation is begun by placing a numerical weight on a node. This activation is then allowed to spread through all of the links which go out from this node; that is, the numerical weight is increased on nodes which are on the other end of these links. The higher the weight of a link, the higher the activation which can reach a node through that connection.

There are four kinds of s-links in Sherlock's semantic network:

1. GENERALIZATION;
2. EQUIVALENT;
3. HAS;
4. INHIBITION.

3.1.2.1. GENERALIZATION S-LINK

The simplest s-link is a GENERALIZATION link. It identifies the conceptual classes of which a node is a member. It is illustrated as a single-headed arrow between two nodes (Figure 3.2). The node at the bottom of the arrow is a specialization of the node which is being pointed to. As with the other s-links, a GENERALIZATION link consists of connections in both directions. The direction of the arrow in the figure indicates the direction of the connection with the higher weight. Thus in Figure 3.2 the weight from ROBIN to BIRD will be higher than the weight from BIRD to ROBIN.

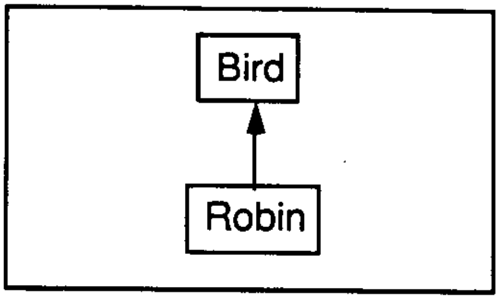


Figure 3.2: GENERALIZATION Link

This differential in weights is what allows the GENERALIZATION link to operate as an is-a link during inheritance. An is-a link indicates that the concept at the bottom of the link inherits the properties of the concept at the top of the link. Thus ROBIN inherits the properties of BIRD. BIRD could also have a GENERALIZATION link to ANIMAL, meaning that ROBIN inherits the properties of BIRD and ANIMAL. All of the concepts above a concept in an IS-A hierarchy are referred to as the ancestors of that concept. All of the concepts below a concept in an IS-A hierarchy are referred to as the descendents of that concept.

The network recognizes that anything that is true of a node is also true of any of its descendents. It is less likely that something which is true of a node will also be true of any of its ancestors. The weights on a GENERALIZATION link can be roughly interpreted as the likelihood that the concept at one end is an example of the thing at the other end. Thus the greater weight on the connection from ROBIN to BIRD indicates that there is a greater chance that a robin is a bird than there is that a bird is a robin.

The weights can only roughly be interpreted in this manner since their actual importance is in establishing relative strengths. The actual weight of any connection in the network is not important; only its relationship to the other strengths in the network is important.

3.1.2.2. EQUIVALENT S-LINK

An EQUIVALENT s-link is a GENERALIZATION link with strong and equal weights in both directions. An EQUIVALENT s-link is illustrated by a two headed arrow (Figure 3.3).

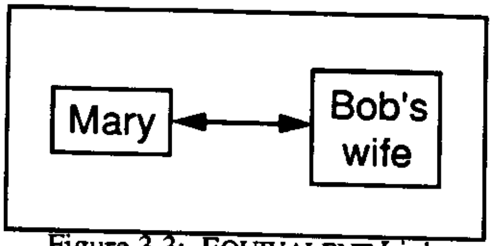


Figure 3.3: EQUIVALENT Link

Because the weights are high and equal, the network recognizes that anything which is true of once concept will also be true of the other. Thus any node which is a role or descendent of MARY will behave during spreading activation as if it is also attached to BOB'S WIFE.

3.1.2.3. HAS S-LINK

A role is linked to the node to which it belongs with a HAS S-link. A HAS S-link is illustrated with a one-headed arrow containing the "has" label (Figure 3.4).

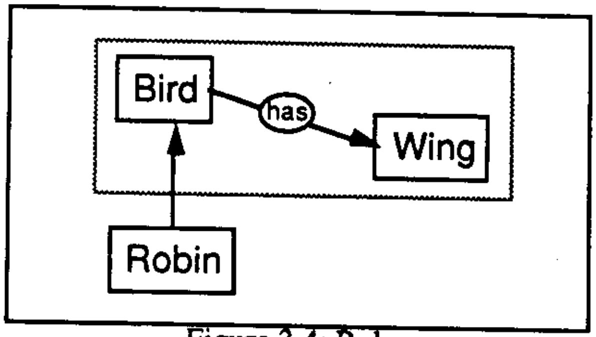


Figure 3.4: Roles

The weight from a role to its owner can be roughly interpreted as an indication of how salient a feature it is of the owner concept. Thus a strong weight from WING to BIRD will indicate a wing has a good chance of making us think of a bird. The weight from the owner to its role indicates how essential that attribute is to the owner concept. The link from BIRD to WING would be strong indication that having a wing is important evidence that something is a bird.

The GENERALIZATION link from ROBIN to BIRD in Figure 3.4 allows the network to infer that robins have wings although this is not explicitly represented. The WING role for ROBIN would only be indicated if there were something special to say about it, for example to indicate that all robins' wings are of a certain color.

It is this ability to inherit attributes which gives the network much of its power. At each level of an **is-a** hierarchy we need only define those roles which distinguish a concept from its ancestors.

3.1.2.4. INHIBITION

If you reach into your pocket and find a coin, you can probably identify its denomination without taking it out to examine every detail. If it is bigger but thinner than a nickel, but too small to be a half dollar, you will decide that it is a quarter without seeing that George Washington is on one side or that the words "quarter dollar" are engraved on the other side. You can do this because once you have decided that it is a coin, you know that it must be one of six possibilities (assuming that you have no foreign currency in your pocket). Not only must it be one of these things, it cannot be more than one. It cannot be both a quarter and a nickel.

If we know that something belongs to one of several mutually exclusive categories we do not need complete evidence to classify it. We merely need enough evidence to say that it is more likely that it belongs to one of the categories, than it is that it belongs to any of the others. In order for Sherlock to take advantage of this we must explicitly represent mutually exclusive categories.

A mutually exclusive relationship between concepts is represented by linking those concepts to an inhibition node. An inhibition node is illustrated by an X inside of a circle (Figure 3.5).

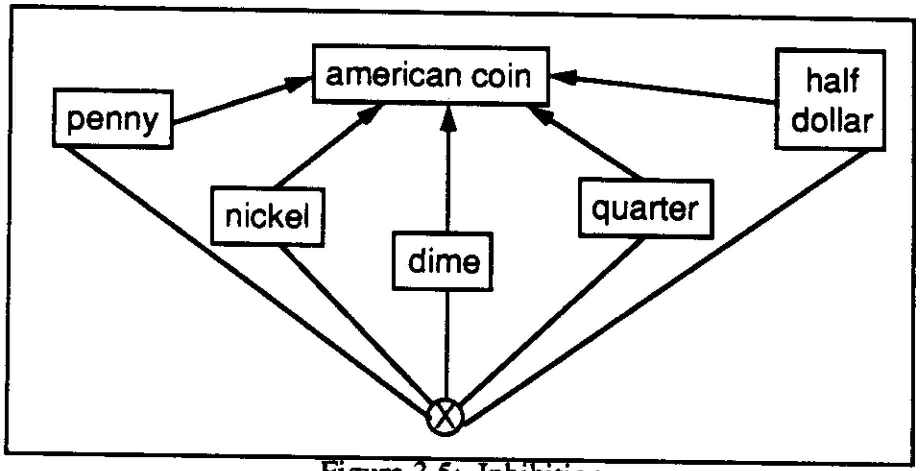


Figure 3.5: Inhibition

The nodes that it inhibits are linked by lines to the inhibition node. The INHIBITION links in Figure 3.5 would indicate, for example, that if something is a quarter it cannot be a penny, nickel, dime or half dollar. INHIBITION links have no meaningful weights. During activation, if any of the nodes attached to a particular INHIBITION link is activated, the activation of all other nodes attached to the same INHIBITION link will be dampened.

The group of concepts connected to one inhibition node is known as a winner-take-all (WTA) network (Feldman & Ballard, 1982). As any nodes in a particular WTA network become activated, they compete with the other nodes in that network. Eventually the node with the highest activation in that network "wins" the activations from the other nodes in the same network.

If not for INHIBITION s-links, it would be very hard for the network to classify concepts. This is because the network, similar to the person reaching into his pocket, seldom has enough direct evidence to classify a concept. By taking advantage of the knowledge of mutually exclusive categories the network can make optimum use of whatever evidence it does have.

The GENERALIZATION, EQUIVALENT, HAS and INHIBITION links are supplied in Gasser's network package.

3.2. REPRESENTING THE CONSIDERATION TEXT

The localist spreading activation network just described is used to represent the content of the consideration text presented in Figure 1.3. The resulting representation will be referred to as Sherlock's semantic network for the consideration text. The semantic network enables Sherlock to operate as if it knows the content of the text. As with any text, the authors of the consideration text assume that the reader has some background knowledge. We thus begin with the basic concepts upon which the consideration text is based.

3.2.1. BASIC CONCEPTS

There are three primitive concepts in Sherlock's semantic network: acts, facts and things.

Primitive refers to the fact that these concepts are not defined. The network does not know what an act is, only that anything which belongs to the conceptual class ACT will have certain attributes. A portion of the representation for these concepts is illustrated in Figure 3.6. All other concepts are defined as specializations of one of these three primitives.

3.2.1.1. ACTS

An ACT is an event or action. According to Figure 3.6, ACT has three roles which Sherlock will be concerned with: RESULTS, CONDITIONS, and ACTOR. The ACTOR of an ACT is the agent who carries out the ACT or causes the ACT to happen. A condition of an ACT is a fact that must be true in order for the ACT to occur. The restriction that a condition must be a fact is represented by a GENERALIZATION s-link from CONDITIONS to FACT. Similarly a result is a fact that will be true after the act occurs.

Every type of ACT or instance of ACT inherits these roles. The conditions for a particular ACT will be represented by specializations of that ACT's CONDITIONS role.

One type of ACT shown in Figure 3.6 is MTRANS. MTRANS is a primitive borrowed from Schank (1975) which represents any kind of mental transfer or communication. The roles represented in Sherlock for MTRANS are:

1. FROM - the entity the information is coming from;
2. TO - the entity receiving the information;
3. CONTENT - the information being transferred.

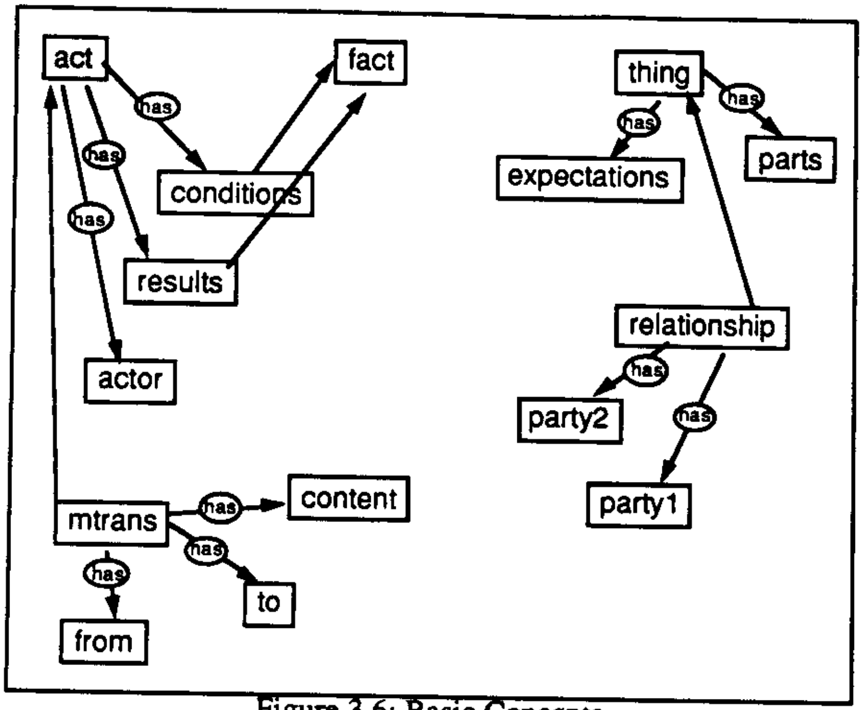


Figure 3.6: Basic Concepts

3.2.1.2. FACTS

A fact is a statement. Most facts are stated implicitly in the network through the use of s-links. The HAS s-link from RELATIONSHIP to PARTY1 , for example, can be seen as representing the fact that PARTY1 is a part of RELATIONSHIP. The FACT node, on the other hand, represents explicitly stated facts. Descendents of the FACT node represent statements such as:

Bob has given the money to Sue

A legal obligation exists

A moral obligation exists

3.2.1.3. THINGS

A THING is an entity or physical object. According to Figure 3.6, THING has one role which Sherlock is concerned with: EXPECTATIONS. EXPECTATIONS are descendents of facts which one would expect to be true if a particular thing exists. For example, if ice exists in a particular place, one would expect it to be cold in that place.

One subtype of THING illustrated in Figure 3.6 is RELATIONSHIP. As with the primitive concepts, RELATIONSHIP is not really defined. All Sherlock knows is that a relationship has two roles: PARTY1 and PARTY2.

3.2.2. AGREEMENT

The concept of agreement is mentioned in the text, but not defined. The author of the consideration text is relying on the reader's knowledge of the concept of an agreement, defining contract only in terms of how it differs from any other kind of agreement. The representation of agreement in Sherlock's semantic network is illustrated in Figure 3.7.

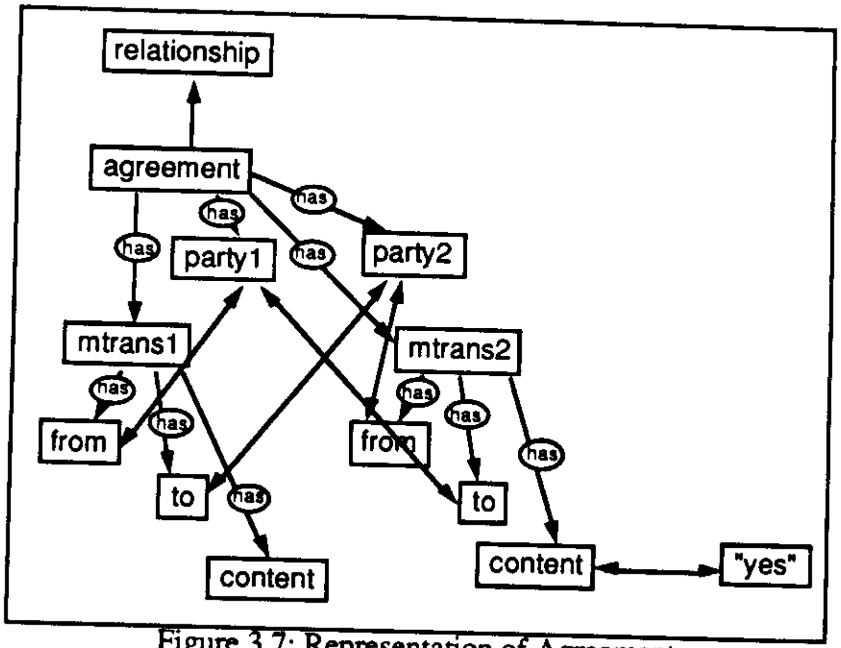


Figure 3.7: Representation of Agreement

AGREEMENT is a specialization of RELATIONSHIP, and thus inherits the two roles: PARTY1 and PARTY2. In addition, AGREEMENT has two new roles: MTRANS1 and MTRANS2. Although not shown in Figure 3.7, each mtrans is a specialization of MTRANS and thus inherits the roles FROM, TO, and CONTENT. The inherited roles of MTRANS1, MTRANS2 and AGREEMENT are represented at this level in order to indicate the relationship between them in an agreement.

Figure 3.7 indicates that the FROM role of MTRANS1 is equivalent to PARTY1 in an agreement, and the TO role is equivalent to PARTY2. The reverse is true for MTRANS2. In other words the agreement consists of two communication acts, one from party1 to party2, and the second from party2 to party1. In the proto-typical agreement the content of the second communication is some kind of affirmative, indicated by the EQUIVALENT link between MTRANS2:CONTENT[3] and "YES".

3.2.3. PROMISE

The representation of a promise is illustrated in Figure 3.8.

---

[3] The colon is used as a shorthand for referring to the role of a specific node. MTRANS2:CONTENT refers to the CONTENT role of MTRANS2.

PROMISE is a type of MTRANS whose CONTENT role consists of PR-ACT (the act that is being promised) and PR-CONDITION (condition for the promise). The FROM and TO roles of MTRANS are also inherited by PROMISE, but are not explicitly represented since there is no special information about them which has to be indicated at the level of PROMISE. The PR-ACT is an act whose ending time is in the future (not shown in the figure). The PR-CONDITION role is for any conditions which are communicated as part of the promise. For example, John promises Mary that he will go to the museum with her if it is raining. In John's promise going to the museum is the PR-ACT; raining is the PR-CONDITION.

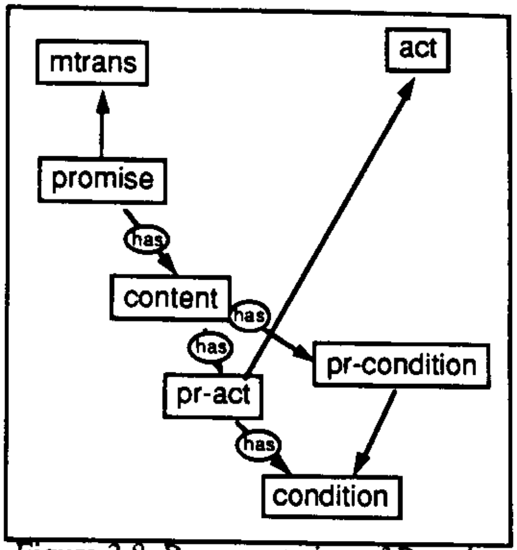


Figure 3.8: Representation of Promise

Not every promise will have a condition, but the role is there if it is needed. The representation at this level tells the network that if there is a condition, it is also a condition of the PR-ACT, indicated by the GENERALIZATION link from PR-CONDITION to PR-ACT:CONDITION.

3.2.4. CONSIDERATION

The text's representation of consideration is illustrated in Figure 3.9.

> *Consideration must: (a) be lawful*

CONSIDERATION is a specialization of LEGACT which is a type of ACT. LEGACT and ILLEGACT are two mutually exclusive sub-types of ACT as are RECQACT (required act) and OPTACT (optional act). Notice that the network has no knowledge of what it means to be a legal act, just that if an act is illegal that it cannot be legal, indicated by the inhibition node between LEGACT and ILLEGACT.

> *Consideration may be the giving of another promise, or it may be an act. Consideration may also consist of forbearance*

There are three sub-types of CONSIDERATION: (1) CONS-ACT, consideration which is an act, (2) CONS-PROM, consideration which is a promise, and (3) CONS-FORB, consideration which is a forbearance. Each of these is represented as the intersection between CONSIDERATION and either ACT, PROMISE, or FORBEARANCE. That is, CONS-ACT represents anything which is both a CONSIDERATION and an ACT. The three sub-types of consideration are linked to an inhibition node.

> *(b) require conduct that is not already required by law, or must prohibit conduct which is not already prohibited by law*

The two types of consideration which require conduct are CONS-ACT and CONS-PROM. The difference between CONS-ACT and CONS-PROM is that CONS-ACT is the conduct itself and takes place at the time that the contract is made; in CONS-PROM the conduct is the content of a promise and will take place at some time in the future. Thus there is a GENERALIZATION s-link from CONS-ACT to OPTACT, an act which is optional. But for CONS-PROM the GENERALIZATION s-link is from CONS-PROM:CONTENT:PR-ACT to OPTACT.

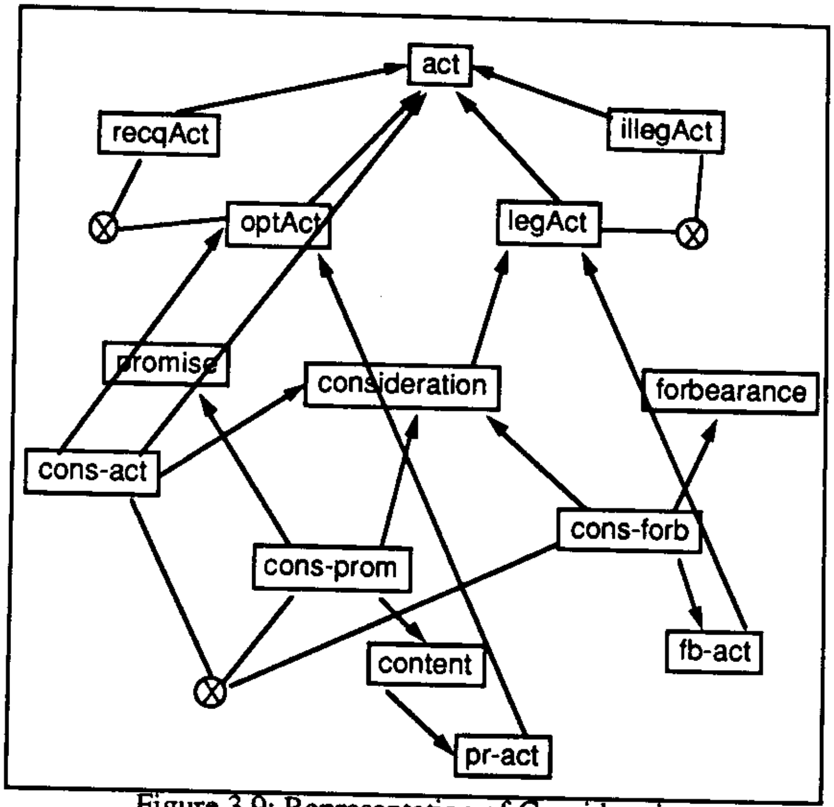


Figure 3.9: Representation of Consideration

> *or must prohibit conduct which is not already prohibited by law*

The only type of forbearance which prohibits conduct is CONS-FORB, and thus this restriction refers only to CONS-FORB. FORBEARANCE has the role FB-ACT, not illustrated, which is the act being avoided. The restriction that consideration prohibit conduct not already prohibited by law is represented by making a GENERALIZATION s-link from CONS-FORB:FB-ACT to LEGACT.

3.2.5. CONTRACT

The aspects of a contract mentioned in the text are illustrated in Figures 3.10 and 3.11.

> *A contract is usually an agreement in which, in effect, one party says to another, "If you do this for me, I shall do that for you." The thing which one person asks another to do in return for a promise is the consideration.*

Contract is represented as a specialization of agreement (Figure 3.10). Subjects occasionally became confused by the use of the word "usually," saying that if a contract is usually an agreement, but not always an agreement, than it cannot be a sub-type of agreement. But "usually" refers to the description of contract which follows. All contracts are agreements, but they do not all fit this proto-typical description exactly.

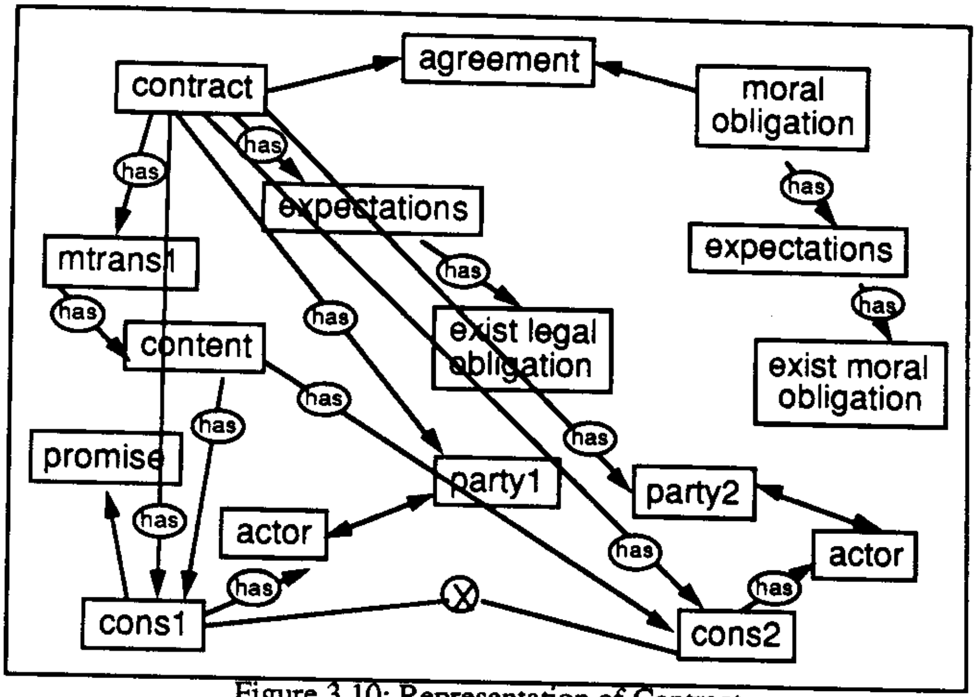


Figure 3.10: Representation of Contract

Since this is the only type of contract the reader is told about, this is the only type of contract currently represented in Sherlock. The other sub-type of agreement represented is a moral obligation. If a contract exists there is an expectation that a legal obligation exists; if there is a moral obligation, there is no expectation that a legal obligation exists. There is no inhibition node between these two types of agreement, meaning that it is possible to have a contract which is also a moral obligation.

The role CONTRACT:MTRANS1 is inherited from AGREEMENT. CONTRACT:MTRANS1:CONTENT is inherited from MTRANS. What distinguishes a contract from other types of agreement (according to the text) is the content of MTRANS1. MTRANS1 is made up of CONS1 and CONS2, which are both specializations of CONSIDERATION (the GENERALIZATION s-links to CONSIDERATION are not illustrated in Figure 3.10). The text only identifies CONS2 as being consideration, but both are actually consideration. The only restriction on CONS1 is that it must be a promise, represented by a GENERALIZATION s-link from CONS1 to PROMISE.

If you do this for me, I shall do that for you.

The relationship between CONS1 and CONS2 is illustrated in Figure 3.11.

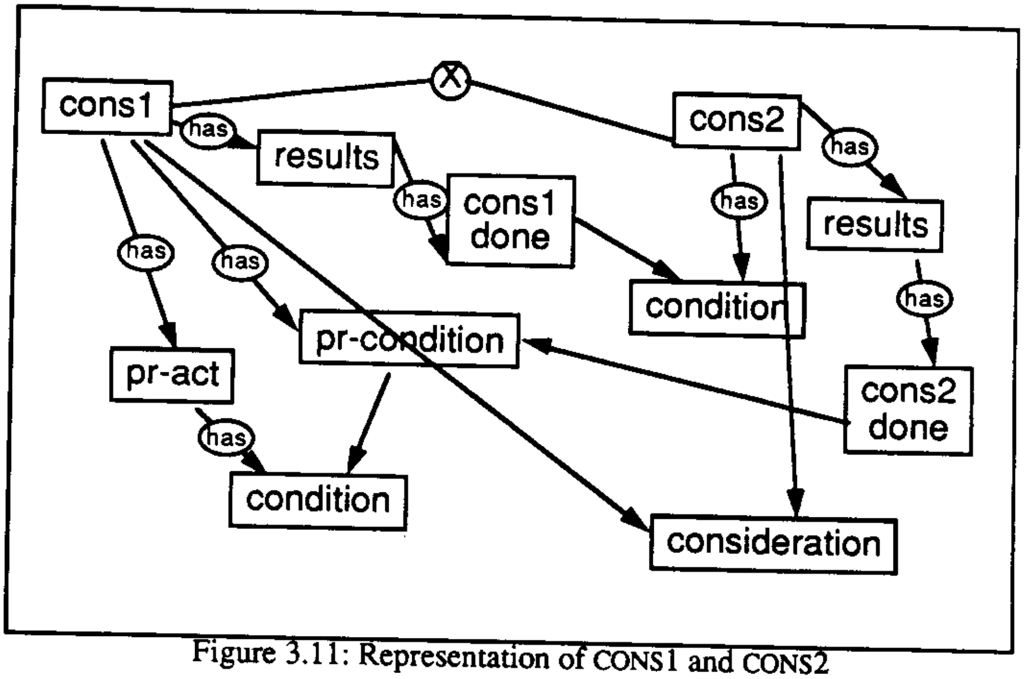


Figure 3.11: Representation of CONS1 and CONS2

The "if" in the above text indicates that party2's action is a condition of party1's promise. This is represented by a GENERALIZATION s-link from CONS2-DONE to CONS1:PR-CONDITION (i.e., the consideration that is a condition of the promise). CONS2-DONE is a role of CONS2:RESULTS representing the fact that the conduct represented by CONS2 has occurred.

What is only implicit in the text is that party2 carries out CONS2 because of party1's promise. This is indicated explicitly in Sherlock's representation by a GENERALIZATION s-link from CONS1-DONE to CONS2:CONDITION.

3.3. REPRESENTING ICON INTERPRETATIONS

The concept of an icon must be represented separately from the concepts which the icon might be used to represent. In Sherlock the interpretation of an icon is represented with an s-link in the semantic network. This is done by making a GENERALIZATION s-link from a concept which is a possible interpretation of an icon to the INTERPRETATION-OF-ICON (IOI) role of the node corresponding to that icon. Figure 3.12 illustrates the relationship between icons, nodes, and concepts.

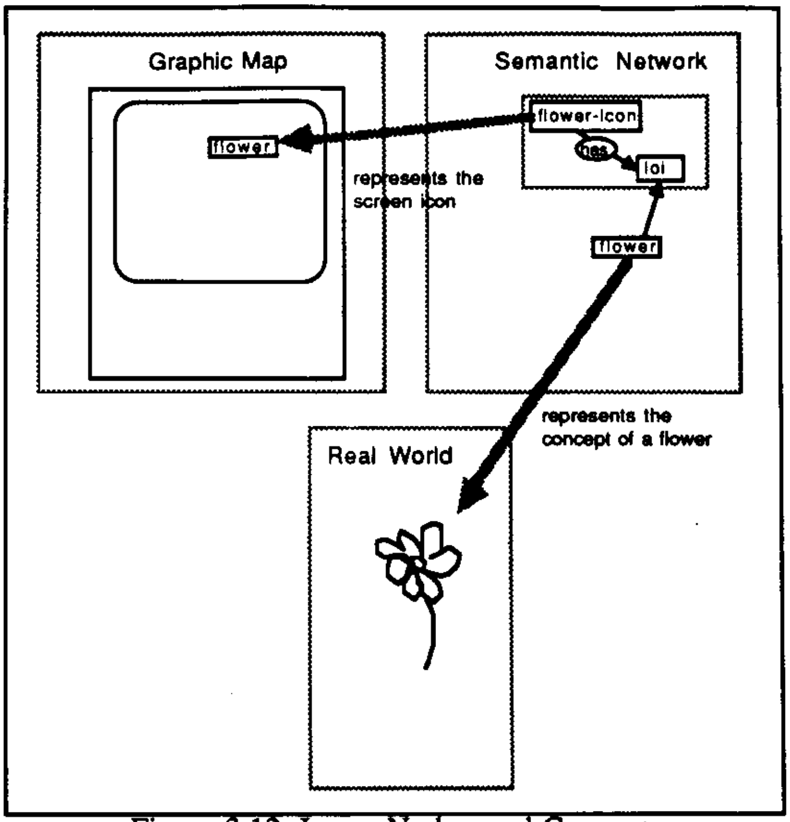


Figure 3.12: Icons, Nodes, and Concepts

Elements in the semantic network represent both the icons on Sherlock's screen and concepts in the real world. In Figure 3.12 FLOWER-ICON represents the icon "Flower" on the screen. The node FLOWER represents the concept of a flower. FLOWER-ICON inherits the role IOI in order to represent all of FLOWER-ICON's possible interpretations. In this case the only interpretation the network knows about is FLOWER. This interpretation is represented by the GENERALIZATION s-link from FLOWER to FLOWER-ICON:IOI.

In this manner all of the icons presented to the learner for making a graphic map of the consideration text are represented in the semantic network. Below are examples of how the consideration text icons are represented.

3.3.1. CONTRACT

As was mentioned in Chapter 2, there are three possible interpretations for the "contract" icon. These interpretations are illustrated in Figure 3.13.

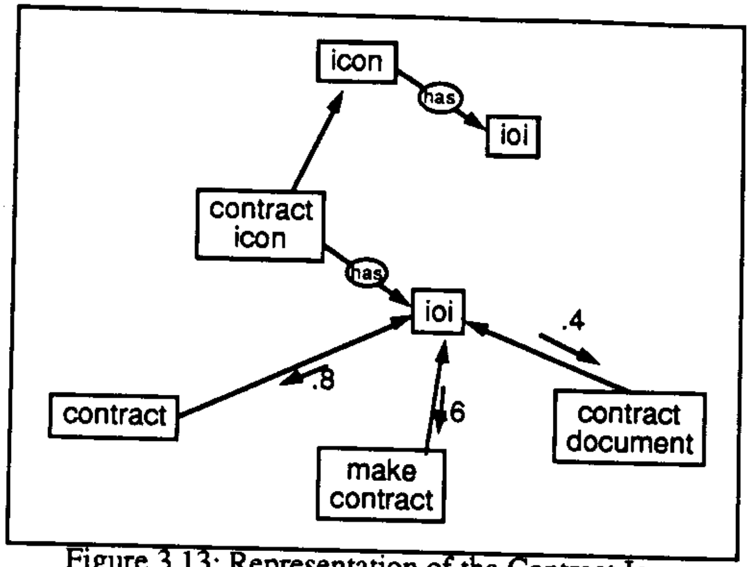


Figure 3.13: Representation of the Contract Icon

CONTRACT represents the concept of a contractual relationship. The weight of .8 on the s-link from CONTRACT-ICON:IOI to CONTRACT indicates that this is the most probable interpretation of "contract." The weights on the s-links which represent interpretations are hand-coded to indicate the frequency of usage of each interpretation during the pilot studies. The next most likely interpretation of "contract" is MAKE-CONTRACT, which represents the act of creating a contractual relationship. The least likely interpretation of "contract" which Sherlock knows about is CONTRACT-DOCUMENT, which represents a document containing the terms of a contractual relationship.

In English, the word "contract" can legitimately be used for any of these meanings. The intended meaning can usually be inferred from the context. For example:

a. "John and Mary entered into a contract" — a contractual relationship;

b. "John showed Mary the contract" — a written document;

c. "John decided to contract with Mary to get the job done" — the act of making a contract.

On the other hand, a particular context can lead to multiple interpretations. For example, the sentence "John violated the contract" can mean that:

a. John destroyed a written document;

b. John set up a contract incorrectly so that it was not valid;

c. John failed to honor the terms of a contractual agreement.

In this context only the third interpretation indicates a correct understanding.

In the graphic mapping task, a learner must interpret an icon with a meaning that is consistent with the context in which he is using that icon. Sherlock must maintain all possible interpretations of an icon, and then figure out both the learner's context and his interpretation.

3.3.2. PROMISE, ACT, AND FORBEARANCE

As was mentioned above, the generic "promise" icon is sometimes used to represent examples of consideration which are promises. This use of the generic icon for a more specialized interpretation also occurs for "act" and "forbearance." Sherlock's representation for these three icons is shown in Figure 3.14. The piece of text in quotes in these figures indicates the actual text writing on the icon being represented.

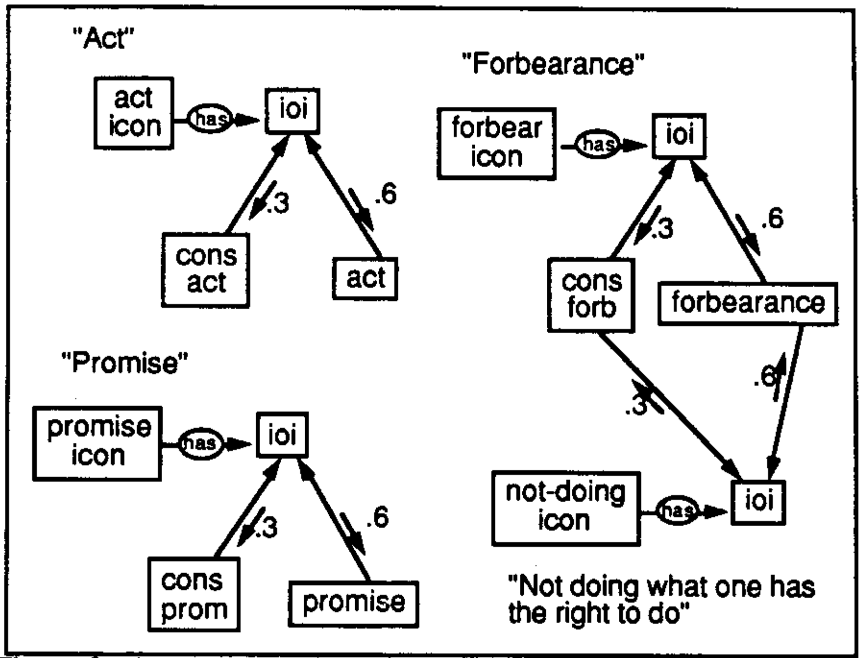


Figure 3.14: Representation of Promise, Act, and Forbearance Icons

In each case, the node representing the more generic (i.e., preferred) interpretation of these icons has the higher weight.

In addition to an icon having multiple possible interpretations, it is possible for a semantic node to be represented in a graphic map by more that one icon. For example, the "Not doing what one has the right to do" icon, represented by the NOT-DOING-ICON, can have the same interpretations as the "forbearance" icon. Multiple icons are provided in this case because the concept FORBEARANCE is referred to in the consideration text as both "forbearance" and "not doing what one has the right to do."

3.3.3. ICONS FOR PARTS OF A CONTRACT

To demonstrate how icons map to concepts, Figure 3.15 illustrates the representation of the icons used to represent parts of a contract.

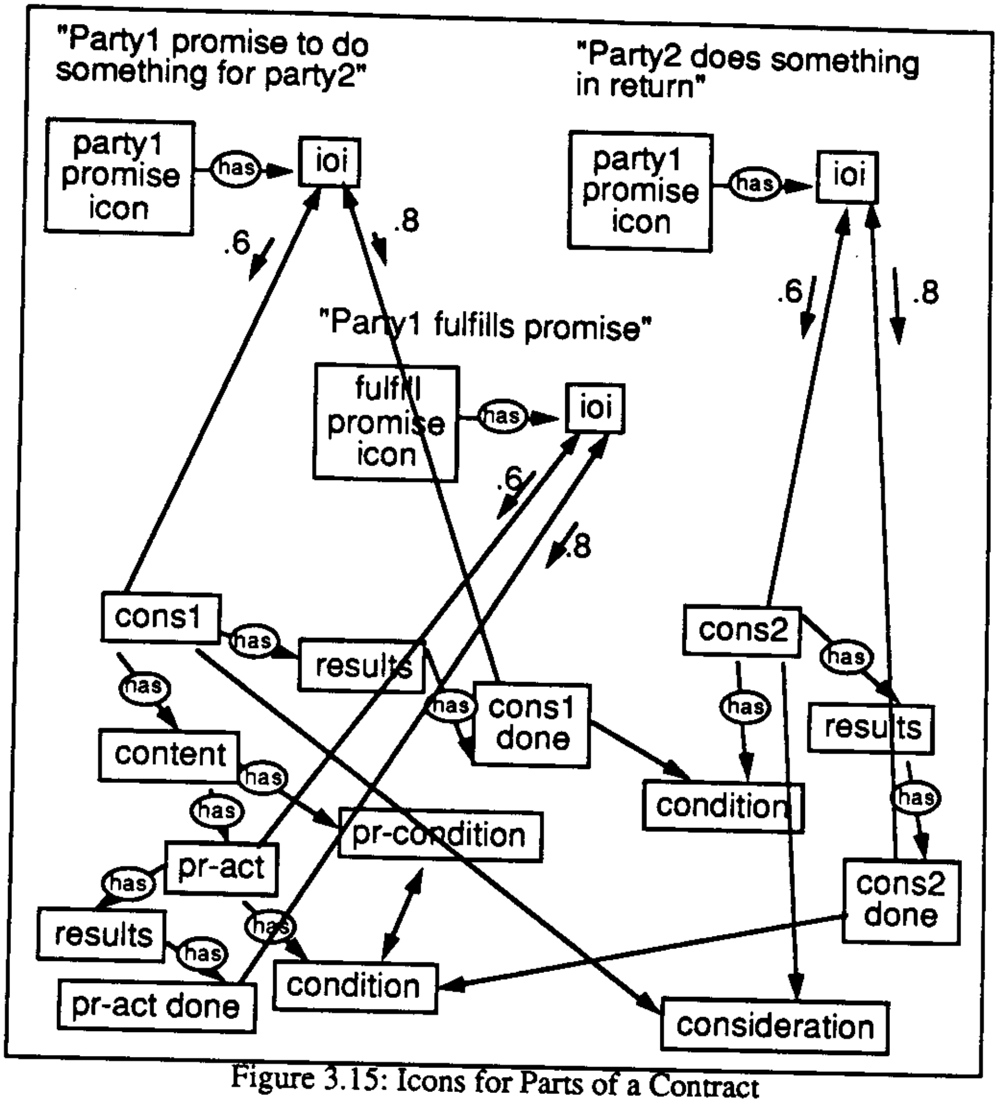


Figure 3.15: Icons for Parts of a Contract

The icon "Party1 promise to do something for party2" has two interpretations: CONS1 and CONS1-DONE which is a role of CONS1:RESULTS. The two interpretations are used because in English usage the distinction between an act, and the fact that the act has occurred, is not always clear. Similarly the icon "Party2 does something in return" may be interpreted as CONS2 or CONS2:RESULTS:CONS2-DONE. The icon "Party1 fulfills promise" may be interpreted as PR-ACT or PR-ACT:RESULTS:PR-ACT-DONE.

3.4. SUMMARY

The semantic network in Sherlock provides a non-ambiguous means of describing knowledge. The network can be used by Sherlock to determine the inferences which are possible given a particular representation. The network consists of nodes which represent concepts connected by s-links which represent relationships between the concepts.

Three types of nodes are used in Sherlock's semantic network:

1. head nodes -- which represent main concepts;
2. role nodes -- which represent attributes of main concepts;
3. inhibition nodes -- which represent the mutually exclusive relationship which exists between a given set of concepts.

Sherlock uses four types of s-links:

1. GENERALIZATION s-links -- which indicates the conceptual classes to which a node belongs;
2. EQUIVALENT s-links -- which indicates pairs of nodes which represent the same concept;
3. HAS s-links -- which link a main concept to one of its role nodes;
4. INHIBITION s-links -- which link concepts to inhibition nodes.

The semantic network is used to represent the content of the consideration text, and the background knowledge necessary to understand the text. In addition the network is used to represent the icons which the learner uses in a graphic map. The network is also used to represent the relationship between these icons and the concepts which they can be used to represent.

Chapter 4

Representing Strategy

4.0. INTRODUCTION

There are three goals in representing strategy in Sherlock:

1. represent the strategy a learner uses to build a graphic map;
2. represent the tutor's knowledge of the strategies learners may use to build graphic maps;
3. represent the tutor's strategy for tutoring a learner.

Strategy knowledge in Sherlock is represented by condition-action rules. The condition portion of a rule consists of facts. The action portion of a rule contains actions which are appropriate if all of the facts in that rule's condition portion are currently true.

It was mentioned in Chapter 1 that a major challenge in learning-strategy research is that successful learners are usually unable to articulate their strategy for learning. They cannot explain their strategy because the knowledge necessary to perform a task is different from the knowledge necessary to explain the strategy for accomplishing the task.

Because the two types of knowledge are different, an expert at a particular task may not make the best teacher for that task. Conversely, the best teacher for a particular task will not necessarily be the best practitioner of that task.

To distinguish between the ability to do and the ability to explain, Sherlock's rules are represented in two distinct representation schemes:

1. The knowledge necessary to perform a task is represented in a production system;
2. Knowledge necessary to explain the strategy is represented in the semantic network.

Each of Sherlock's rules is represented in one or both of these schemes, depending on the use Sherlock must make of that rule.

- If Sherlock needs a rule only to perform a task, it is represented only in the production system. The rules representing Sherlock's rules for tutoring are represented only in the production system since Sherlock must perform the task of tutoring but does not need to explain the task of tutoring.
- If Sherlock needs a rule both to explain a task and to perform the task, the rule is represented in both the semantic network and the production system. Sherlock's rules for building a graphic map are represented in both schemes since Sherlock must use them to decide what is appropriate in a graphic map, and to explain the task of graphic mapping to the learner.

4.1. STRATEGIC KNOWLEDGE

Production systems (Newell, 1967) have been popular both as a notation system for indicating an individual's strategic knowledge and as a cognitive model for human information processing (Neeches, Langley & Klahr, 1987). In representing a portion of an individual's knowledge in a production we are saying;

> If the condition statements of this production were true...
>
> the individual would take the actions indicated by the action portion of this production.

An individual's task performance knowledge is represented by a large number of productions. According to the productions system model, an individual's action at any given moment of time is determined by the facts that are true at that moment, and the productions that individual has that match the current true facts.

A basic production system consists of two interacting data structures operated on by a simple processing cycle. The two data structures are:

1. working memory, consisting of a set of symbolic data items;
2. production memory, consisting of condition-action rules.

4.1.1. WORKING MEMORY

Sherlock's working memory consists of two components:

1. a set of slots and their fillers, which serves as Sherlock's short-term memory;
2. the semantic network, which serves as Sherlock's long-term memory.

4.1.1.1. SHORT-TERM MEMORY

The short-term memory consists of variables such as:

current-goal	;This is the goal that Sherlock is currently attempting to satisfy.
plan-used	;This is the plan that Sherlock believes the learner used to make the current gm-link.
sherlocks-move	;This is the plan that Sherlock would have used to link the two icons which the learner just linked.

This type of working memory is typical of most production systems. These elements are directly accessible to the production system, and most easily modified by the production system. It is straightforward to determine if an antecedent which relates to short-term memory is satisfied.

4.1.1.2. LONG-TERM MEMORY

Allowing the production system to use the semantic network as part of working memory is a more complicated process. This use of the semantic network necessitates capabilities that allow consequents of productions to initiate spreading activation, and antecedents to compare activation patterns. These capabilities are novel features added to Sherlock, not available in Gasser's network package

In order for productions to compare activation patterns, Sherlock first translates activation patterns into facts in short term memory. The facts can then be used directly by the production system.

4.1.2. PRODUCTION MEMORY

Sherlock's production memory consists of rules of the form:

IF <circumstances>

THEN <do action>

Each rule has a strength component, indicating how good the rule is. Rule strength ranges from -1 to +1. Rules with a strength of less than zero are considered buggy.

Sherlock has buggy rules only in the representation of the graphic mapping strategy. These buggy rules are hand coded in Sherlock to represent the faulty reasoning that subjects used in constructing graphic maps in the initial pilot studies. Sherlock can sometimes infer a learner's faulty reasoning, but it has no ability to explain why it is faulty. If Sherlock can match the learner's reasoning against one of the known buggy rules, however, it can use the canned explanation hand coded for that buggy rule.

Unlike Sherlock, human tutors are usually able to decide why incorrect reasoning is incorrect. Similar to Sherlock, however, a human tutor will also have memory of previously encountered faulty reasoning. If a human tutor recognizes faulty reasoning as being an example previously encountered faulty reasoning, she may, as does Sherlock, use an explanation that had been used before.

4.2. KNOWLEDGE ABOUT STRATEGY

Rules which Sherlock must be able to "think about" are implemented as nodes in the semantic network. This implementation enables Sherlock to model thinking about actions as opposed to just deciding what action to take given a set of circumstances. A rule is represented in the network as a node with two roles: ANTECS (antecedents) and CONSEQS (consequents) (Figure 4.1).

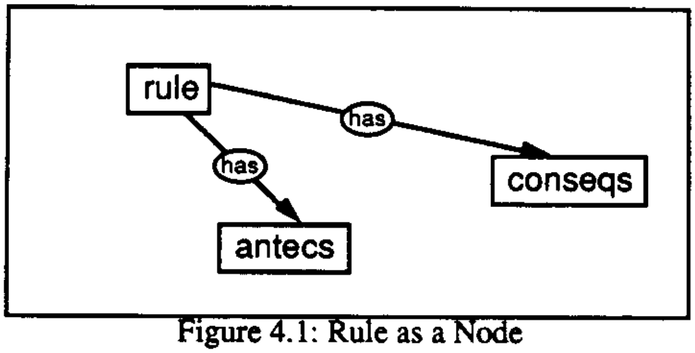


Figure 4.1: Rule as a Node

Each rule Sherlock knows about is a specialization of RULE. The antecedents of each rule are represented as roles of that rule with GENERALIZATION links to the ANTECS role of RULE. Similarly the consequents of each rule are represented as roles of that rule with GENERALIZATION links to the CONSEQS role of RULE.

One of the rules for making an IS-A gm-link is illustrated in Figure 4.2 as it is represented in the semantic network. IS-A-RULE has two roles which are specializations of RULE:ANTECS. These roles are specialization of the facts: X-IS-Y and Y-UNLIK-X. The meaning of these facts will be discussed below.

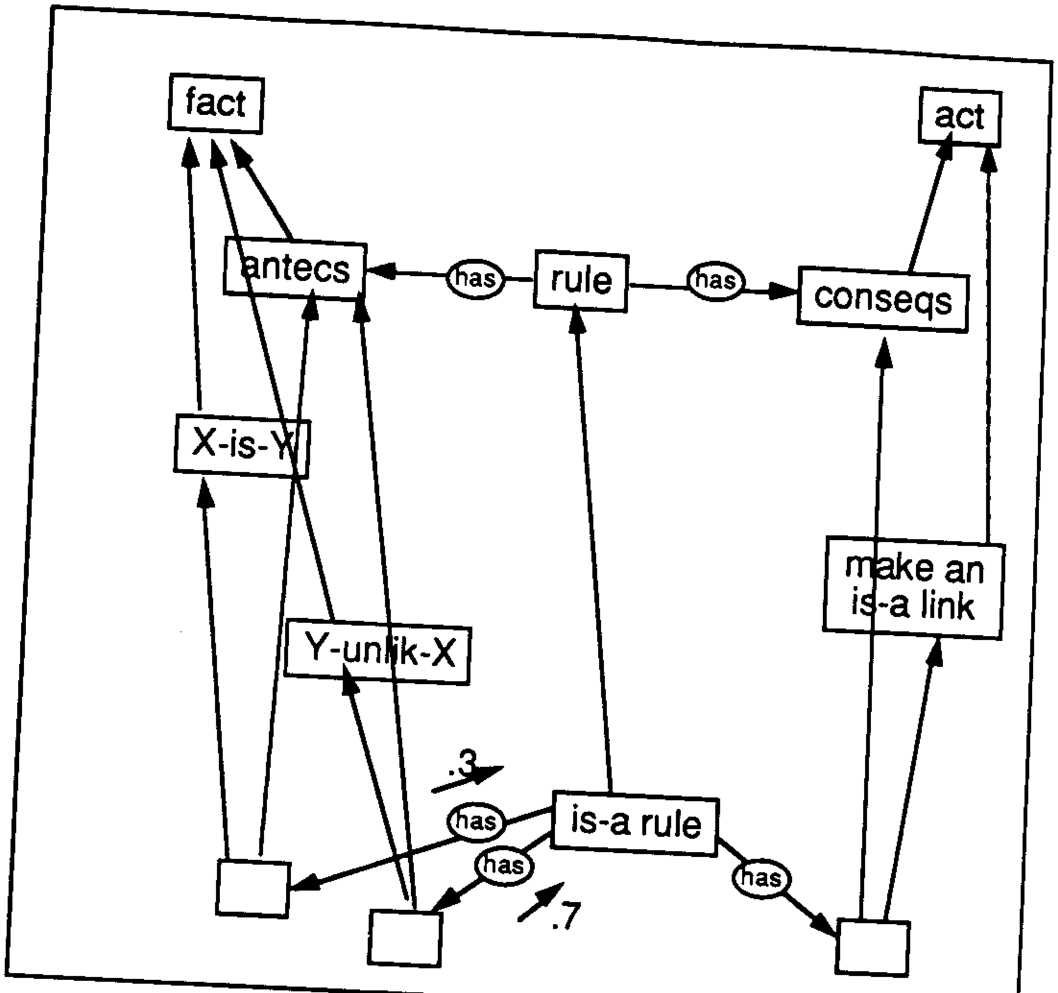


Figure 4.2: IS-A Rule as a Node

IS-A-RULE has one role, which is a specialization of RULE:CONSEQS, which is a specialization of the act of making an IS-A gm-link.

The weights on the s-links going from the roles to IS-A-RULE represent, as do all weights on HAS s-links, how salient a characteristic this role is for the rule. In other words, how important is it that you know that this fact is true, in order to use this rule? This weighting allows Sherlock to classify a rule despite incomplete information about the facts.

These weights are represented only in the semantic network; they are not represented in the production system The production system is not concerned with the relative importance of antecedents since it is intended to model performance of tasks that have become automatic. That is, tasks that do not have to be thought about. Thus the production system uses an all-or-nothing approach to matching productions. A rule either fits the current situation, or it does not. A task is no longer automatic when we stop to think about it. In Sherlock's model when reasoning about the task is required, the rules in the semantic network are used.

4.3. SHERLOCK'S STRATEGY FOR CONSTRUCTING A GRAPHIC MAP

4.3.1. ANTECEDENTS

The rules for making gm-links use thirteen facts as antecedents. These facts are represented as nodes in the semantic network that are descendents of FACT. The thirteen fact nodes, and their meaning are:

1.	X-IS-Y	There is a high probability that any concept which is an X will also be a Y.
2.	Y-IS-X	There is a high probability that any concept which is an Y will also be a X.
3.	X-NOT-Y	There is little probability that any concept which is an X will also be a Y.
4.	Y-NOT-X	There is little probability that any concept which is an Y will also be a X.
5.	X-MAYBE-Y	It is possible that any concept which is an X will also be a Y.
6.	Y-UNLIK-X	The probability that a concept which is a Y is also an X is less than the probability that a concept which is an X is also a Y.
7.	EQ-CH	The probability that Y is an X is equal to the probability that an X is a Y.
8.	X-HAS-Y	There is a high probability that any concept which is an X will have an attribute which is a Y.
9.	X-COND-Y	There is a high probability that any concept which is an X is also a concept which is a CONDITION of Y.
10.	Y-EXPECT-X	There is a high probability that any concept which is Y is also a concept which is an EXPECTATION of X.
11.	Y-RESULT-X	There is a high probability that any concept which is Y is also a concept which is an RESULT of X.
12.	SAME-CONC	X and Y are the same concept.

13.	X-PART	It is possible that any concept which is an X will also be a PART.

4.3.2. RULES

The following are rules for constructing a graphic map. All of these rules are illustrated as explicit rules although they are actually represented in both the production system and the semantic network.

Each of these rules has an additional antecedent which is not listed. This antecedent states that the current goal is to make a gm-link between icon-X and icon-Y. X and Y in the antecedents refer to the current interpretations of icon-X and icon-Y.

4.3.2.1. RULES FOR MAKING AN IS-A GM-LINK

Figure 4.3 contains Sherlock's rules for making an IS-A gm-link.

```
is-a-rule
    IF      X-is-Y
    and     Y-unlik-X
    THEN    make an IS-A gm-link from X to Y

weak-is-a-rule
    IF      X-maybe-Y
    and     Y-unlik-X
    THEN    make an IS-A gm-link from X to Y
```

Figure 4.3: Rules for Making an IS-A Gm-Link

IS-A-RULE represents Sherlock's basic rule for making an IS-A gm-link. An IS-A gm-link is appropriate when X is a member of Y's conceptual class. The best that spreading activation can do is to indicate the probability that X is associated with Y. If this probability is high, there is a good chance that X is a member of Y's conceptual class. The probability that an X is a Y, however, will also be high if the two concepts are equivalent, or if Y is an important member of X's conceptual class.

By also looking at the probability that a Y is an X, Sherlock can determine which of these three possibilities is most likely:

1. If it is less likely that a Y is an X, then X is probably a member of Y's conceptual class;
2. If it is just as likely that a Y is an X, then X and Y are probably equivalent;

3. If is is more likely that a Y is an X, then Y is probably a member of X's conceptual class.

Thus the second antecedent is needed to ascertain that the first antecedent is due to the fact that X is a member of Y's conceptual class.

The WEAK-IS-A-RULE is a non-optimal version of the IS-A-RULE. This rule indicates that X might be a member of Y's conceptual class even if the probability that an X is a Y is low, but existent. It is even more important, however, for Sherlock to know that this probability is still higher than the probability that a Y is an X.

Figure 4.4 contains buggy rules for making an IS-A gm-link.

buggy-rev-is-a-rule

IF X-is-Y
and Y-unlik-X
THEN make an IS-A gm-link from Y to X

buggy-half-is-a-rule

IF X-maybe-Y
THEN make an IS-A gm-link from X to Y

Figure 4.4: Buggy Rules for Making an IS-A Gm-Link

Use of the BUGGY-REV-IS-A-RULE would indicate that the learner made an IS-A gm-link for the right reasons, but made the link in the wrong direction. The BUGGY-HALF-IS-A-RULE is a buggy version of IS-A-RULE because it lacks the second antecedent. Use of this rule might indicate that the learner is not considering the possibilities that X is equivalent to Y, or that Y is an X.

4.3.2.2. RULE FOR MAKING A NOT GM-LINK

Sherlock knows only one rule for making a NOT gm-link (Figure 4.5).

not-rule

IF X-not-Y
and Y-not-X
THEN make a NOT gm-link from Y to X

Figure 4.5: Rule for Making a NOT Gm-Link

The NOT-RULE simply states that if there is no chance that an X is a Y, and no chance that a Y is an X, then a NOT gm-link will be appropriate between X and Y.

4.3.2.3. RULES FOR MAKING AN EQUIV GM-LINK

Figure 4.6 contains Sherlock's rules for making an EQUIV gm-link.

equal-high-rule

IF X-is-Y
and eq-ch
THEN make an EQUIV gm-link from Y to X

same-rule

IF same-conc
THEN make an EQUIV gm-link from X to Y

Figure 4.6: Rules for Making an EQUIV Gm-Link

It was mentioned earlier that when X-is-Y is true, there are three possibilities. The EQUAL-HIGH-RULE covers the second possibility. The second antecedent states that if it is also known that the probability that a Y is an X is equal to the probability that an X is a Y, then an EQUIV gm-link is appropriate.

SAME-RULE states that when X and Y are the same concept an EQUIV gm-link is also appropriate. This situation occurs when two icons can be interpreted as referring to the same concept node in the semantic network.

Figure 4.7 contains buggy rules for making an EQUIV gm-link.

```
buggy-dub-is-a-rule
    IF      X-is-Y
    and     Y-is-X
    THEN    make an EQUIV gm-link from Y to X

buggy-equal-rule
    IF      eq-ch
    THEN    make an EQUIV gm-link from X to Y
```

Figure 4.7: Buggy Rules for Making an EQUIV Gm-Link

The BUGGY-DUB-IS-A-RULE says that if there is a high probability that an X is a Y and a high probability that a Y is an X then an EQUIV gm-link would be appropriate. The problem with this rule is that even if the probability is high in both directions, if the probabilities are not equal, one of the concepts is more likely to be a member of the other's conceptual class, rather than being equivalent. If the probabilities were both high and equal, the EQUAL-HIGH-RULE above would have been used.

The problem with the BUGGY-EQUAL-RULE is the inverse of the problem with the BUGGY-DUB-IS-A-RULE. According to this rule, if the probability that an X is a Y is equal to the probability that a Y is a X then an EQUIV gm-link is appropriate. If two concepts are totally unrelated, however, the probabilities will both be zero, and thus equal. An EQUIV link would obviously be inappropriate between to unrelated concepts.

4.3.2.4. RULES FOR MAKING GM-LINKS FOR ATTRIBUTES

There are two kinds of gm-links for attributes: PART and PROP. Figure 4.8 contains the rules for making these gm-links.

part-rule

IF Y-has-X
and X-part
THEN make a PART gm-link from Y to X

maybe-part-rule

IF Y-has-X
THEN make a PART gm-link from X to Y

prop-rule

IF Y-has-X
THEN make a PROP gm-link from X to Y

Figure 4.8 Rules for Making GM-Links for Attributes

The distinction between a part-of relationship and a property-of relationship is sometimes unclear. Learners are given the following guidelines to help choose between these two gm-links:

1.	PART	If one concept is an identifiable physical part of a second concept, then a PART link is more appropriate
2.	PROP	If one concept is an attribute which characterizes a second concept, then a PROP link is more appropriate.

The PART-RULE covers the first guideline. This rule says that if X is probably an attribute of Y and X is probably a part then a PART gm-link is appropriate. The PROP-RULE is intended to cover the second guideline. This rule says that if X is probably an attribute of Y, but there is no evidence that X is a part, then a PROP gm-link is appropriate.

There are situations, however, where the distinction between a part relationship and a property relationship is blurred. For example, what is the relationship between "John throws the ball" and John? Is John a part of the act, or is John a property of the act? Since there is no actor gm-link, the learner must choose the attribute gm-link which seems most appropriate. The MAYBE-PART-RULE means that Sherlock will consider either a PROP gm-link or a PART gm-link as appropriate when there is no evidence that the attribute is a part of something.

4.3.2.5. RULES FOR MAKING LEADS GM-LINKS

Figure 4.9 contains Sherlock's rules for making a LEADS gm-link.

leads-rule

IF X-cond-Y
THEN make a LEADS gm-link from X to Y

result-rule

IF Y-result-X
THEN make a LEADS gm-link from X to Y

expect-rule

IF Y-expect-X
THEN make a LEADS gm-link from X to Y

Figure 4.9: Rules for Making a LEADS Gm-Link

Acts in Sherlock have roles for conditions and expectations. The LEADS-RULE says that if a fact is a condition of an act, then it is appropriate to make a LEADS link from the fact to the act. Similarly, the RESULT-RULE says that if a fact is a result of an act, then it is appropriate to make a LEADS link from the act to the fact. The EXPECTATIONS role of THING is used to represent facts which would be true, if a particular type of THING exists. Thus if a fact is an expectation of a thing, the EXPECT-RULE indicates that it is appropriate to make a LEADS link from the thing to the fact.

There is one buggy rule for a LEADS link which Sherlock knows about (Figure 4.10).

buggy-prop-leads-rule

IF X-has-Y
THEN make a LEADS gm-link from X to Y

Figure 4.10: Buggy Rule for a LEADS Gm-link

The PART-LEADS-RULE has the same antecedent as MAYBE-PART-RULE and PROP-RULE, but results in a LEADS gm-link. This particular rule is included in Sherlock's knowledge because it represents a common misuse of the LEADS link.

4.4. SHERLOCK'S STRATEGY FOR TUTORING

4.4.1. ANTECEDENTS

Below are the short-term memory elements which are used by the tutoring strategy rules:

from-icon	the icon at the beginning end of the current gm-link (also *to-icon*);
from-icon-children	list of possible interpretations for *from-icon* which have not already been tried (also *to-icon-children*);
f	The "from" concept, that is, Sherlock's current interpretation of *from-icon* (also *t* for the "to" concept);
goal	list of goals which Sherlock is currently working on;
eq-goal?	returns true if its argument is the first goal on the list;
sherlocks-move	the rule Sherlock would have used to link the current icons;
production-link	access function which returns the type of gm-link which a rule would have led to;
link-made	the type of gm-link which the learner made;
plan-used	the rule which Sherlock has determined that the learner used to make the current gm-link;
plan-used-status	Sherlock's evaluation of the appropriateness of the rule which Sherlock believes the learner just used;
new-interp	Sherlock's current interpretation of the icons which were linked;
possible-prods	list of rules for making gm-links which have antecedents which are currently true;
why-wrong-learners-plan	list of wrong facts which the learner believes, and which led to the inappropriate rule which the learner used.

4.4.2. RULES

The following are examples of the rules Sherlock uses for tutoring. These rules are represented only in Sherlock's production system since Sherlock has no need to reason about its own tutoring strategies. The rules will be illustrated using the code that creates the rule in the production system.

Figure 4.11 contains the Lisp code that implements one of the rules Sherlock uses to decide initially if the learner's gm-link makes sense.

```
(tproduction!
 'check-best
 '(                            ;** ANTECEDENTS **
   (eq-goal? 'eval-link)       ;the current goal is to
                               ;  evaluate the link
   *sherlocks-move*            ;sherlock would have
                               ;  linked these icons
   (eq? (production-link       ;sherlock would have
          *sherlocks-move*)    ;  made the same link
          *link-made*)         ;  the learner made
   )
 '(                            ;** CONSEQUENTS **
   (set *plan-used*            ;conclude that the
          *sherlocks-move*)    ;  learner used the
                               ;  same rule sherlock
                               ;  would have used
   (set *plan-used-status*     ;conclude that the
           'ok)                ;  learner's rule is OK
   (pop *goal*)                ;remove current goal
                               ;  because it has been
                               ;  it has been
                               ;  satisfied
   (goal! 'change-net)         ;add goal to consider
   )                           ;  changing the network
 .9)                           ;this rule has a high
                               ;  strength so that it
                               ;  will always be used
                               ;  if it is true
```

Figure 4.11: Check-best Rule

This rule is appropriate if the goal is to evaluate a gm-link, and Sherlock has determined the rule it would have used to make a gm-link between the learner's two icons (*sherlocks-move*), and that rule would have led to the same link that the learner made (*link-made*). In this case Sherlock would assume that the plan that the learner used (*plan-used*) is the same as Sherlock's plan, and thus that the learner's plan is acceptable. Sherlock would stop trying to evaluate the link at this point and would check and see if the use of this plan implies any need to make a change in the semantic network.

Figure 4.12 contains an example of a rule that alters the semantic network following the evaluation of a gm-link.

```
(tproduction!
 'assign-concept-role
 '(                                   ;** ANTECEDENTS **
   (eq-goal? 'change-net)             ;the current goal is to
                                      ;  determine if the
                                      ;  network should be
                                      ;  changed
   (eq? *link-made* 'part)            ;the learner made a
                                      ;  PART gm-link
   (eq? *plan-used-status*            ;the rule used was
        'ok)                          ;  appropriate for
                                      ;  these icons
   (not                               ;the role which is the
   (eq? f-child-role-t *f*))          ;  part is not the from
   )                                  ;  concept

 '(                                   ;** CONSEQUENTS **
   (assign-concept                    ;make an s-link from
     f-child-role-t                   ;  the role which is
     *from-icon*)                     ;  the part to the from
                                      ;  icon's ioi role
   (pop *goal*)                       ;remove current goal
                                      ;  because it has been
                                      ;  been satisfied
   (goal! 'get-new-move)              ;set a new goal to wait
   )                                  ;  for the learner's
                                      ;  next move
.9)
```

Figure 4.12: Assign Concept Role

When Sherlock is done evaluating a gm-link, the goal is set to change the network. Setting the goal just means that Sherlock will check to see if a change in the network is appropriate given the link which the learner just made. This rule only applies if a PART gm-link is made, and if Sherlock has decided that the link was appropriate.

Whenever a PART link is appropriate, the concept associated with the from-icon is a child of some role, or is itself a role, of the to-concept. If the from-concept was not itself the role, then this rule says to make an explicit *f* from the concept which was the role to the from-icon's IOI role.

Figure 4.13 contains an example of the rules which check alternate interpretations for the icons which the learner linked.

```
(tproduction!
 'new-from-icon
 '(                             ;** ANTECEDENTS **
   (eq-goal? 'eval-link)        ;current goal is to
                                ;  evaluate the link
   (<(accuracy *new-interp*)    ;using the present
      .8)                       ;  interpretations the
                                ;  fact accuracy was
                                ;  less than 80%
   (> (length                   ;there is at least one
    *from-icon-children*) 1)    ;  more interpretation
   )                            ;  for the from icon
 '(                             ;** CONSEQUENTS **
   (store-interp)               ;remember this interp
   (set *possible-prods*        ;forget the current
       nil)                     ;  analysis
   (goal! 'eval-link)           ;set a new goal to
                                ;  to evaluate the link
   (pop                         ;remove the current
     *from-icon-children*)      ;  interpretation for
                                ;  the from icon, so
                                ;  that the next one
                                ;  will now be used
   )
 .4)
```

Figure 4.13: New-From-Icon Rule

This rule is appropriate whenever evaluating a gm-link if it has been determined that agreement between Sherlock's facts and the learner's facts is less than 80% and there is still at least one interpretation for the from-icon which has not been considered. If the rule is used Sherlock will temporarily set aside the interpretation it has been considering and try evaluating the gm-link from scratch, this time using the next possible interpretation for the from-icon.

Figure 4.14 contains one of the rules for providing tutoring when text understanding is diagnosed as part of the problem.

```
(tproduction!
 'wrong-facts
 '(
                                     ;** ANTECEDENTS **
   (eq-goal? 'tutor-text)            ;the current goal is to
                                     ;  provide any
                                     ;  appropriate tutoring
                                     ;  for the text
   *plan-used*                       ;sherlock found the
                                     ;  rule which the
                                     ;  learner used
   (> (length                        ;there is at least one
  *why-wrong-learners-plan*)         ;  wrong fact which led
      0 )                            ;  the learner to use
      )                              ;  this rule


 '(
                                     ;** CONSEQUENTS **
   (output-msg
'(c "You seem to
     believe that:" t))
   (explain-fact                     ;generate the wrong
  *why-wrong-learners-plan*)         ;  facts in English
   (output-msg
      '(t "I do not agree." ))
   (run-menu 'ok-menu)
   (hide-menu 'ok-menu)
  )
.7
)
```

Figure 4.14: Wrong Facts

This rule assumes that the goal has already been set to provide tutoring on text understanding. The other two antecedents state that Sherlock has been able to determine the plan that the learner used and that there is at least one false fact which the learner believes, that led to this plan.

4.5. SUMMARY

Strategy rules in Sherlock are represented in two schemes: in a production system and in the semantic network. The production system represents the knowledge needed to actually carry out a strategy. It is intended to be efficient, but does not allow reasoning about the strategy.

Rules encoded within the semantic network represent the knowledge about a strategy. It is intended to allow reasoning by Sherlock about the strategy, for example, inferring the reasons for actions. It would be cumbersome, however, for actually directing action.

Strategy for constructing a graphic map is represented by rules which indicate an appropriate gm-link given a set of true facts about the relationship between two concepts. The rules considered to be good are represented in the production system. These rules allow Sherlock to determine the type of link it would have made between the two icons which the learner linked. The good and buggy rules are represented in the semantic network allowing Sherlock to determine which rule a learner used.

Sherlock's strategy for tutoring is represented by rules that indicate appropriate actions or conclusions given a set of true facts regarding conclusions about the learner's knowledge. These rules are represented only in the production system allowing the production system to direct Sherlock's processing.

Chapter 5

Spreading Activation

5.0. INTRODUCTION

Sherlock uses a form of spreading activation originally designed and implemented by Gasser (1988) and used primarily for classification, determining the appropriate conceptual classes for an instance or sub-type. But the same process also allows the network to determine the relationship between two concepts.

Spreading activation accomplishes two functions in Sherlock:

1. The classification capability is used to classify the plan that a learner used to link two icons as being an instance of one of the plans represented in the network.
2. The relationship-finding capability is used to determine the relationship between two icons, to decide what type of gm-link Sherlock would have made between the two icons.

Spreading activation is used to model these functions for two reasons:

1. It is an accepted mechanism for modelling cognition (Anderson, 1983; Gasser, 1988; Sumida, Dyer & Flowers, 1988; Lange & Dyer, 1988);
2. It provides a simple mechanism for determining whether (and how) two concepts are related.

The simplest means of finding the relationship between two concepts in a semantic network is a blind search through all possible paths. But a blind search is efficient only if all possible paths can be searched simultaneously. Spreading activation in Sherlock accomplishes the task of searching all possible paths in parallel. Spreading activation provides a relatively efficient parallel search by controlling the search at the node level. Each node controls the spread of activation which reaches it. Each node has enough information to determine what it should do when it is activated.

Because activation is controlled locally, some mechanism must be provided for attaching any global meaning to activation. The mechanism used in Gasser's network package is a marker that is assigned to each initial activation. The marker spreads with the activation, providing a pointer back to the node that was originally activated. Using markers, links can represent different kinds of relationships by allowing a link to change an activation's marker, as the activation passes through the link. Just as the strength of the activation reaching from one node to a second node indicates the strength of the association between those nodes, the marker reaching the second node indicates the nature of that association.

5.1. ACTIVATION

Each node can have numerous activation levels, one for each different marker that has reached it. Each activation level ranges from 0 to 100. Activation spreads in discrete time steps. At each time step, activation spreads from any newly activated node through each of that node's outgoing connections. The only exception is that activation does not spread back through the link it just came from. The strength of the activation is equal to the original activation multiplied by the weight of the connection through which it flowed.

When activation reaches a node it will do one of two things:

1. If the new activation has a marker that has already reached that node, the new activation will be added to the old activation level associated with that marker, up to the maximum of 100.
2. If the new activation has a marker that has **not** already reached that node, a new activation level will be created, associated with the marker of the new activation, and the strength of the activation reaching the node.

As was indicated in Chapter 3, the weight of a link will be between 0 and 1. Since the activation is multiplied by the weight of each link it passes through, and each weight is less than one, the strength of the activation has a tendency to decay as it spreads. This tendency to decay insures that activation will not spread indefinitely. The weight of a link also helps to helps to establish the degree of associativity between concepts. Since all nodes are part of the same network, it is always possible to find a path from one node to any other node. But if the path is too long and too indirect the activation will disappear before it reaches the second node. The shorter the path, and the stronger the links along a path, the stronger the association between the concepts.

To demonstrate how activation spreads we will begin with a simple case. In Figure 5.1 the node VERTEBRATE receives an initial activation of 100 associated with marker A (indicated by "A100" above the node VERTEBRATE in the figure). The choice of marker designation (i.e., A) is arbitrary.

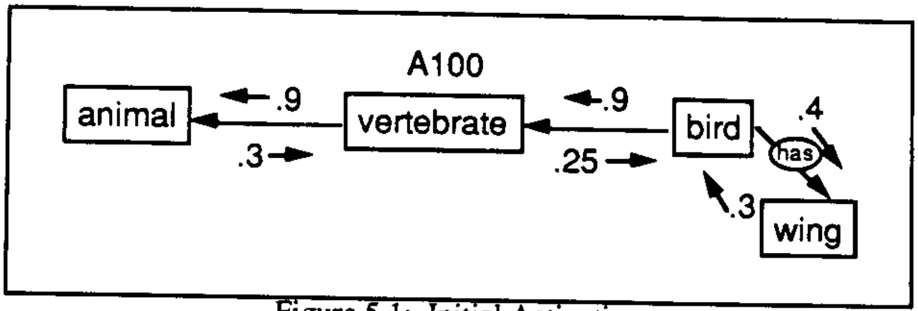


Figure 5.1: Initial Activation

The numbers above and below the links indicate the strength of the connections in each direction[1].

During the first time step, the activation spreads to ANIMAL and BIRD, the two nodes linked to VERTEBRATE. ANIMAL now has an activation of 90 (.9 * 100) and BIRD has an activation of 25 (.25 * 100) as illustrated in Figure 5.2. Notice that activation has not reached WING yet because it is two links away from VERTEBRATE.

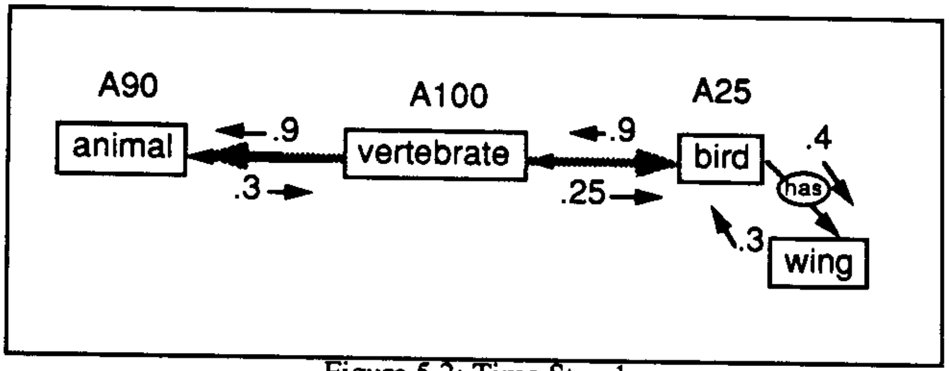


Figure 5.2: Time Step 1

5.2. MARKERS

The activations in Figure 5.2 maintained the same marker (A). Normally we would not know what kind of a path connected VERTEBRATE to BIRD. But we know by looking at the marker for the activation on BIRD, A25, that the activation originated at VERTEBRATE, and that it is associated with VERTEBRATE itself, and not some attribute of VERTEBRATE.

---

[1] The strengths indicated are for demonstration purposes only.

When activation crosses a HAS link the marker changes. If it crosses from a node to one of the node's roles, a new marker (in this case B) is assigned as in time step 2, illustrated in Figure 5.3. If it crosses from a role to the role's owner it is changed back to the marker of the original owner, if it that marker is known. Thus we know that if activation reaches a node with a marker that is different from its source, that it is associated with a role of the source, rather than an ancestor or descendent of the source. Here, activation reaches WING with strength of 10 and the new marker B. The marker B points to WING as its origin and to VERTEBRATE as its original owner.

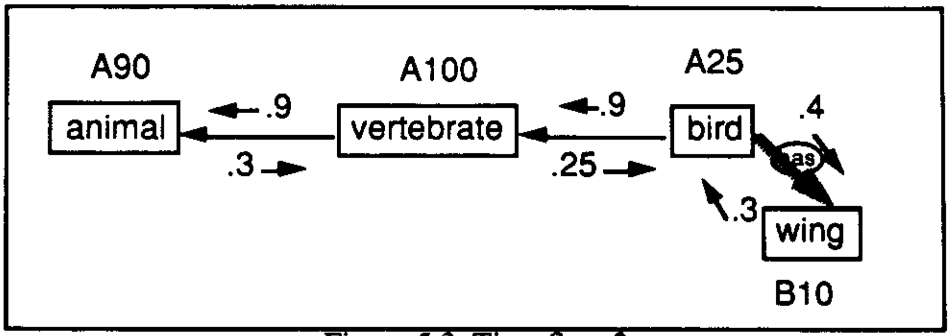


Figure 5.3: Time Step 2

Crossing a role link is the only way that a marker can change. Thus we do not have to know the actual path that exists between two nodes. By looking at the activation on a node we can judge the association that exists between that node and the node that received initial activation. As indicated above, the strength of the activation corresponds to the strength of the association. If it has the same marker as the source, we know it is related to the concept itself. If it has a different marker, we know that it is related to some attribute of the original concept. We can even compare the marker to the markers on the roles of the original concept to determine which attribute it is related to.

This use of markers and activation strengths to establish the relationship between concepts is demonstrated in section 5.5.

5.3. INHIBITION

When activation reaches an inhibition node, a more complicated process takes place. An inhibition node represents an exclusive relationship. It indicates that if a concept is a descendent of one of the nodes attached to the inhibition node, it cannot be a descendent of any of the other nodes attached to the same inhibition node. Thus, any evidence that a concept is a member of one conceptual class can be interpreted as evidence that it is less likely to be a member of any of the other conceptual classes.

To take advantage of this evidence, the activations with the same label on each of the nodes connected to an inhibition node are compared and adjusted. Basically the strongest activations are increased, and the weaker ones are dampened. In Figure 5.4 the nodes attached to the inhibition node have the activations indicated at time step *n*. (We are not concerned here with how these nodes received their present activations and, thus, the original source and the nodes and links which would have spread the activation to these nodes are not illustrated.) During the next time step the inhibition node will adjust the activations to the values shown in time step *n+1*. This process will be repeated each succeeding time step, taking into account any new activation which reaches any of these nodes.

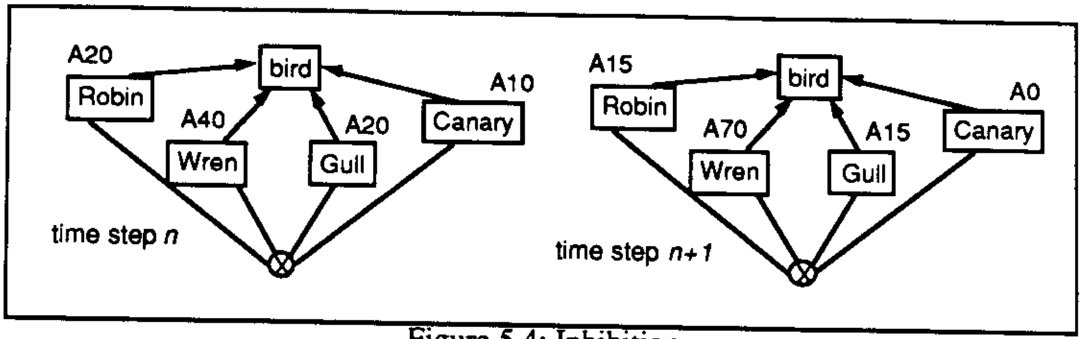


Figure 5.4: Inhibition

5.4. CLASSIFICATION

To illustrate how these mechanisms for spreading activation can be used to classify a concept, consider the simplified knowledge pictured in Figure 5.5.

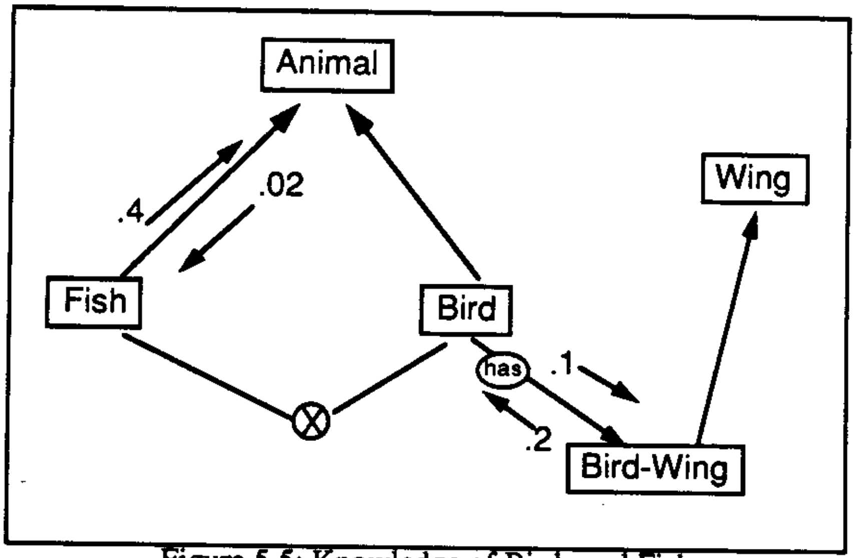


Figure 5.5: Knowledge of Birds and Fish

According to this, all the network knows about are two types of animals: birds and fish. And the only thing known about these is that a bird has a type of wing.

Now assume that the network encounters something that it knows is an animal, and has a wing (Figure 5.6).

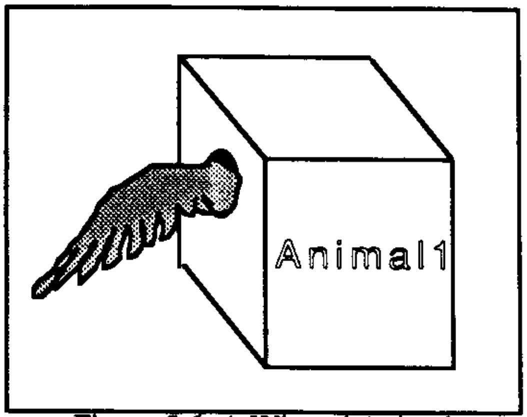


Figure 5.6: A Winged Animal

This instance is represented as ANIMAL1 in Figure 5.7.

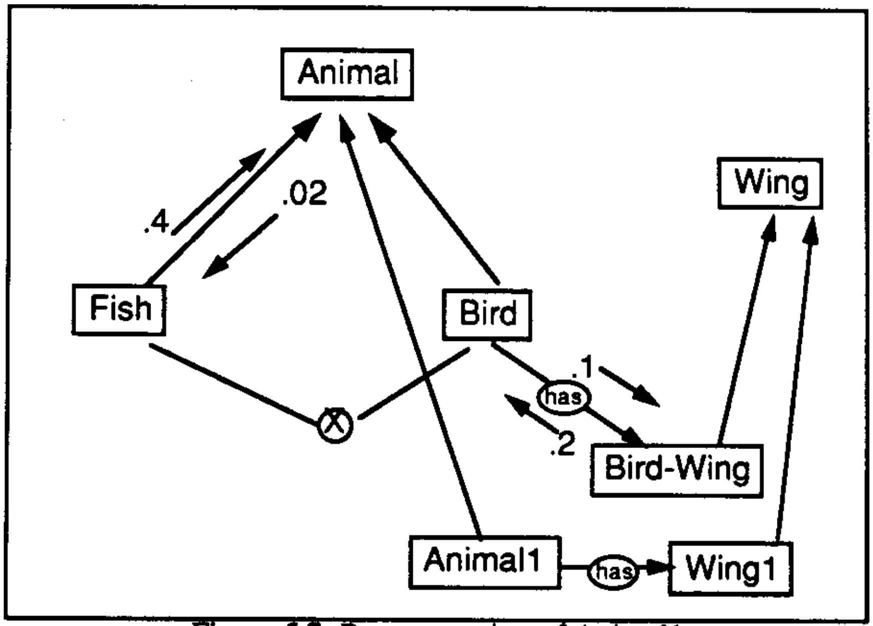


Figure 5.7: Representation of Animal1

To classify this instance we place an initial activation on ANIMAL1 (A100) and its role WING1 (B100) as shown in Figure 5.8. The marker A points to ANIMAL1. The marker B points to WING1 as its origin and to ANIMAL1 as its original owner, since it is being activated as a role of ANIMAL1.

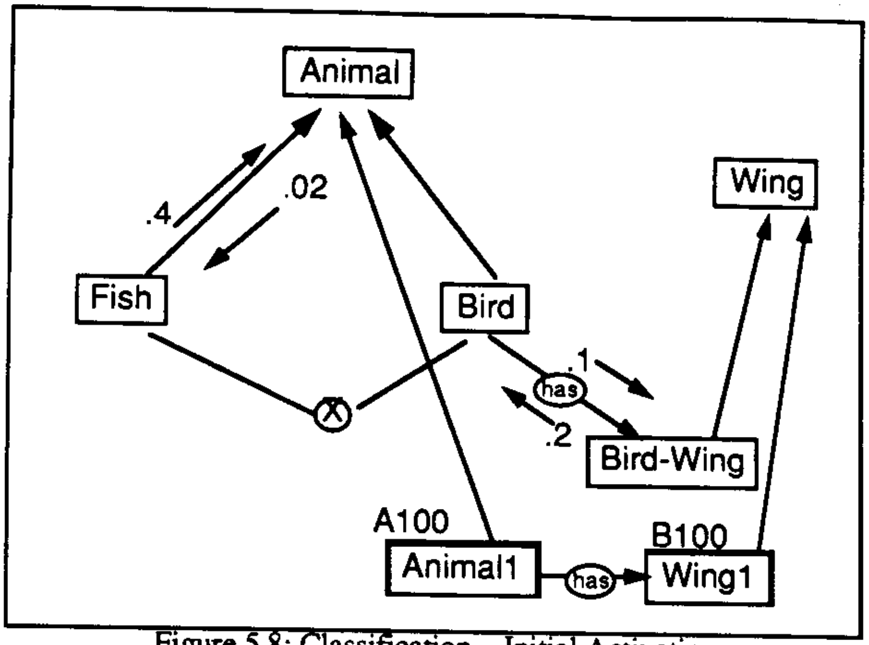


Figure 5.8: Classification—Initial Activation

During the first time step, the activation from ANIMAL1 reaches ANIMAL with a strength of 100 since we know it is an animal. Similarly the activation from WING1 spreads to WING with a strength of B100 (Figure 5.9).

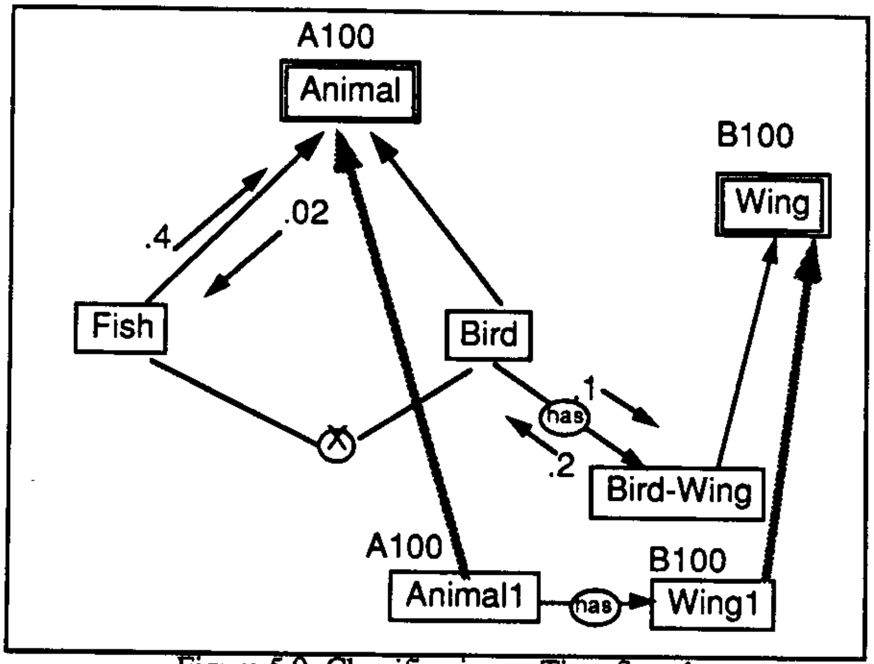


Figure 5.9: Classification —Time Step 1

In Figure 5.10 the activation spreads from ANIMAL down to FISH and BIRD with smaller strengths (A2). Since the activations on FISH and BIRD are equal, the inhibition node they are attached to would not change their activations at this point.

Meanwhile the activation from WING goes down to BIRD-WING with a small activation (B10).

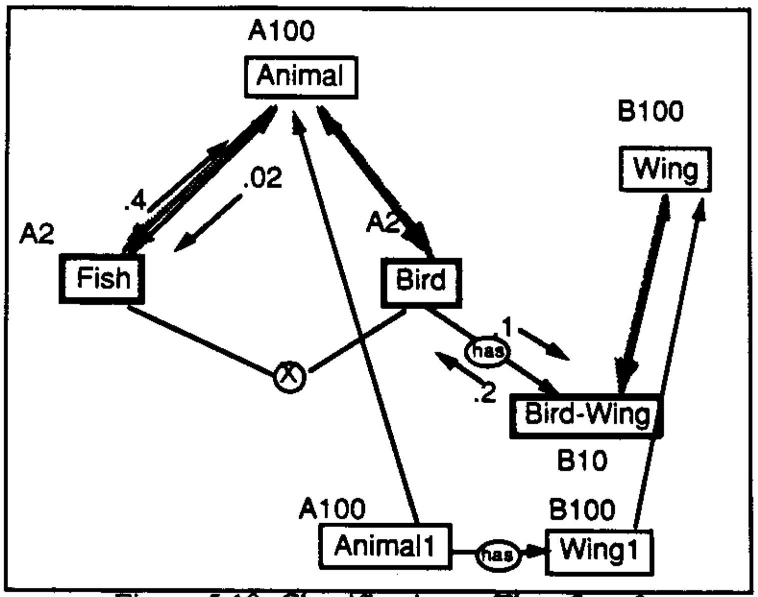


Figure 5.10: Classification —Time Step 2

In Figure 5.11 the activation crosses from BIRD-WING to BIRD.

Since the activation is crossing a role link, the marker changes. Because it is crossing from a role to an owner, the activation tries to change the marker to that of its original owner. It was indicated above that the marker B points to ANIMAL1 as its original owner. Therefore the marker changes to A, the marker associated with ANIMAL1. Thus the activation reaching BIRD is A2 bringing BIRD's activation to A4.

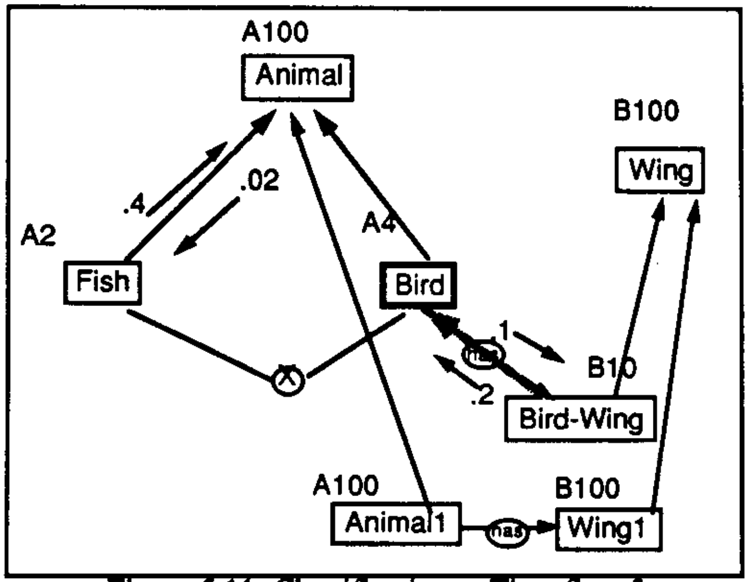


Figure 5.11: Classification —Time Step 3

Since BIRD now has a higher activation that FISH, the inhibition node begins to dampen the activation on FISH and increase the activation on BIRD. In a few time steps the network will settle on BIRD by giving BIRD the maximum activation of 100 (Figure 5.12).

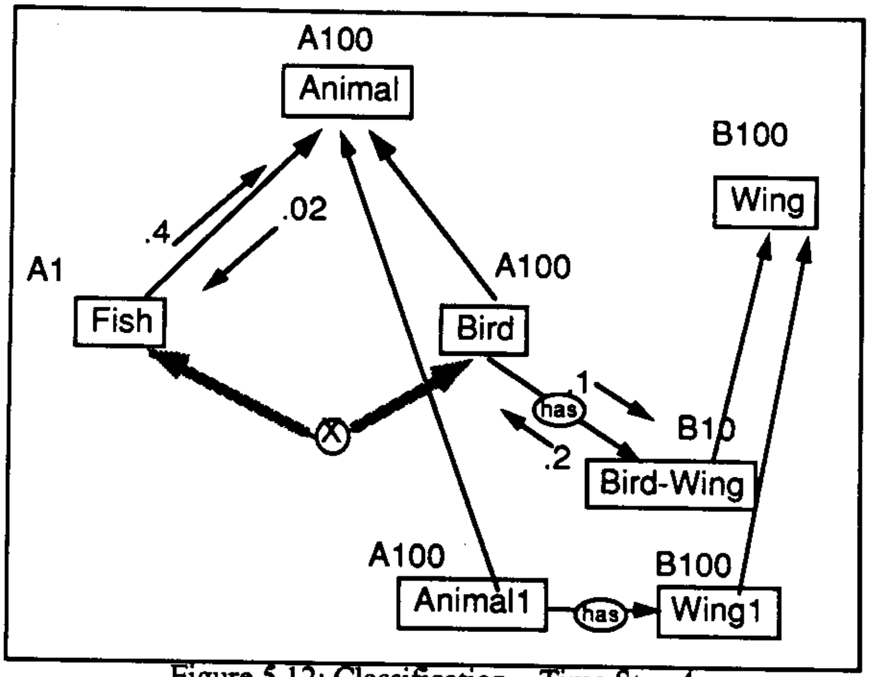


Figure 5.12: Classification—Time Step 4

Because the activation stops spreading with an activation of A100 on BIRD we can infer that ANIMAL1 is an instance of BIRD.

5.5. FINDING A RELATIONSHIP

Sherlock uses spreading activation to determine the relationship between two concepts. The relationship is then used to decide what type of gm-link Sherlock would have made, and compare that to the learner's gm-link.

In Chapter 4 (4.3.1) the antecedent facts used in Sherlock's graphic mapping strategy rules were listed. In order to use these rules to choose a gm-link, Sherlock must be able to automatically infer which of these facts are true of the relationship between any two concepts. Determining what is true about the relationship between two concepts is done by activating the nodes corresponding to the two concepts in the semantic network, and then allowing the activation to spread. Each of the antecedent facts is defined in terms of the activation pattern we would expect if the fact is true. Sherlock, then, is able to set each antecedent fact as true or false, depending on whether its requisite activation pattern is found.

It is assumed that each fact will be true for a range of configurations in the semantic network. For a definition to be useful it must be:

a. flexible enough to recognize most configurations in this range;

b. restrictive enough to ignore configurations that are outside of this range.

The following are the definitions of antecedent facts, in terms of activation patterns.

5.5.1. DEFINING X-IS-Y

X-is-Y is true if activation reaches Y with the same label as X and a strength greater than *min*. In Figure 5.13 CONTRACT is X and RELATIONSHIP is Y.

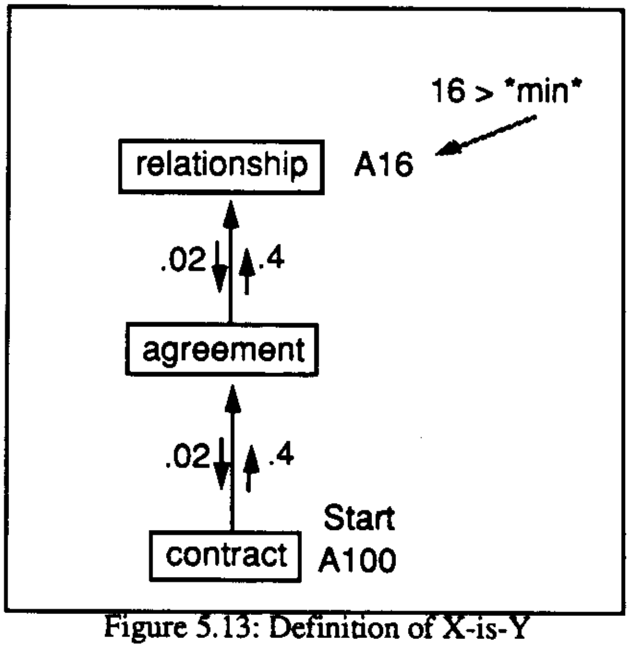


Figure 5.13: Definition of X-is-Y

X-is-Y is true for these two concepts because activation reaches RELATIONSHIP with a strength of 16 and the same label (A) as CONTRACT. X-is-Y will normally be true if Y is above X in the same inheritance tree. It is possible for X-is-Y to be true if Y is below X, but only if the two nodes are very close (one or two links apart) and the links are strong.

Thus X-is-Y does not indicate that X is a member of Y's conceptual class, only that there is a high probability that anything that is an X will also be a Y.

5.5.2. DEFINING Y-UNLIK-X

Y-unlik-X is true if the activation reaching X from Y (with the same label as Y) is less than the activation reaching Y from X (with the same label as X). In Figure 5.14, once again, CONTRACT is X and RELATIONSHIP is Y.

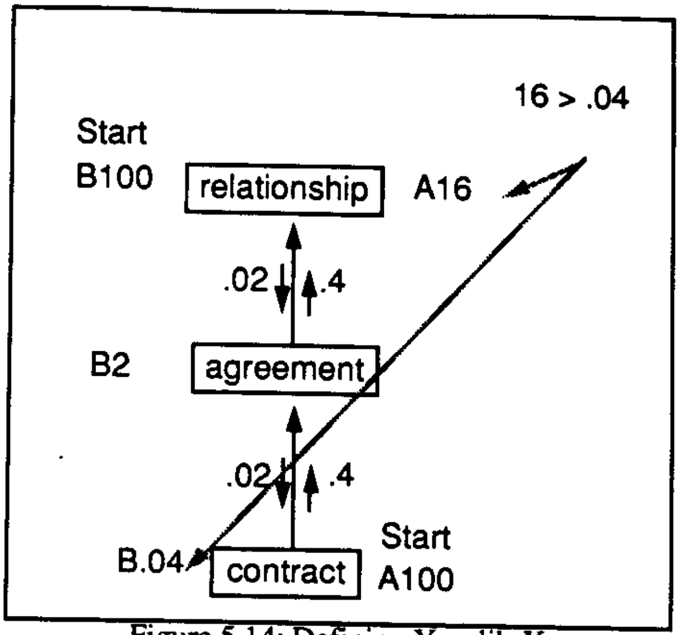


Figure 5.14: Defining Y-unlik-X

Y-unlik-X is true for these concepts because the activation which reached CONTRACT from RELATIONSHIP was only .04, which is less than the activation of 16 which reached RELATIONSHIP from CONTRACT.

5.5.3. DEFINING X-NOT-Y

X-not-Y is true if no activation reaches Y from X. In Figure 5.15 X is CONTRACT and Y is BIRD.

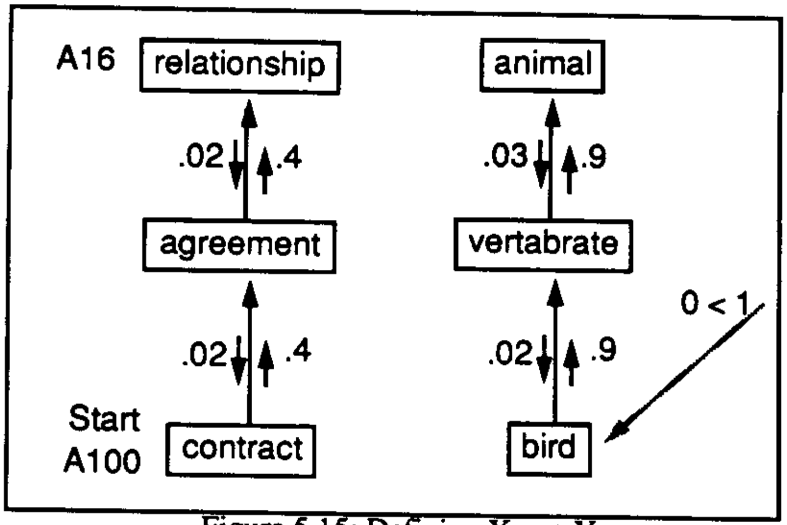


Figure 5.15: Defining X-not-Y

X-not-Y is true for these concepts because no activation reaches BIRD from CONTRACT. Note that an inhibition node is not necessary for Sherlock to decide that X-not-Y is true.

5.5.4. DEFINING EQ-CH

Eq-ch is true if the activation which reaches Y from X (with the same label as X) is equal to the activation which reaches X from Y (with the same label as Y). In Figure 5.16 eq-ch applies to the concepts PARTY1 and JOHN.

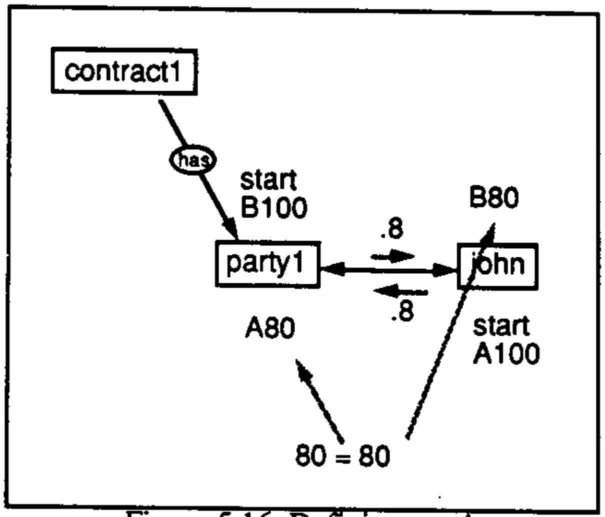


Figure 5.16: Defining eq-ch

Eq-ch is true for these concepts because the activation which reaches PARTY1 from JOHN is 80, which is equal to the activation of 80 which reaches JOHN from PARTY1.

5.5.5. DEFINING X-HAS-Y

X-has-Y is true if activation reaches Y from X with a different label from X and activation greater than minimum. In Figure 5.17 X is CONTRACT and Y is PARTY1.

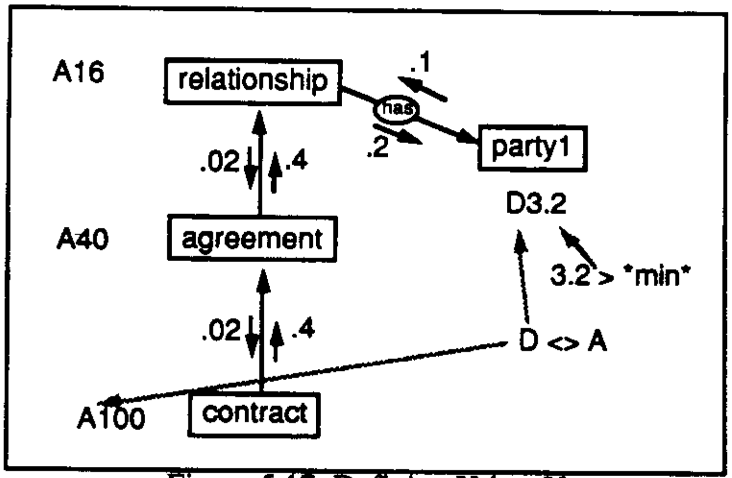


Figure 5.17: Defining X-has-Y

X-has-Y is true for these concepts because activation reaches PARTY1 from CONTRACT with a new label and a strength of 3.2 which is greater than minimum.

5.5.6. DEFINING X-COND-Y

X-cond-Y is true if activation reaches a role of Y from X and that role is a descendent of ACT:CONDITION. In Figure 5.18 X is CONS1-DONE and Y is CONS2.

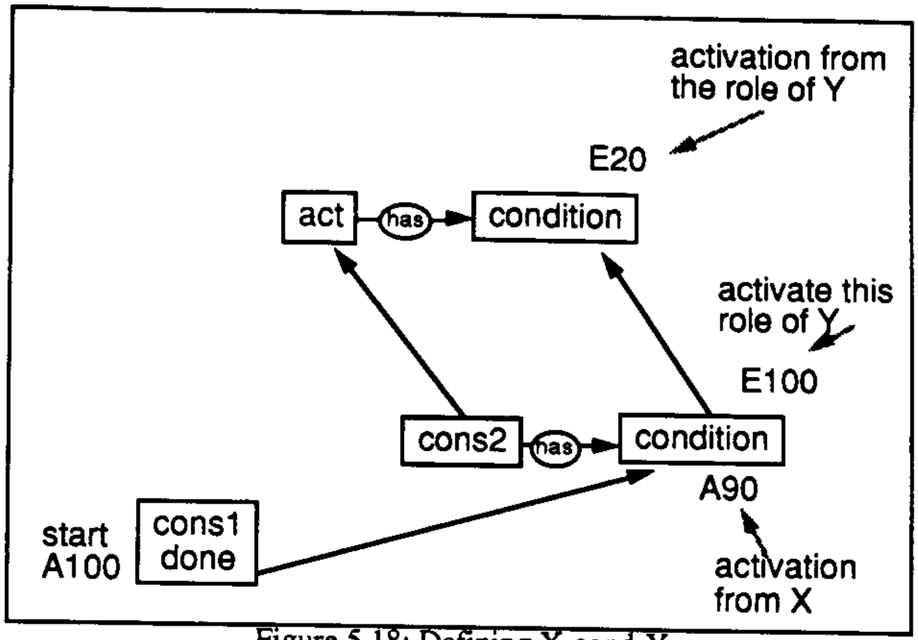


Figure 5.18: Defining X-cond-Y

To determine if X-cond-Y is true, Sherlock checks the roles of CONS2 to determine if activation from CONS1-DONE has reached any of them. In this case CONS2:CONDITION has received an activation of 90 from CONS1-DONE. CONS2:CONDITION is now activated to see if activation reaches ACT:CONDITION. Since activation reaches ACT:CONDITION, X-cond-Y is true.

5.6. SUMMARY

A knowledge representation must be designed to generate inferences that go beyond the knowledge that is explicitly represented. To generate inferences we must specify the processes that operate on the knowledge. Sherlock uses a spreading activation mechanism to generate inferences from its semantic network.

One goal in Sherlock was to use a process designed for more generic cognition and determine if the results of that cognition could be used to evaluate a link. The spreading activation mechanism in Sherlock was designed by Gasser (1988) to classify concepts. It has been shown, however, that the same network can be used to determine the strength and nature of the relationship between two nodes.

To determine the relationship between two nodes in the network, Sherlock must determine what types of paths exist between the two nodes. Spreading activation provides the most efficient mechanism for finding these paths. Activation spreads in all possible paths from each of the nodes in parallel. Once activation has stopped spreading, Sherlock can determine from the activations that reached the two nodes, what types of paths exist.

Possible facts about the relationship between two concepts have been defined in terms of activation patterns in the semantic network. These definitions are able to recognize most possible network configurations for which the fact is true, while ignoring most configurations for which the fact is not true. These facts are used as antecedents in the rules representing Sherlock's strategy for building a graphic map.

Chapter 6

Using Rules

6.0. INTRODUCTION

It was indicated in Chapter 4 that rules in Sherlock are represented in two separate schemes, each intended for a different function:

1. rules in the production system are used to model the use of strategy by both Sherlock and the learner;
2. rules in the semantic network are used to think about the strategy the learner was using.

The production system is designed to both direct Sherlock's tutoring and to use the knowledge in the semantic network to decide what Sherlock would have done every time the learner makes a gm-link. The rules in the semantic network are used to classify the learner's plan as being an instance of one of these plans.

6.1. MODELLING THE USE OF STRATEGY

Sherlock's production system represents its strategic knowledge, that is the knowledge that controls actions. The rule application mechanism is the process that operates on the productions. It is the rule application mechanism that actually controls actions, using the strategic knowledge contained in the production system.

The rule application mechanism uses a simple processing cycle. During each cycle one production **fires**, at which point its action portion is carried out. This processing cycle is started by giving the rule application mechanism a goal. Everything that happens during the processing cycles is designed to try and accomplish this goal. The cycle will continue to repeat itself until either the goal is accomplished, or the system determines that it cannot accomplish the goal.

The processing cycle consists of:

1. the match process, which finds productions whose antecedents are currently true;
2. the conflict resolution process, which selects which of the matching productions should be used during a given cycle;
3. the act process, which carries out the consequents of the chosen productions.

The consequents of productions will often direct the system to add or change information in working memory. Thus during each cycle there is a good chance that the match process will find different productions whose antecedents are true.

6.1.1. THE PROCESSING CYCLE

6.1.1.1. MATCHING PROCESS

The matching process creates a list of productions that are appropriate for the current cycle. A production is put on this list of possible productions if it meets two criteria;

1. each of the antecedents of the production can be matched against the information that is currently in working memory;
2. the production has not already fired for the current goal.

The second criteria is necessary to insure that the rule application mechanism does not get caught in an endless loop. If a production fired that did not change anything in working memory, for example, without the second criteria that production would continue to fire during each succeeding cycle.

6.1.1.2. CONFLICT RESOLUTION PROCESS

The conflict resolution process chooses one rule from the list of possible productions. If there is only one possible production, there is no conflict. If there is more than one possible production, Sherlock chooses the production with the highest strength.

To enable Sherlock to abandon goals for which there are no appropriate productions, Sherlock has the nothing-rule pictured in figure 6.1

The nothing-rule has only one antecedent, T. Since this will always be true, the nothing-rule is considered for any goal Sherlock is pursuing. Because the nothing-rule strength is zero, however, any other rule with a positive strength will be used first. If a goal has not been satisfied, and there is no good rule left, the nothing-rule will direct Sherlock to abandon the current goal, rather than firing a buggy rule.

```
(tproduction!
 'nothing
 '(
   t
   )
  '(
    (pop *goal*)
   )
 0)
```

Figure 6.1: Nothing Rule

In Sherlock's model a buggy rule represents a strategy that an individual has, but no longer considers appropriate. A bad rule cannot be "erased" from the production system, instead it is given a negative weight so that it will not be used under ordinary circumstances.

6.1.1.3. ACT PROCESS

The consequent of each rule in the production system is a list of Lisp expressions. Once Sherlock has chosen the one rule that will be used for the current cycle, the antecedents for that rule are executed. That rule is then added to the list of rules that have been used for the current goal.

The cycle is then repeated. Rules will be considered in the next cycle according to any changes the preceding rule made in short-term memory or goals.

6.1.2. GOALS

It was indicated above that the function of the processing cycle is to attempt to accomplish the goal that is given to the rule application mechanism. When the rule application mechanism begins this goal is called the **current goal.** During each processing cycle, one of four situations will exist for the current goal;

1. All of the conditions necessary to satisfy the current goal are true; in which case the goal has been accomplished.
2. All of the conditions necessary to satisfy the current goal are not true, but the system knows an action that can satisfy some of the conditions; in which case the system will try that action and then reconsider the current goal.
3. All of the conditions necessary to satisfy the current goal are not true, but the system knows a sub-goal that can satisfy some of the conditions; in which case the system will temporarily make the sub-goal the current goal.

4. All of the conditions necessary to satisfy the current goal are not true, and the system does not know a sub-goal or action that can satisfy any of the conditions; in which case the system will abort the current goal.

Sherlock uses a goal stack to keep track of goals. The function of the goal stack is to insure that the system returns to a previous goal once a sub-goal has either been accomplished or aborted. The top goal on the stack is the current goal.

Sherlock has productions for the following goals:

make-gm-link	choose an appropriate gm-link for two icons;
eval-link	evaluate the gm-link that the learner just made;
feedback	provide feedback to the learner for the gm-link the learner just made;
change-net	consider changing the semantic network to reflect the learner's current beliefs;
find-plan	find the plan that the learner used to make the current gm-link
pick-plan	pick from among the possible plans the most likely plan the learner used to make the current gm-link (used as a sub-goal for find-plan);
tutor-text	provided tutoring for a misunderstanding of knowledge contained in the text;
tutor-plan	provide tutoring for a misunderstanding of the strategy for constructing a graphic map.

To illustrate the role of goals in controlling the rule application mechanism, let us consider Sherlock's strategic knowledge for tutoring. When the learner makes a link, the rule application mechanism is given the goal to provide the learner with feedback (FEEDBACK). In order to provide feedback, Sherlock must know whether the learner's link was good. Thus during the first processing cycle for the goal FEEDBACK, one missing necessary condition is knowing if the link is good. To satisfy this condition there is a production that says in effect;

If the goal is to provide feedback
and you do not know if the link is good
then put the sub-goal eval-link on top of the goal stack

Now that the current goal is to evaluate the link, the rule application mechanism will consider productions that can satisfy this goal. A necessary condition for evaluating a link is to know what the relationship is between the two icons. Sherlock does not know what the relationship is, but there is a production that can initiate an action to find this out;

If the goal is to evaluate the link
and you do not know what the relationship is between the icons
then do a spreading activation on the two icons

The above production does not change the goal stack since it directly executes the action rather than creating a sub-goal. This production does not remove the current goal because the action only supplies a necessary condition of the goal, it does not accomplish the goal. Thus, once this action is complete, the rule application mechanism will again consider productions for accomplishing the goal of evaluating the link.

At this point it is possible that sufficient conditions exist for accomplishing the goal of evaluating the link. For example, if the spreading activation has determined that Sherlock would have made the same gm-link the learner made, the following production will fire;

If	the goal is to evaluate the link
and	you know what kind of gm-link Sherlock would have made
and	Sherlock would have made the same kind of gm-link the learner made
then	decide that the learner used the same production Sherlock would have used
and	decide that the plan status is OK
and	remove the current goal

If the conditions for this production are not true, other possibilities include;

1. deciding that the learner used an acceptable production that is different from the production that Sherlock would have used;
2. deciding that the system does not know what production the learner used, so the system sets a sub-goal to try and classify the learner's plan;
3. deciding that the system does not have enough information to evaluate the link, so the goal of evaluating the link is abandoned.

Let us assume that the learner made the same gm-link Sherlock would have made, so that the above production is fired. By identifying the plan and labelling it as OK, this production has accomplished the goal of evaluating the link. That is why the last consequent of this production removes the current goal EVAL-LINK.

Once EVAL-LINK is removed from the goal stack, the next goal on the stack, FEEDBACK, is again the current goal. This time there is a production that recognizes that there are sufficient conditions for satisfying the goal of providing feedback;

If	the goal is to provide feedback
and	you do know that the link is good
and	you know that the learner used the same production that Sherlock would have used
then	tell the learner that his action was very good
and	remove the current goal

Once the current goal FEEDBACK is removed from the stack, there are no goals left on the stack. When the goal stack is empty the rule application mechanism stops, and Sherlock waits for the learner to make another gm-link.

Sherlock maintains the goal stack as a list. A sample goal stack is included in the sample transcript shown in Figure 6.2.

```
CYCLE   4  GOALS:  ((FIND-PLAN) (EVAL-LINK NO-
PRODS FIND-PLAN) (FEEDBACK NOT-EVALD))
PRODUCTION:  CHECK-ALL-PLANS
```

Figure 6.2: Sample Goal Stack

Each goal on the goal stack is represented as a list. The first item on each goal's list is the name of the goal. The remaining items on the list are the names of the rules that have already fired for that goal. The first goal on the goal stack list is the current goal, in this case FIND-PLAN. Since FIND-PLAN is the only item on the first goal's list, we know that no productions have fired yet for this goal.

Anytime a production sets a new goal it is *pushed* onto the top of the list. When the current goal is satisfied, or Sherlock gives up on the current goal, the current goal is *popped* off of the list, making the second goal the first and thus the current goal.

The second goal in Figure 6.1 is EVAL-LINK. Sherlock will return to this goal when the goal FIND-PLAN is popped off the list. We can see in Figure 6.1 that the last time Sherlock worked on the goal EVAL-LINK two productions were already fired; NO-PRODS FIND-PLAN. These two productions, then, cannot be considered when Sherlock returns to this goal.

6.2. MODELLING THINKING ABOUT STRATEGY

Sherlock uses the representation of rules in the semantic network to make inferences about the strategy that the learner was using. Rules in the semantic network are used to answer two kinds of questions:

1. Which plan was the learner trying to use;
2. Why didn't the learner use a more appropriate rule.

6.2.1. DETERMINING THE LEARNER'S PLAN

To decide what rule Sherlock believes the learner was trying to use, the representation of rules in the semantic network is used. Figure 6.3 contains the representation for three of the rules for making a gm-link.

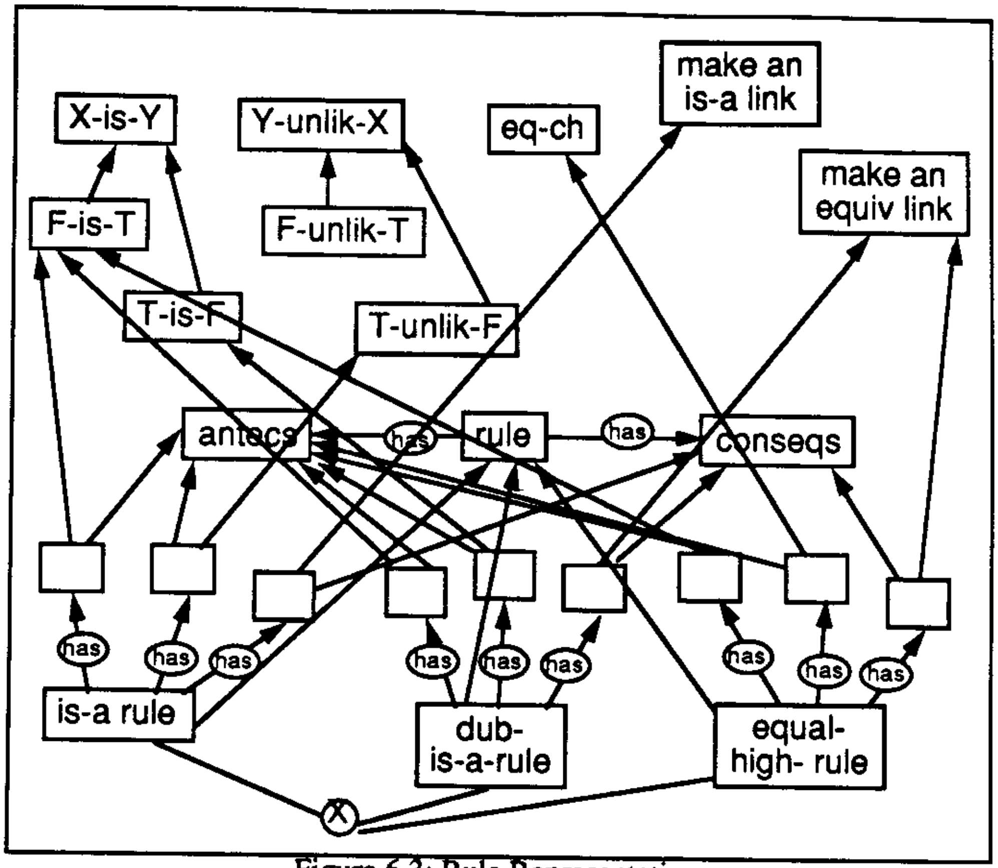


Figure 6.3: Rule Representation

6.2.1.1. CLASSIFYING THE PLAN

Assume, for example, that the learner makes the EQUIV gm-link shown in Figure 6.4.

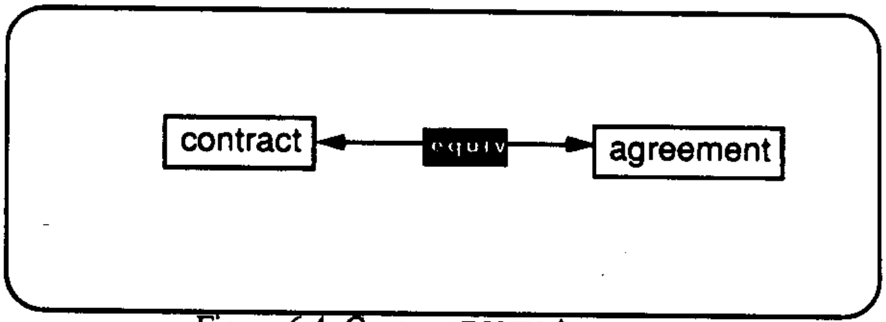


Figure 6.4: Contract EQUIV Agreement

Sherlock does a spreading activation on the concepts CONTRACT and AGREEMENT to determine that the following facts are true about the relationship between these concepts:

F-IS-T

T-UNLIK-F

The rules in the production system are then used to determine what Sherlock would have done. Sherlock determines that three rules are potential productions when the above facts are true:

IS-A-RULE

WEAK-IS-A-RULE

BUGGY-HALF-IS-A-RULE

Of these rules, the first two are non-buggy, and thus represent rules that Sherlock would have used to link the two icons. Both of the non-buggy rules would have directed Sherlock to make an IS-A gm-link between these two icons. There is no potential production that would have led to the EQUIV gm-link that the learner made.

Since Sherlock would have made a different gm-link than the learner, the learner is asked to indicate the reasons he made this link. He gives the following two reasons:

1. A contract is probably an agreement (F-IS-T).
2. The probability that a contract is an agreement is equal to the chance that an agreement is a contract (EQ-CH).

Sherlock's goal is to classify the learner's plan as being an instance of one of the rules represented in the semantic network. The first step is to represent everything Sherlock knows about the learner's plan. This is done by;

1. placing an initial activation on RULE with the label A
2. placing a temporary link[1] between each of the two facts indicated above and the ANTECS role of RULE;
3. placing an initial activation on the two facts and ANTECS with the label B;
4. placing a temporary link between MAKE-AN-EQUIV-LINK and the CONSEQS role of RULE;
5. placing an initial activation on MAKE-AN-EQUIV-LINK and CONSEQS with the label C

Figure 6.5 contains the temporary links (displayed as grey links) and activations indicated above.

---

[1] This is the mechanism in Gasser's network package for simulating the existence of an unknown concept that you wish to classify.

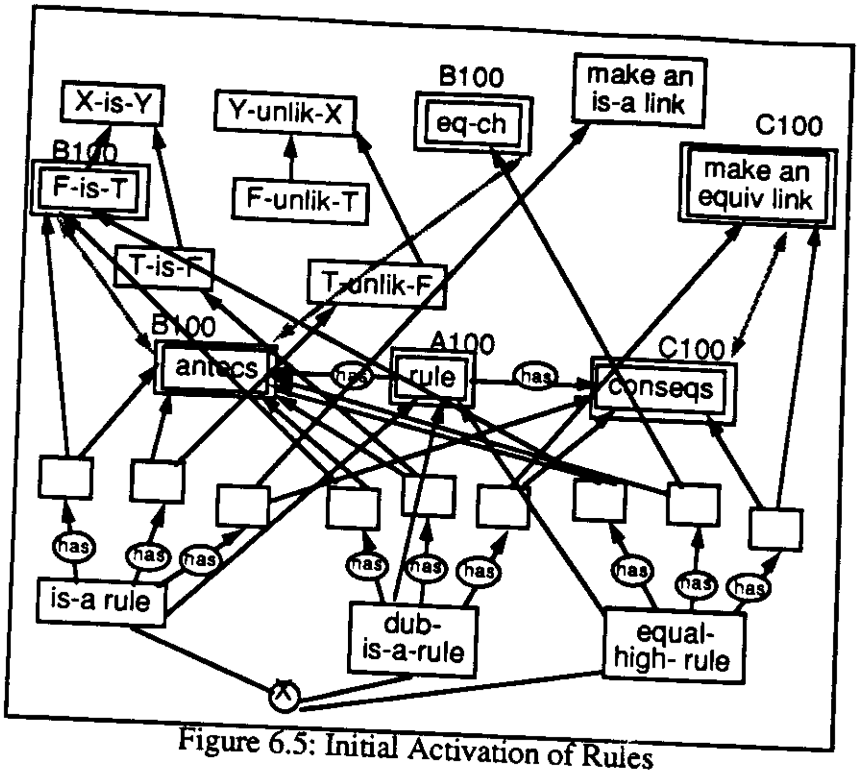


Figure 6.5: Initial Activation of Rules

During the first time step the activation spreads to the specializations of the facts and the action which are roles of rules (Figure 6.6). Each rule also receives some activation from RULE.

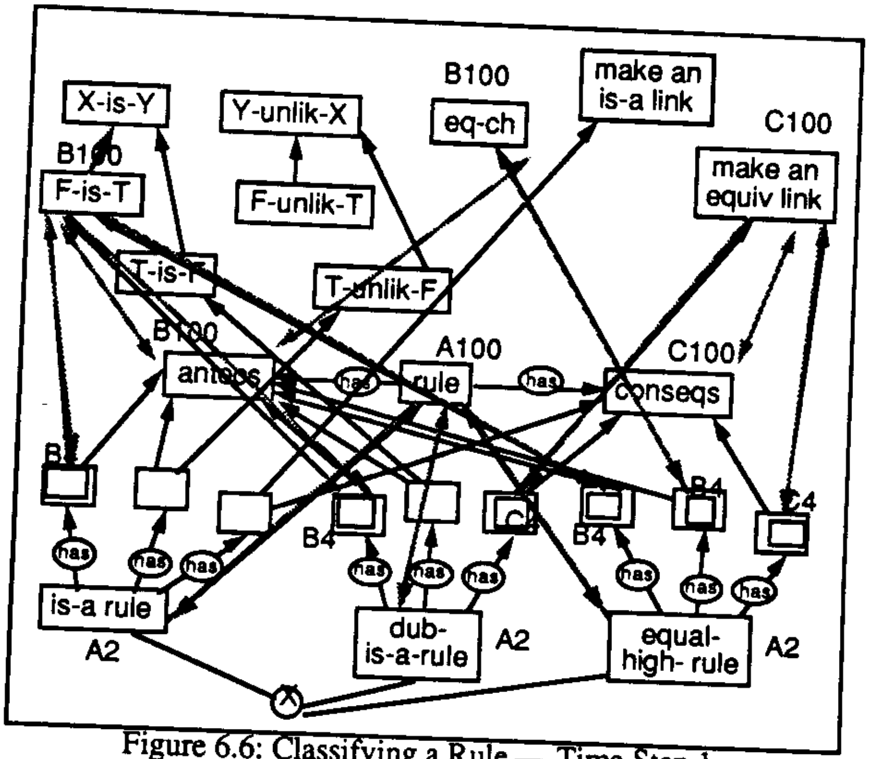


Figure 6.6: Classifying a Rule — Time Step 1

The nodes with double boxes (in the figures) are the newly activated nodes, and thus activation will spread from the nodes in the next time step. In time step 2 the activations reach the rules as shown in Figure 6.7.

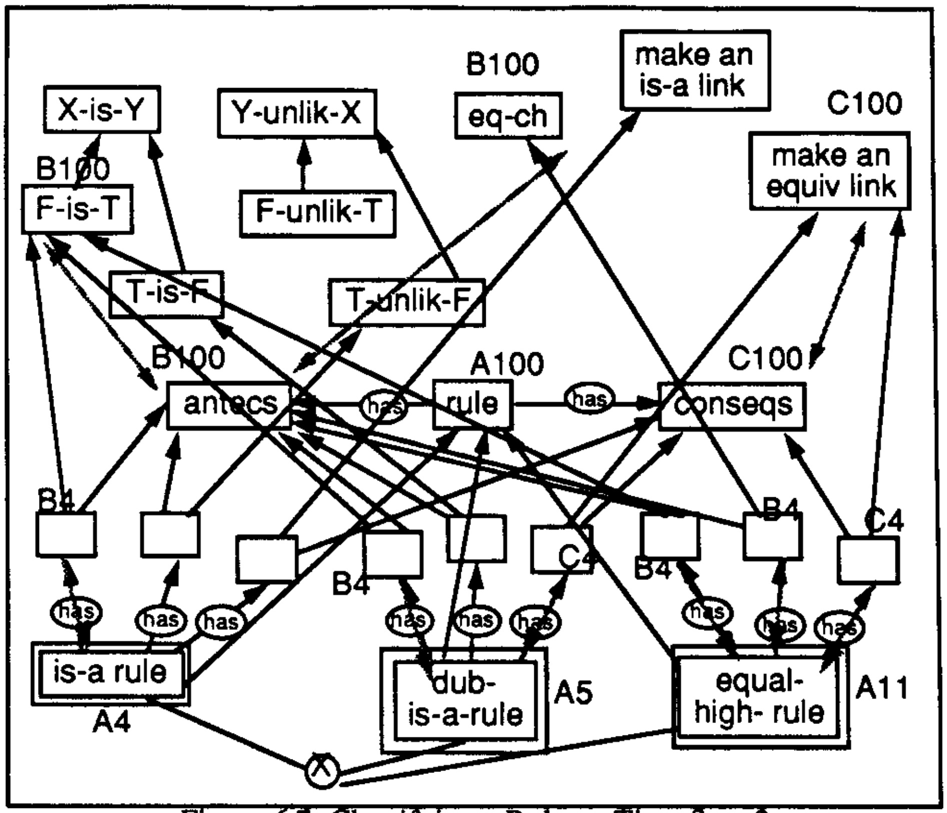


Figure 6.7: Classifying a Rule — Time Step 2

In the next two time steps, the inhibition node connecting each type of rule will adjust the activations on the rules. In time step 5 the network will settle on EQUAL-HIGH-RULE (Figure 6.8).

Sherlock thus concludes that the learner's plan was an instance of EQUAL-HIGH-RULE.

In the above example, classification was straight forward since all of the roles of the rule were indicated. The same process, however, will usually find the closest match when partial information or even contradictory information is available.

6.2.1.2. PROBLEMS WITH RULE CLASSIFICATION

There are two, in some ways opposite, problems with using the classification properties of the semantic network to determine which plan the learner has used.

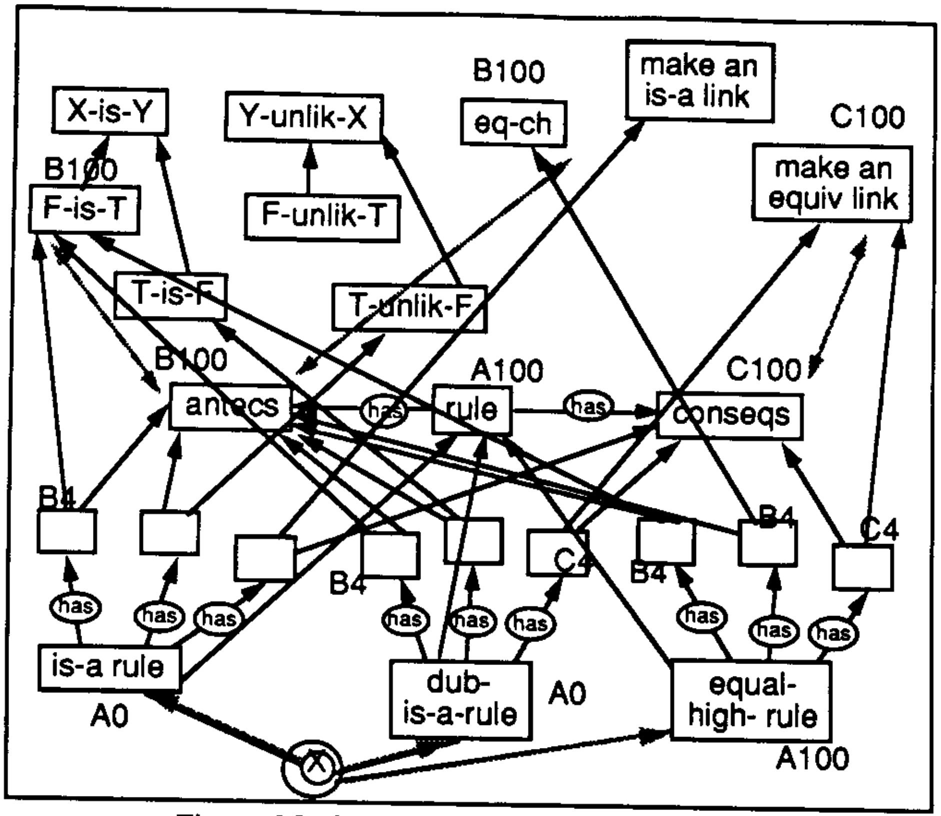


Figure 6.8: Classifying a Rule — Time Step 5

6.2.1.2.1. UNABLE TO CHOOSE AMONG EQUALLY VIABLE OPTIONS

There are multiple rules in Sherlock's strategy leading to each gm-link type. Each of these rules has antecedents that are sufficient conditions for making the link. Occasionally learners will indicate that they made a gm-link because of all of the antecedents that would have led to that gm-link type.

When Sherlock tries to classify the plan, each of the rules is equally possible because all of their antecedents were indicated, and their consequents matched the link the learner made. Since each of rules receives equal activation, the inhibition node is not able to assist the network in settling on one of them. Therefore Sherlock cannot decide what plan the learner was using.

Humans often have multiple reasons for taking a particular action. Assume, for example, that a human being has the rules listed in Figure 6.9 for leaving the house in the morning.

If it is raining
then bring an umbrella

If the clouds are grey
then bring an umbrella

If the weatherman said it might rain
then bring an umbrella

If I am travelling to a city where it is raining
then bring an umbrella

Figure 6.9: Rules for leaving the house in the morning

Imagine that on a particular morning this human being looks out the window and sees that it is raining. He remembers that the weather man said that it would rain today. He looks at the clouds and they are a dark grey. To make matters worse, he is flying to Seattle today. It seems clear that he would take an umbrella, but why?

It is likely that when we have a number of competing rules that would lead to the same action, we recognize that and take the common action without worrying about which specific rule was most responsible. Sherlock needs to know which rule was used so that it knows which false facts need to be explained to the learner. But if a number of rules all lead to the same action, and the learner believes the facts for each of the rules, Sherlock has to be able to refute each of the rules in order to convince the learner that the link was not good.

6.2.1.2.2. SETTLES ON WEAK RULE WITHOUT SUFFICIENT EVIDENCE

The second rule classification problem is the reverse of the above problem. The network will classify something as an instance of a concept with very little evidence if there is no contradictory evidence and there is no better possibility. Thus if only a weak antecedent of one of Sherlock's rules matches the facts which the learner has indicated, but that fact appears in no other link rule, and the consequent matches the action the learner took, Sherlock will decide that that is the plan which the learner used.

If we believe that human beings can only be male and female, it takes very little evidence to convince us that a particular instance is one or the other. The problem in Sherlock is while Sherlock has to account for the fact that other possible plans exist, it just does not know what they are. Thus Sherlock may incorrectly decide that the learner used an instance of one of the known rules, rather than creating a new type of rule that better describes the rule that the learner used.

6.2.2. DECIDING WHICH FACTS ARE PROBLEMS

After Sherlock does a spreading activation to classify the learner's plan, the semantic network is used to determine which facts may be causing problems. Sherlock looks for two types of fact problems:

1. false facts that were used to pick the inappropriate plan because the learner believes them to be true;
2. true facts that that might have led to the appropriate plan if the learner believed them.

6.2.2.1. FALSE FACTS

To determine which false facts are a problem, Sherlock takes the following steps:

step 1 determine which roles of the learner's plan have an activation greater than 2, and the same marker that started on ANTECS.

These are the roles which are specializations of facts that the learner indicated are reasons he used the plan. Roles with an activation of less than 2 probably received their activation from the rule, and not from any facts that were indicated by the learner. In Figure 6.8 we see that EQUAL-HIGH-RULE-F-IS-T and EQUAL-HIGH-RULE-EQ-CH (the two antecedent roles of EQUAL-HIGH-RULE) each have an activation of 4 with the label B (B is the label on ANTECS).

step 2 Sherlock finds the ancestor of each of the roles found in step 1 that is a descendent of RELATIONSHIP-FACT.

In Figure 6.8 this includes F-IS-T and EQ-CH.

Sherlock's short-term memory still has a list of facts that are true about the current relationship.

step 3 Sherlock checks the ancestors found in step 1 to see if they are on the list of true facts about relationships. If an ancestor is not true, it is put on the list *why-wrong*.

The list *why-wrong* represents the wrong facts that the learner believes, that led to the use of an inappropriate plan. In this case the fact EQ-CH is the only fact on the list *why-wrong*.

6.2.2.2. TRUE FACTS

To determine which true facts are a problem, Sherlock takes the following steps:

step 1 determine which roles of the plan Sherlock would have used have an activation less than 2.

If a role has an activation of less than 2, then that role was not indicated by the learner as a reason he made the link. In Figure 6.8 we see that IS-A-RULE-T-UNLIK-F and IS-A-RULE-MAKE-AN-IS-A-LINK have no activation.

step 2 Sherlock activates each of the roles in step 1 to see which ones are descendents of RELATIONSHIP-FACT. The parents of these roles are then placed on the list *why-not*

The list *why-not* represents the true facts that the learner does not believe, that might have led to the use of the appropriate plan. Sherlock does not have to check to see if they are true. Since they are antecedent roles of the plan Sherlock would have used, they must be true. The fact T-UNLIK-F is an antecedent of the plan Sherlock would have used, and thus Sherlock knows that is it true.

In this case the fact T-UNLIK-F is the only fact on the list *why-not*.

6.2.2.3. PROBLEMS WITH DEFINING FACTS

Sherlock's ability to use a learner's fact beliefs is contingent on receiving an accurate answer to the question, "Why did you do that?"

Inaccuracies in learners' answers have been observed in several ways:

1. A learner says one thing (during think-aloud protocols), then picks a seemingly contradictory fact.
2. Think-aloud protocols indicate that a learner interprets a fact differently from Sherlock.
3. A learner picks two contradictory facts.

A major portion of the problem is the difficulty of expressing Sherlock's facts in natural language to the learner. The facts are represented internally in Sherlock as specific activation patterns in the network. But even if a learner has something analogous to Sherlock's semantic network, he has no way of observing activation patterns in that network.

Thus the facts must be stated in the common terms we use to describe relationships. But this introduces two problems:

1. There is no one-to-one correspondence between natural language statements and actual relationships.
2. The way that a given relationship is expressed will vary from learner to learner.

6.3. SUMMARY

Rules represented in the production system are used to model how an individual uses strategic knowledge to accomplish a task. A three step processing cycle determines what actions should be taken at any given moment. Sherlock uses this processing cycle for two purposes:

1. to determine what kind of gm-link Sherlock would make between any two icons using the productions for making gm-links;

2. to direct the tutoring session using the productions that represent Sherlock's tutoring strategy.

Rules represented in the semantic network allow Sherlock to make inferences about the learner's knowledge and strategy for constructing a graphic map. Spreading activation is used to classify the learner's plan as an instance of one of the rules represented in the network.

A straight matching of facts and rules is capable of classifying the learner's plan only if we have a complete and accurate list of the facts that the learner believes. Spreading activation gives Sherlock the possibility of classifying the plan even if the information that the learner gives Sherlock is not complete or is inaccurate. Being able to classify the plan despite faulty information is important since it is difficult for learners to accurately assess why they did a particular action. Straight matching is thus ineffective in a tutoring systems.

Classifying the learner's plan is not essential for recognizing an error, only for more accurately indicating to the learner the cause of the error. If Sherlock cannot classify a learner's plan, Sherlock can only indicate to the learner that the plan is not recognized.

Other functions are able to use the semantic network to determine the relationship between facts and rules.

Two factors cause problems for the processing of rules in the semantic network:

1. the inability of the network to recognize that a concept does not fit any existing category;
2. the difficulty people have in explaining why they took a particular action.

Both factors also cause problems in human tutoring. Humans have a tendency to try and fit new experiences into old categories, rather than create new conceptual classes. The separation of strategic knowledge from knowledge about strategy reflects the difficulty we have in explaining our actions.

Chapter 7

Trace and Implementation of Sherlock

7.0. INTRODUCTION

Sherlock is implemented in T (Reese, Adams, & Meehan, 1984), a dialect of Lisp (Weissman, 1967). The program runs on Apollo workstations using a graphics interface written by Seth Goldman (at the UCLA AI Lab) and Gasser's semantic network package (1988).

The current version of Sherlock evaluates each link as it is made. The program asks, in effect, "Is there any reason I would have made that type of gm-link between those two icons?" If Sherlock cannot find a good reason for the link, an attempt is made to determine why the learner made the link. Sherlock will then provide the learner with feedback based on the diagnosis.

Evaluation, diagnosis, and feedback are controlled by Sherlock's tutoring strategies. These strategies are represented in the production system as described in Chapter 4. The tutoring strategies are an attempt to model the thoughts of a human tutor while watching a learner build a graphic map. This chapter will describe these tutoring strategies both in terms of human tutoring and their implementation in Sherlock.

What follows is a trace of Sherlock as the program evaluates and responds to the links made by Janey in Chapter 1. Janey began by making the link illustrated in Figure 7.1.

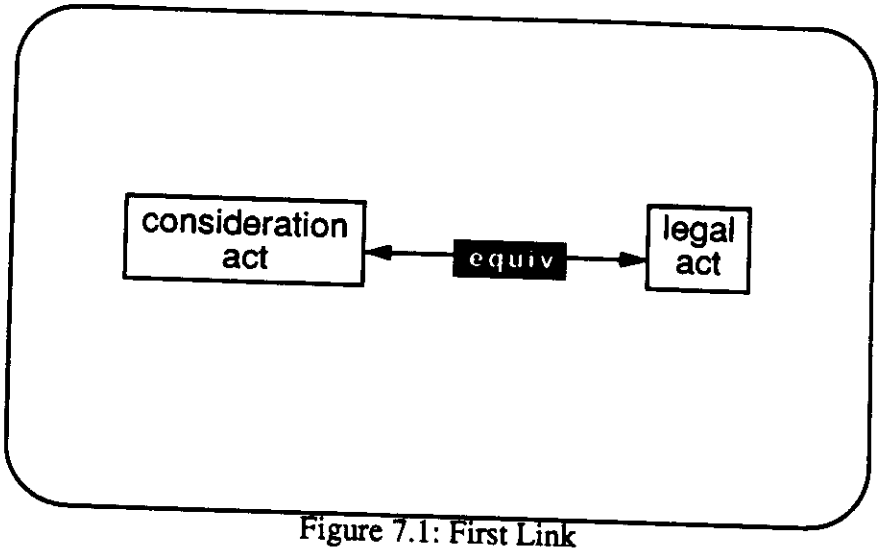


Figure 7.1: First Link

7.1. DETERMINING THE RELATIONSHIP BETWEEN ICONS

When the learner makes a link, the production system is started with a goal to provide the learner with feedback.

Figure 7.2 shows the first 2 cycles of the production system.

```
I-CONS-ACT-ICON -- EQUIVS -> I-LEGACT-ICON

Starting production system.

CYCLE    1  GOALS:   ((FEEDBACK))
PRODUCTION:  NOT-EVALD

CYCLE    2  GOALS:   ((EVAL-LINK) (FEEDBACK NOT-
EVALD))
PRODUCTION:  NO-PRODS
[Binding *DEST*]
First propagation: from node CONS-ACT
Initial excitation:
   CONS-ACT A
```

Figure 7.2: Starting the Production System

For each cycle of the production system the transcript includes the cycle number, the goal list, and the production which is executed in that cycle. As was described in

Chapter 6, each goal on the goal list is itself a list. The first item is the goal, the remaining items are the productions that have already been used for that goal.

In cycle 1 (Figure 7.2) we see that the current goal is to provide feedback, and that no productions have yet been used for this goal. The production chosen for cycle 1 is NOT-EVALD. NOT-EVALD says that if the current goal is to provide feedback and you do not know what you would have done with these icons, then set a goal to evaluate the link.

In cycle 2 the goal FEEDBACK remains on the goal list with the production that has just fired but the current goal is now EVAL-LINK. The production used in cycle 2 is NO-PRODS which says that if the goal is to evaluate the link and you do not know what you would have done with these icons, then activate the nodes corresponding to the strongest interpretations for each of the icons, and allow that activation to propagate. The last thing that takes place in Figure 7.2 is that the node corresponding to the from-icon, CONS-ACT, is activated with the label A.

Figure 7.3 contains the transcript describing the results of the first two time steps of spreading activation after CONS-ACT is activated.

```
T I M E   S T E P   1
Inhibiting from CONSIDERATIONS for A
   CONS-ACT 100
   CONS-PROM 0
   CONS-FORB 0
Excitation from CONS-ACT with A
   CONS-ACT-ICON::IOI 10
   ACT-ICON::IOI 10
   OPTACT 40
   CONSIDERATION 40

T I M E   S T E P   2
Inhibiting from OPTIONALITY for A
   RECQACT 0
   OPTACT 40
Excitation from CONS-ACT-ICON::IOI with A
   IOI 4
Excitation from ACT-ICON::IOI with A
   ACT 6
   IOI 8
Excitation from OPTACT with A
   OPTACT-ICON::IOI 4
   ACT 10
Excitation from CONSIDERATION with A
   RETURN-FOR-PROM-ICON::IOI 4
   CONSIDERATION-ICON::IOI 4
   LEGACT 16
   ACT 14
```

Figure 7.3: Activation Spreads from CONS-ACT

During the first time step the inhibition node which connects the types of CONSIDERATION is activated (Figure 7.4). This activation has little impact since CONS-ACT already has the maximum activation and there is no activation on CONS-FORB and CONS-PROM. This inhibition merely insures that if any activation does eventually reach CONS-FORB or CONS-PROM it will be ignored. Also of no importance in time step 1 is the activation which reaches the IOI roles of CONS-ACT-ICON and ACT-ICON.

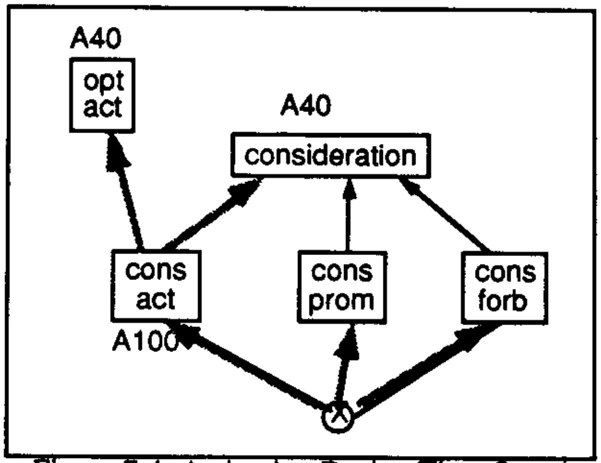


Figure 7.4: Activation During Time Step 1

The important activations in step 1 are those that spread up to OPT-ACT and CONSIDERATION, the two conceptual classes of which CONS-ACT is a member. In step 2 (Figure 7.5) the activation from CONSIDERATION spreads to LEGACT with a strength of 16. The activation continues to spread for one more time step, but is of no importance to this evaluation.

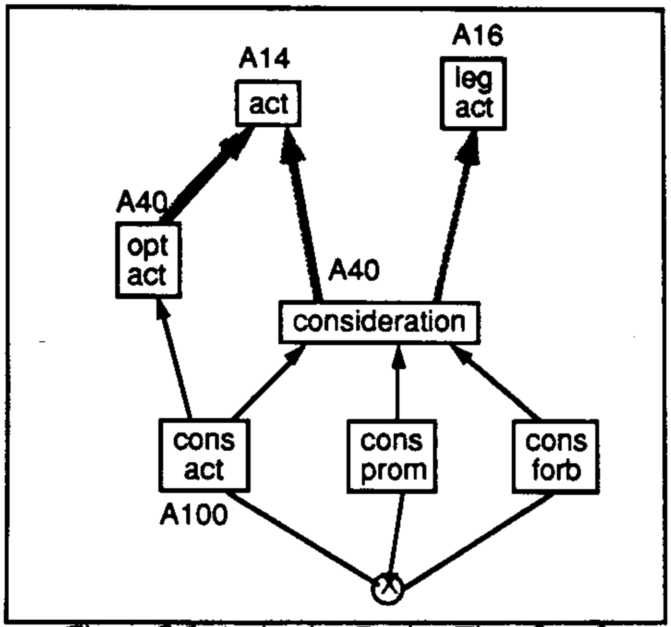


Figure 7.5: Activation During Time Step 2

Once the activation from CONS-ACT stops spreading, Sherlock sets facts about the "from" icon as being true or false. Each fact is set by examining the current activation on certain nodes in the network. The only fact set to true at this point is F-IS-T. F-IS-T is set by checking the current activation on the "to" node. If there is an activation level on the "to" node associated with the same marker as the "from" node, and a strength greater than 2, F-IS-T is set to true. In this case LEGACT has an activation of 16 associated with the A marker, so F-IS-T is set to true.

Figure 7.6 contains the portion of the transcript indicating the spreading activation from the LEGACT, the node corresponding to the to-icon.

```
Second propagation: to node
Initial excitation:
   LEGACT A

T I M E   S T E P   1
Inhibiting from LEGALITY for A
   ILLEGACT 0
   LEGACT 100
Excitation from LEGACT with A
   LEGACT-ICON::IOI 10
   ACT 10

T I M E   S T E P   2
Excitation from LEGACT-ICON::IOI with A
   IOI 4
Excitation from ACT with A

reached limit
```

Figure 7.6: Second Activation

Figure 7.6 indicates that the activation from LEGACT quickly dies out, with nothing of importance to the present evaluation being activated. At this point facts about the "from" node are set to true or false. The only fact set to true is T-UNLIK-F. T-UNLIK-F is set to true if the current activation on "from" node is less than the activation that had been found on the "to" node following the activation from the "from" node. In this case CONS-ACT currently has no activation so it is less than the activation that was found on LEGACT after the first activation.

7.2. DETERMINING WHAT SHERLOCK WOULD HAVE DONE

Based on the activations of CONS-ACT and LEGACT the transcript in Figure 7.7 indicates that Sherlock would have used IS-A-RULE to make an IS-A link from "consideration act" to "legal act".

```
Therefore Sherlock would:

Action:
Make a IS-A link  from CONS-ACT to LEGACT
Because of rule: IS-A
```

Figure 7.7: Sherlock's Conclusion

The IS-A-RULE is shown in Figure 7.8 (repeated from Figure 4.3).

is-a-rule

IF	X-is-Y
and	Y-unlik-X
THEN	make an IS-A link from X to Y

Figure 7.8: IS-A-RULE

This rule is used because F-IS-T and T-UNLIK-F were found to be true. Thus Sherlock has instantiated this rule with CONS-ACT as X and LEGACT as Y, making an IS-A link from CONS-ACT to LEGACT. Sherlock chose this rule because it was the matching rule in the production system with the highest strength.

For tutoring purposes, Sherlock also remembers any rules which matched but were not optimal (Figure 7.9).

```
Less optimal actions include:

Action:
Make a IS-A link  from CONS-ACT to LEGACT
Because of rule: WEAK-IS-A


buggy moves include:

Action:
Make a IS-A link  from CONS-ACT to LEGACT
Because of rule: HALF-IS-A
```

Figure 7.9: Non-Optimal Rules

In this case Sherlock found one less optimal rule and one buggy rule. A non-optimal rule is a rule with a strength greater than or equal to zero but less than the strongest matching rule. A buggy rule is a rule with a strength less than zero.

Control now returns to the production system as indicated in Figure 7.10.

```
CYCLE    3  GOALS:   ((EVAL-LINK NO-PRODS)
(FEEDBACK NOT-EVALD))
PRODUCTION:  FIND-PLAN

CYCLE    4  GOALS:   ((FIND-PLAN) (EVAL-LINK NO-
PRODS FIND-PLAN) (FEEDBACK
NOT-EVALD))
PRODUCTION:  CHECK-ALL-PLANS
```

Figure 7.10: Return to Production System

In cycle 3 the goal is still to evaluate the link. The rules considered for this goal check to see if any of the potential rules found in cycle 2 (in this case IS-A-RULE and WEAK-IS-A-RULE) have consequents that lead to the same gm-link the learner made (i.e., CHECK-BEST in Figure 4.11).

7.3. GETTING MORE INFORMATION

Since neither IS-A-RULE or WEAK-IS-A-RULE lead to the EQUIV link which Janey made, the production FIND-PLAN is fired. FIND-PLAN sets a new goal to find the plan which the learner used. The production fired for this goal is CHECK-ALL-PLANS which generates the menu shown in Figure 7.11.

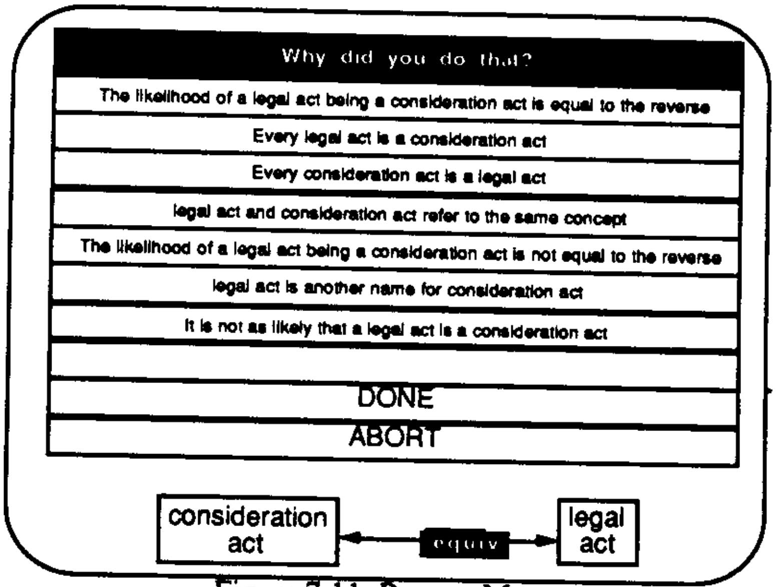

Why did you do that?
The likelihood of a legal act being a consideration act is equal to the reverse
Every legal act is a consideration act
Every consideration act is a legal act
legal act and consideration act refer to the same concept
The likelihood of a legal act being a consideration act is not equal to the reverse
legal act is another name for consideration act
It is not as likely that a legal act is a consideration act
DONE
ABORT
consideration act
equiv
legal act

Figure 7.11: Reason Menu

The CHECK-ALL-PLANS production will direct Sherlock to attempt to classify the learner's plan by using spreading activation on the plans in the semantic network, and then to set a new goal to pick which plan the learner used.

Janey responded to the menu in Figure 7.11 with the choices indicted in Figure 7.12.

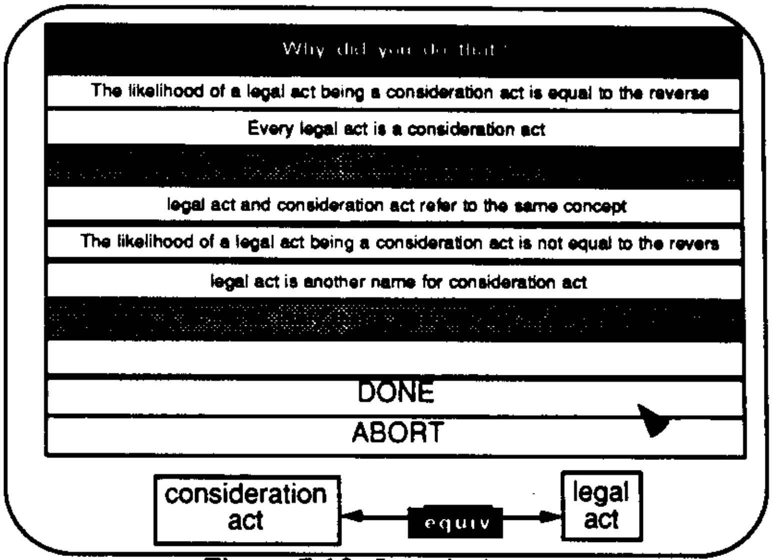

Why did you do that?
The likelihood of a legal act being a consideration act is equal to the reverse
Every legal act is a consideration act
legal act and consideration act refer to the same concept
The likelihood of a legal act being a consideration act is not equal to the revers
legal act is another name for consideration act
DONE
ABORT
consideration act
equiv
legal act

Figure 7.12: Janey's Answer

7.4. CLASSIFYING THE LEARNER'S PLAN

Sherlock assumes that she has used a rule which has the chosen facts as antecedents and the act of making an EQUIV link as a consequent. To classify this rule as one of the Sherlock's rules, Sherlock represents everything known about the learner's rule in the semantic network (Figure 7.13).

```
User chose the following reasons:
"It is not as likely that a LEGACT is a CONS-
ACT"
"Every CONS-ACT is  a LEGACT"
Initial excitation:
   RULE A
   ANTECS B
   R-T-UNLIK-F B
   R-F-IS-T B
   Creating temporary links from (R-T-UNLIK-F
R-F-IS-T) to ANTECS
   CONSEQS C
   MAKE-EQUIV C
   Creating temporary links from (MAKE-EQUIV)
to CONSEQS
```

Figure 7.13: Activating Nodes to Classify the Rules

Sherlock creates temporary s-links from the facts which she has chosen to the ANTECS role of RULE and from the action she took (MAKE-EQUIV) to the CONSEQS role of RULE. MAKE-EQUIV represents the act of making an EQUIV gm-link, the link which Janey made. Each of these nodes receives an initial activation of 100 (Figure 7.14).

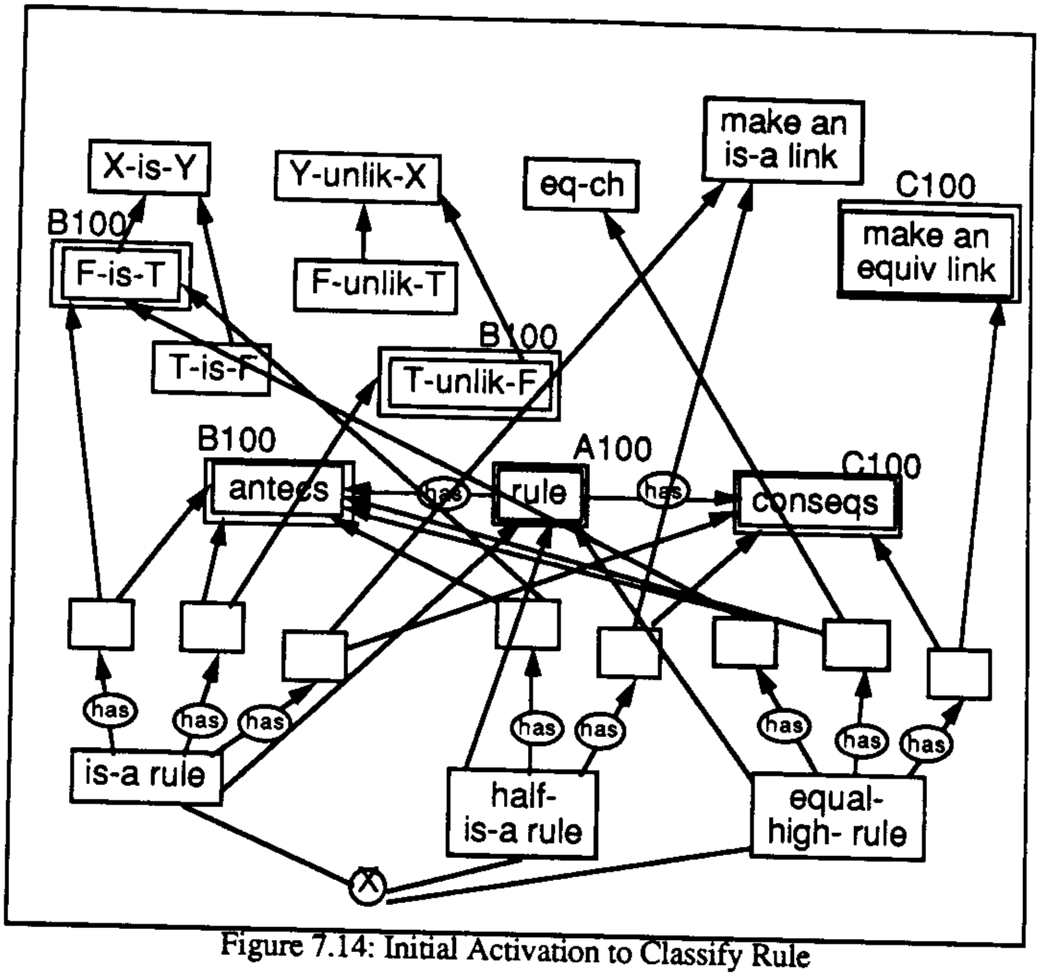


Figure 7.14: Initial Activation to Classify Rule

T-UNLIK-F is a specialization of the fact Y-UNLIK-X which represents the fact that Y-unlik-X is true where Y is the concept currently associated with the to-icon (T) and X is the concept currently associated with the from-icon (F). Similarly F-IS-T represents the fact that X-is-Y is true for the two concepts associated with the current icons.

Figure 7.15 shows a portion of the transcript for time step 1.

```
T I M E   S T E P   1
    .
    .
    .
Excitation from R-T-UNLIK-F with B
   WEAK-IS-A-T-UNLIK-F 4
   IS-A-T-UNLIK-F 4
   X-UNLIK-Y 80
Excitation from R-F-IS-T with B
   EQUAL-HIGH-F-IS-T 4
   DUB-IS-A-F-IS-T 4
   IS-A-F-IS-T 4
   HALF-IS-A-F-IS-T 4
   X-IS-Y 80
   R-F-MAYBE-T 80
Excitation from CONSEQS with C
   ACT 40
Excitation from MAKE-EQUIV with C
   EQUAL-HIGH-LINK 4
   SAME-LINK 4
   EQUAL-LINK 4
   DUB-IS-A-LINK 4
   MAKE-LINK 40
    .
    .
```

Figure 7.15: Classifying the Plan - Time Step 1

In Figure 7.15 the activation from the facts spreads to the specializations of those facts which are specializations of the ANTECS roles of rules. So, for example, the activation from R-F-IS-T reaches IS-A-F-IS-T, a specialization of R-F-IS-T which is a role of IS-A-RULE (Figure 7.16).

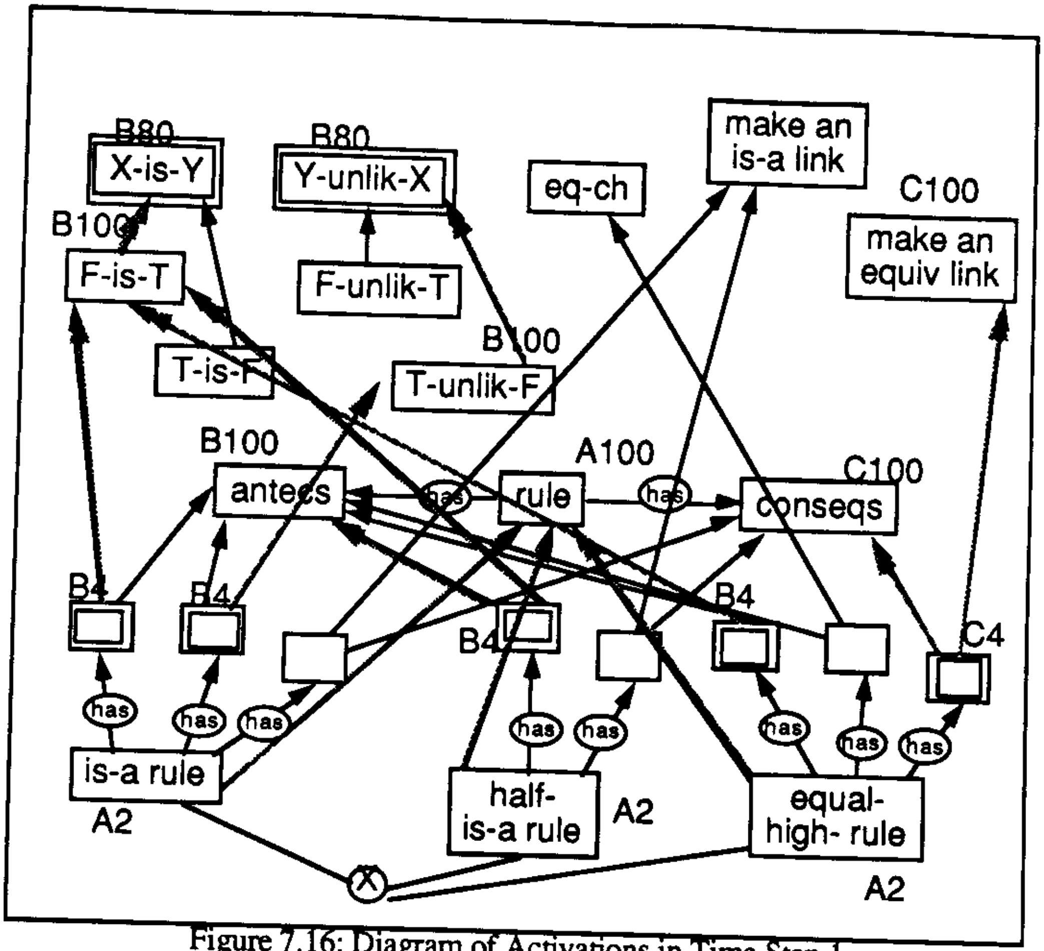


Figure 7.16: Diagram of Activations in Time Step 1

In time step 1 the activation from MAKE-EQUIV also spreads to the specializations of MAKE-EQUIV which are specializations of the CONSEQS roles of rules.

Figure 7.17 shows some of the activation which take place in time step 2.

```
TIME  STEP  2
   .
   .
Excitation from WEAK-IS-A-T-UNLIK-F with B
   WEAK-IS-A-RULE 4, mark: A
Excitation from IS-A-T-UNLIK-F with B
   IS-A-RULE 4, mark: A
Excitation from EQUAL-HIGH-F-IS-T with B
   EQUAL-HIGH-RULE 4, mark: A
Excitation from DUB-IS-A-F-IS-T with B
   DUB-IS-A-RULE 4, mark: A
Excitation from IS-A-F-IS-T with B
   IS-A-RULE 6, mark: A
Excitation from HALF-IS-A-F-IS-T with B
   HALF-IS-A-RULE 5, mark: A
   .
   .
Excitation from EQUAL-HIGH-LINK with C
   R-EQUAL-HIGH 5, mark: A
Excitation from SAME-LINK with C
   R-SAME 3, mark: A
Excitation from EQUAL-LINK with C
   R-EQUAL 3, mark: A
Excitation from DUB-IS-A-LINK with C
   R-DUB-IS-A 5, mark: A
Excitation from MAKE-LINK with C
   ACT 56
Exciting RULENESS for A
```

Figure 7.17: Classifying the Plan - Time Step 2

In time step 2 the activation reaches some of the rules from the antecedent roles which were activated in Figure 7.15. Activation also reaches some rules from the consequent roles which were activated from MAKE-EQUIV. The activation on each rule at the end of time step 2 is dependent on:

1. the weight of the s-link from the role to the rule (the activation on the role is multiplied by the weight of the s-link);
2. the number of roles supplying activation to the rule (all activations with the same marker are added together).

These activations in time step 2 are illustrated in Figure 7.18.

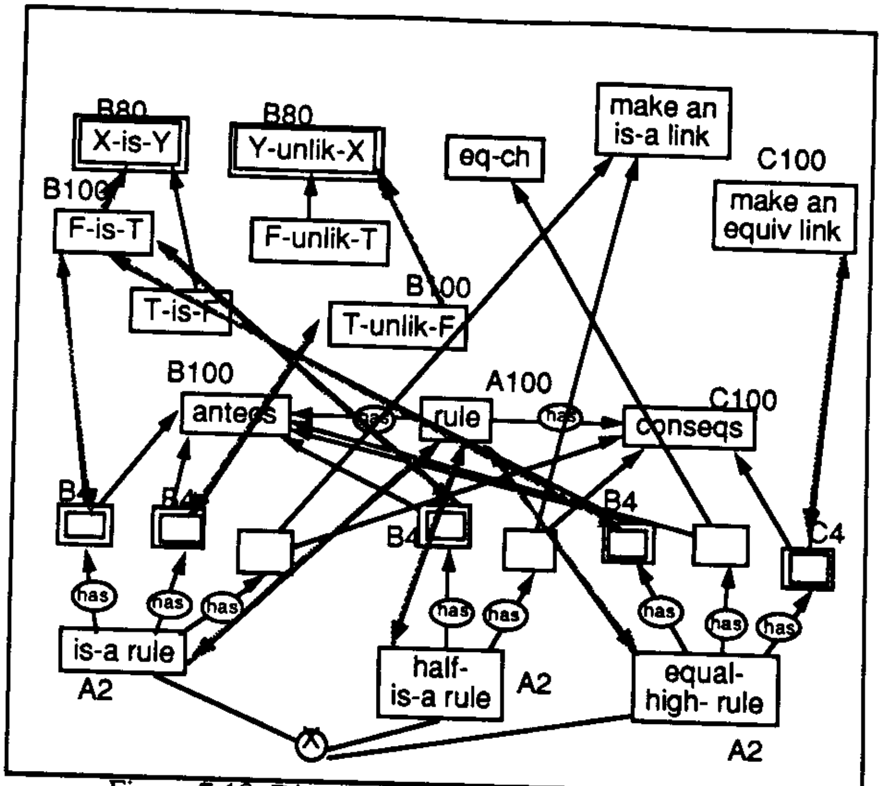


Figure 7.18: Diagram of Activations in Time Step 2

In time step 3 the inhibition node which connects all of the rules begins to adjust the activations on the rules (Figure 7.19).

```
T I M E   S T E P   3
Inhibiting from RULENESS for A
   RESULTS-RULE 0
   EXPECT-RULE 0
   PROP-LEADS-RULE 0
   LEADS-RULE 0
   PART-RULE 0
   PROP-RULE 0
   MAYBE-PART-RULE 0
   EQUAL-RULE 0
   SAME-RULE 0
   EQUAL-HIGH-RULE 25.5
   DUB-IS-A-RULE 25.5
   REV-IS-A-RULE 0
   NOT-RULE 0
   WEAK-IS-A-RULE 13.5
   IS-A-RULE 37.5
   HALF-IS-A 25.5
```

Figure 7.19: Classifying the Plan - Time Step 3

Notice in Figure 7.19 that IS-A-RULE is slightly ahead of the other rules. In the following two times steps, since no new activation reaches any of the rule nodes, the network settles on IS-A-RULE as the most likely classification of the learner's plan (Figure 7.20).

```
T I M E   S T E P   4
Inhibiting from RULENESS for A
   EQUAL-HIGH-RULE 17.5
   DUB-IS-A-RULE 17.5
   WEAK-IS-A-RULE 0
   IS-A-RULE 79.0
   HALF-IS-A-RULE 17.5
    .
    .
    .
T I M E   S T E P   5
Inhibiting from RULENESS for A
   EQUAL-HIGH-RULE 0.0
   DUB-IS-A-RULE 0.0
   IS-A-RULE 100
   HALF-IS-A-RULE 0.0
    .
    .
```

Figure 7.20: Classifying the Plan - Time Steps 4 and 5

Control now returns to the production system with the goal to pick a plan (Figure 7.21). The rule PICK-PLAN is fired at this point.

```
CYCLE   5  GOALS:  ((PICK-PLAN) (FIND-PLAN
CHECK-ALL-PLANS) (EVAL-LINK
NO-PRODS FIND-PLAN) (FEEDBACK NOT-EVALD))
PRODUCTION:  PICK-PLAN

CYCLE   6  GOALS:  ((PICK-PLAN PICK-PLAN)
(FIND-PLAN CHECK-ALL-PLANS)
(EVAL-LINK NO-PRODS FIND-PLAN) (FEEDBACK NOT-
EVALD))
PRODUCTION:  NOTHING

popping goal because couldn't find anything.

CYCLE   7  GOALS:  ((FIND-PLAN CHECK-ALL-
PLANS) (EVAL-LINK NO-PRODS
FIND-PLAN) (FEEDBACK NOT-EVALD))
PRODUCTION:  FOUND-PLAN

Subject used plan: IS-A-RULE
```

Figure 7.21: Setting *plan-used*

PICK-PLAN is shown in Figure 7.22.

```
(tproduction!
 'pick-plan
 '(
                                ;** ANTECEDENTS **
   (eq-goal? 'pick-plan)        ;the goal is to pick
                                ;  the plan the learner
                                ;  just used
   (not *plan-used*)            ;we do not know which
                                  plan the learner used
   *possible-plans*             ;we have a list of
                                ;  possible plans from
                                ;  the spreading activ
   (>
                                ;the activation on the
   (cadar *possible-plans*)     ;  strongest plan is
   *plan-threshold*)            ;  high
   (not (almost-equal?          ;there is a big diff
   (cadadr *possible-plans*)    ;  between the activ on
  (cadar *possible-plans*)))    ;  the strongest plan
   )                            ;  and the second
                                ;  strongest plan
'(
                                ;** CONSEQUENTS **
(set *plan-used*                ;decide that learner
 (caar *possible-plans*))       ;  used the strongest
  )                             ;  plan
.8)
```

Figure 7.22: Pick-plan Rule

PICK-PLAN fires because there was a clear winner in the activation of the rules. Based on PICK-PLAN Sherlock decides that the learner used an instance of the plan IS-A-RULE.

It should be noted that this classification is made despite the fact that the match is not complete; the learner did not make an IS-A gm-link which IS-A-RULE leads to. If Sherlock used a straight matching process, the learner's rule could not be classified in this case. Using spreading activation Sherlock is able to classify the learner's rule as a buggy version of a known rule. Sherlock thus has the information necessary to tutor the learner on the differences between her rule and the known rule.

7.5. EVALUATING THE LEARNER'S BELIEFS

Now that Sherlock has determined which plan it believes the learner used, it assesses the learner's fact beliefs (Figure 7.23).

```
sorting facts:

R-T-UNLIK-F  true? #T    indic? #T   match
R-ANOTHER-NAME  true? ()    indic? ()   match
R-NOT-EQ-CH  true? #T    indic? ()
R-SAME-CONC  true? ()    indic? ()   match
R-F-IS-T  true? #T    indic? #T   match
R-T-IS-F  true? ()    indic? ()   match
R-EQ-CH  true? ()    indic? ()   match
number of matches:6
possible: 7
```

Figure 7.23: Sorting Facts

The truth value of each fact (true?) was set during the activation of the concepts for each of the icons linked. Whether the learner believes a particular fact (indic?) is determined by doing a spreading activation on all of the facts that the learner indicated in the "Why did you do that?" menu and then checking to see if any activation reached that particular fact. Sherlock then compares the truth? value for each fact with the indic? value for that fact. If both are true (meaning the learner believes the fact and the fact is true) or both are false (meaning the learner does not believe the fact and the fact is false) it is a match.

7.6. PROVIDING FEEDBACK

The production system now pops the goal to evaluate the link (cycle 8 in Figure 7.24). This returns to the original goal of providing the learner with feedback. There are productions for this goal which check alternate icon interpretations if that is appropriate. The production NO-ALT-FROMS (cycle 9 in Figure 7.24) fires at this point because there are no alternate interpretations, and sets the current assessment of the learner's fact beliefs as final.

```
CYCLE    8  GOALS:   ((EVAL-LINK NO-PRODS FIND-
PLAN) (FEEDBACK NOT-EVALD))
PRODUCTION:  NOTHING

popping goal because couldn't find anything.

CYCLE    9  GOALS:   ((FEEDBACK NOT-EVALD))
PRODUCTION:  NO-ALT-FROMS
CYCLE    10  GOALS:   ((FEEDBACK NOT-EVALD NO-
ALT-FROMS))
PRODUCTION:  BAD-PLAN

CYCLE    11  GOALS:   ((TUTOR-PLAN) (FEEDBACK
NOT-EVALD NO-ALT-FROMS
BAD-PLAN))
PRODUCTION:  PLAN-TUTOR
```

Figure 7.24: Calling the Plan Tutor

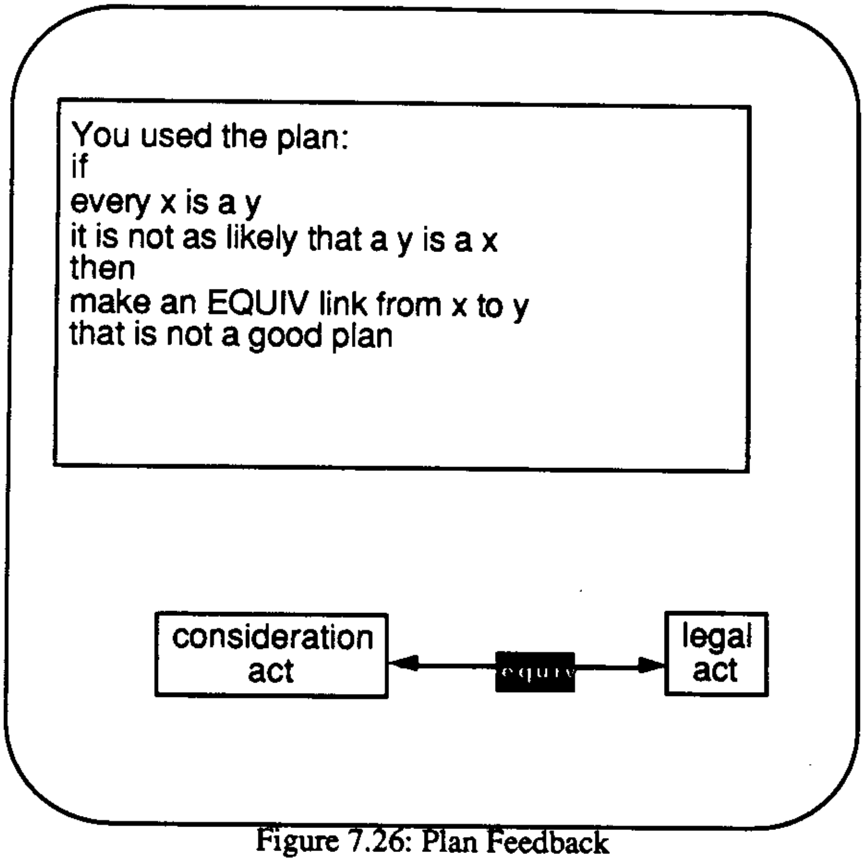
In cycle 10 (Figure 7.24) the production BAD-PLAN fires because the plan which the learner used should not have led to the gm-link which the learner made.

The contents of BAD-PLAN are shown in Figure 7.25.

```
(tproduction!
 'bad-plan
 '(                                ;** ANTECEDENTS **
   (eq-goal? 'feedback)            ;the goal is to provide
                                   ;  feedback
   *plan-used*                     ;we know the plan which
                                   ;  the learner used
   (or                             ;either:
     (<                            ;  the rule was buggy
     (rule-strength
        *plan-used*)
       0)                          ;or
     (neq?                         ;  the link made was
      (production-link             ;  was not the link the
     (get-production               ;  rule should have led
        *plan-used*))              ;  to
    *link-made*)
    )
   )
'(                                 ;** CONSEQUENTS **
  (goal! 'tutor-plan)              ;set the goal to offer
  )                                ;  tutoring regarding
                                   ;  the plan
.8)
```

Figure 7.25: Bad-Plan

The first goal to fire for the goal TUTOR-PLAN is PLAN-TUTOR (cycle 11 in Figure 7.24). PLAN-TUTOR generates the feedback shown in Figure 7.26



You used the plan:
if
every x is a y
it is not as likely that a y is a x
then
make an EQUIV link from x to y
that is not a good plan

Figure 7.26: Plan Feedback

Notice that Sherlock lists the antecedents from the plan which it believes the learner used, but the consequent refers to the gm-link that was actually made.

In Figure 7.27 the production SUGGEST-LINK is fired. The contents of SUGGEST-LINK are shown in Figure 7.28.

```
CYCLE    12   GOALS:   ((TUTOR-PLAN PLAN-TUTOR)
(FEEDBACK NOT-EVALD NO-ALT-FROMS
BAD-PLAN))
PRODUCTION:  SUGGEST-LINK

CYCLE    13   GOALS:   ((TUTOR-PLAN PLAN-TUTOR
SUGGEST-LINK) (FEEDBACK
NOT-EVALD NO-ALT-FROMS BAD-PLAN))
PRODUCTION:  NOTHING

popping goal because couldn't find anything.
```

Figure 7.27: Suggesting the Correct GM-Link

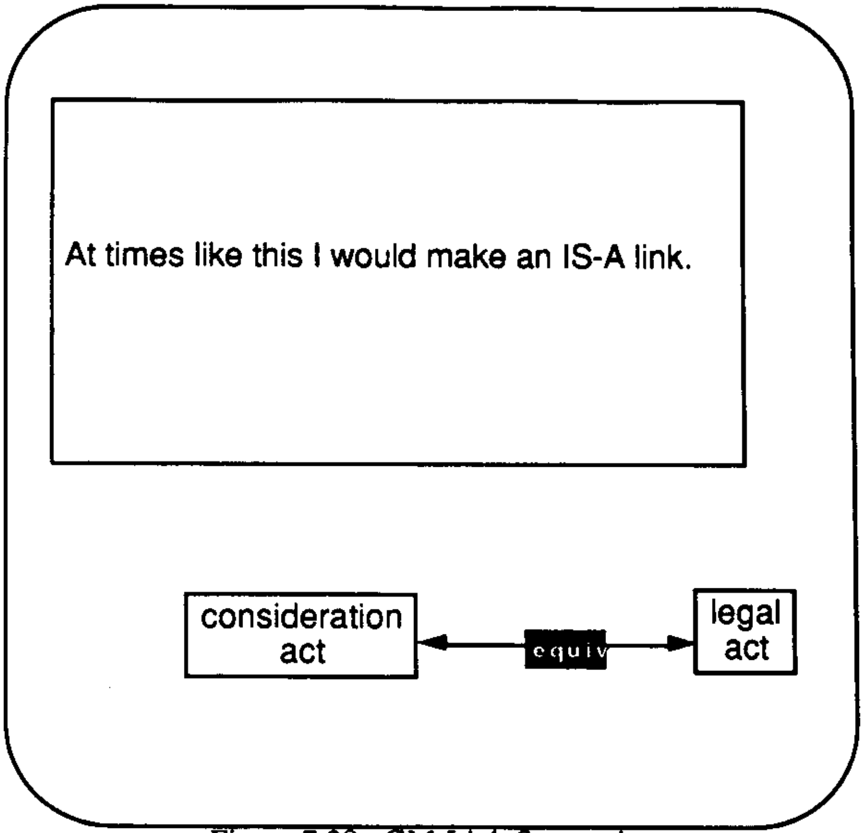
```
(tproduction!
 'suggest-link
 '(                              ;** ANTECEDENTS **
   (eq-goal? 'tutor-plan)        ;the goal is to provide
                                 ;  plan tutoring
   *plan-used*                   ;we know the plan used
   (>                            ;the plan is good
   (rule-strength
     *plan-used*)
    0)
   (neq?                         ;the plan should have
  (production-link               ;  led to a different
     (get-production             ;  gm-link
     *plan-used*))
*link-made*)
      )
'(                               ;** CONSEQUENTS **
   (set tmp                      ;set TMP to the gm-link
(production-link                 ;  the plan should have
(get-production                  ;  led to
    *plan-used*)) )
   (output-msg
'(t "*****************" t        ;suggest that the
 "At times like this             ;  learner use TMP
I would make a "                 ;  in the future
     tmp  " link."  t) )
   (run-menu 'ok-menu)
   (hide-menu 'ok-menu)
  )
 .6
 )
```

Figure 7.28: Suggest-Link

Suggest-link is fired when considering plan feedback if a plan was found and one of two conditions is true:

1. the plan found was buggy;
2. the learner used the wrong gm-link for the plan which was found.

Suggest-link generates the feedback shown in Figure 7.29.



At times like this I would make an IS-A link.

Figure 7.29: GM-Link Suggestion

The production BAD-TEXT sets a goal to consider whether tutoring of the text is appropriate (cycle 14 in Figure 7.30). BAD-TEXT will fire whenever there is not a perfect match on the facts. No productions match for this goal, however, indicating that tutoring of the text is not appropriate here.

```
CYCLE    14   GOALS:   ((FEEDBACK NOT-EVALD NO-
ALT-FROMS BAD-PLAN))
PRODUCTION:  BAD-TEXT

CYCLE    15   GOALS:   ((TUTOR-TEXT) (FEEDBACK
NOT-EVALD NO-ALT-FROMS
BAD-PLAN BAD-TEXT))
PRODUCTION:  NOTHING

popping goal because couldn't find anything.
```

Figure 7.30: Considering Whether to Tutor the Text

Tutoring of the text is appropriate if one or both of the following conditions is true:

1. there are false facts which the learner believes which led to a bad plan;
2. there are true facts which the learner does not believe which would have led to the plan Sherlock chose.

Since Janey picked the two facts which would have led to the plan Sherlock chose, neither of these conditions is true.

Because there were some problems with this link, the learner is now asked if she wants to remove it (Figure 7.31). She answers "Yes," and the link is removed (Figure 7.32).

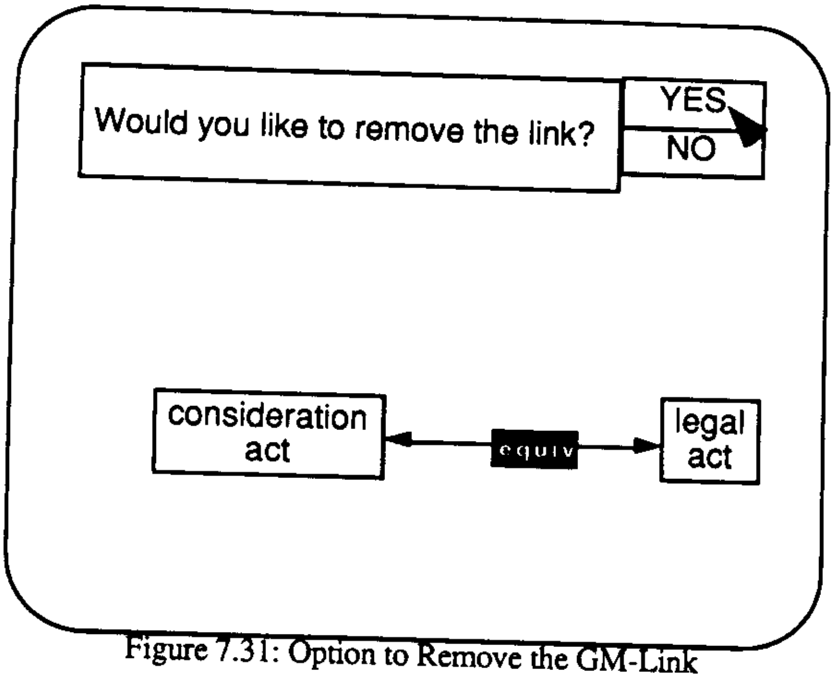


Figure 7.31: Option to Remove the GM-Link

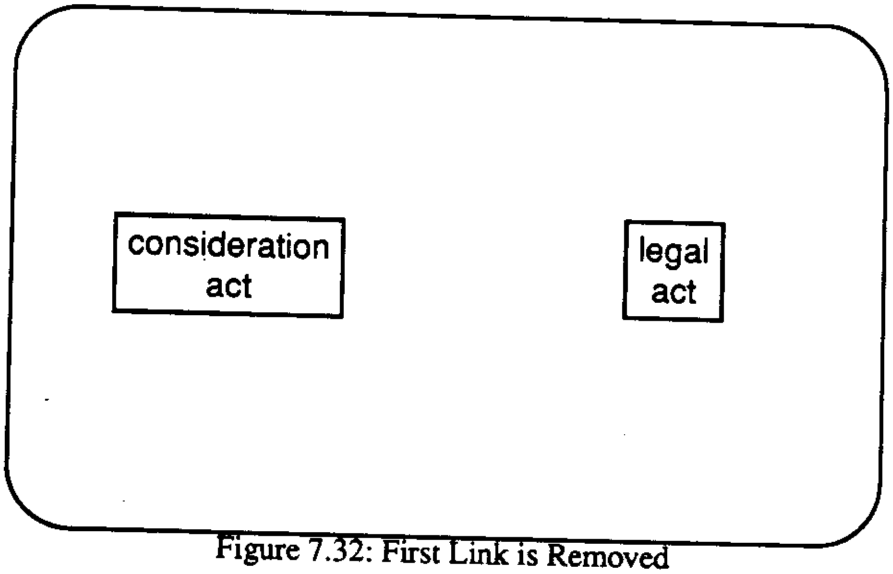


Figure 7.32: First Link is Removed

At this point the top level goal of providing feedback has been satisfied, so the production system stops and waits for the learner to make another gm-link.

7.7. SECOND ATTEMPT

Janey now makes an IS-A link between the same two icons (Figure 7.33).

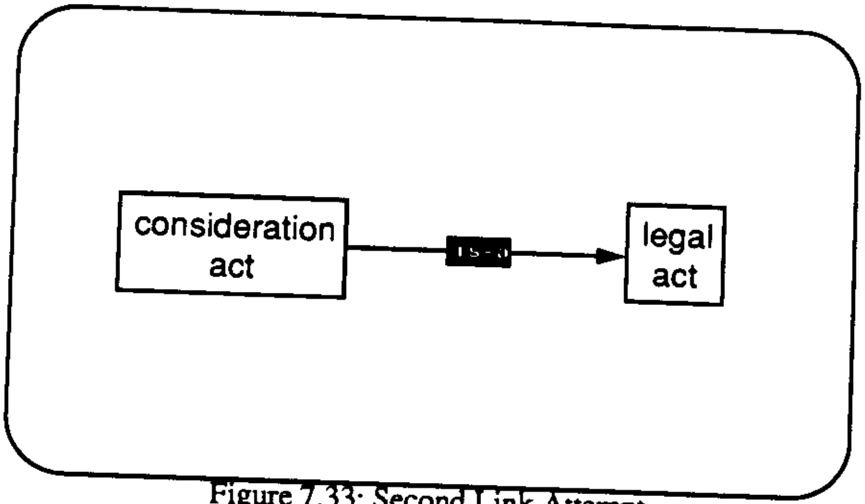
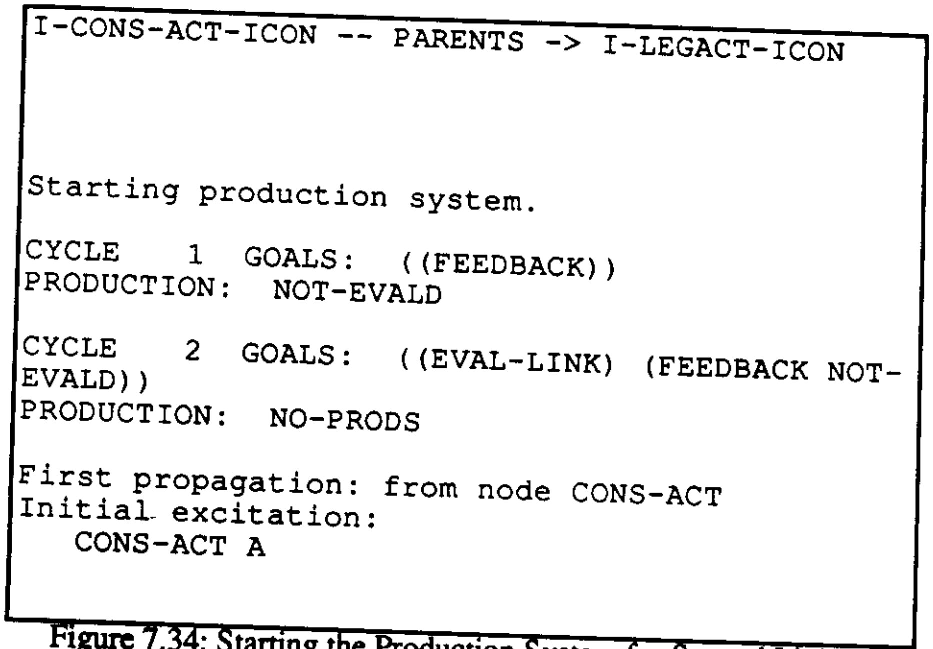


Figure 7.33: Second Link Attempt

The production system begins again, as it did for the first link (Figure 7.34).



```
I-CONS-ACT-ICON -- PARENTS -> I-LEGACT-ICON



Starting production system.

CYCLE    1  GOALS:   ((FEEDBACK))
PRODUCTION:  NOT-EVALD

CYCLE    2  GOALS:   ((EVAL-LINK) (FEEDBACK NOT-
EVALD))
PRODUCTION:  NO-PRODS

First propagation: from node CONS-ACT
Initial excitation:
   CONS-ACT A
```

Figure 7.34: Starting the Production System for Second Link

Once again Sherlock begins by doing a spreading activation on the concepts associated with the icons. In this case the spreading activation is redundant since these are the same two concepts Sherlock just processed. But since Sherlock treats each link independently it begins from scratch.

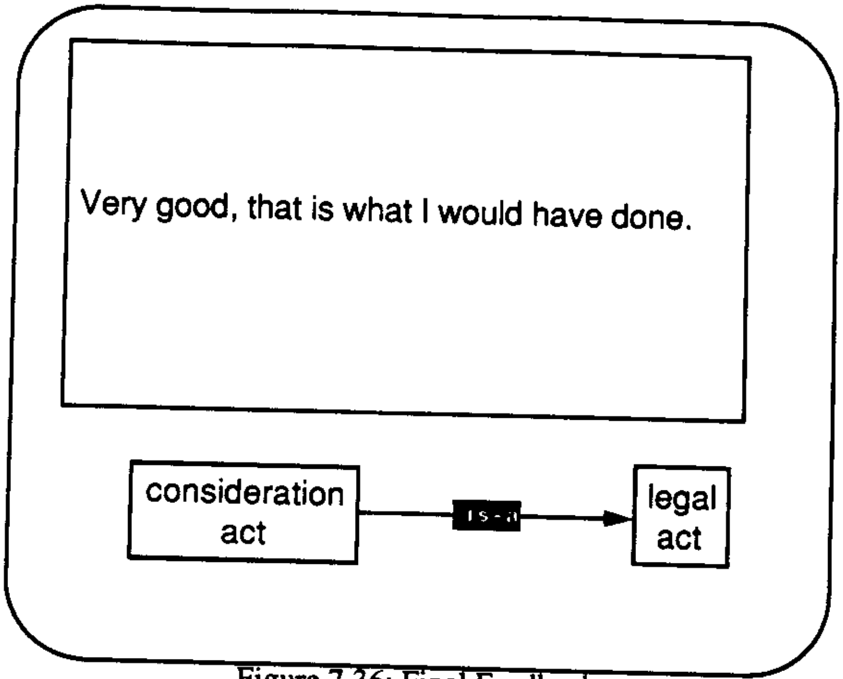
The results of the activation are the same as in Figure 7.7; Sherlock concludes that it would have made an IS-A gm-link between the two icons. The difference this time is that that is the kind of gm-link which the learner made, so the production CHECK-BEST fires in cycle 3 (Figure 7.35).

```
CYCLE    3   GOALS:   ((EVAL-LINK NO-PRODS)
(FEEDBACK NOT-EVALD))
PRODUCTION:  CHECK-BEST

That is what S would have done

CYCLE    4   GOALS:   ((CHANGE-NET) (FEEDBACK
NOT-EVALD))
PRODUCTION:  NOTHING

popping goal because couldn't find anything.

CYCLE    5   GOALS:   ((FEEDBACK NOT-EVALD))
PRODUCTION:  BEST-OK
```

Figure 7.35: End of Evaluation for Second GM-Link

The production CHECK-BEST (contained in Figure 4.11) fires because the gm-link which the learner made matches the gm-link which Sherlock would have made. This production directs Sherlock to pop the goal of evaluating the link, since it concludes that the learner's action is appropriate. The goal is set to consider changing the semantic network No change in the network is appropriate so this goal is also popped (cycle 4 in Figure 7.35).

Finally the productions BEST-OK fires for the evaluation goal, generating the feedback in Figure 7.36.



Very good, that is what I would have done.

Figure 7.36: Final Feedback

7.8. IMPLEMENTATION OVERVIEW

Sherlock currently works only with the consideration text and, to a limited extent, with the birds and fish domain used for training.

The representation for the consideration text provides the learner with 26 icons. There are 150 nodes in the semantic network to represent the consideration text and background. There are 333 s-links connecting these nodes.

The representation for the birds and fish domain provides the learner with 8 icons. The representation for the birds and fish domain relies on some of the nodes and s-links used for the consideration text. In addition there are 28 nodes and 63 s-links added specifically for this domain.

There are 16 rules in the production system for making gm-links. The representation of graphic mapping rules in the semantic network uses 115 nodes connected by 216 s-links.

There are 40 rules in the production system for directing Sherlock's tutoring strategy.

Sherlock uses approximately 3,000 lines of Lisp code in addition to the definitions of the productions, nodes and s-links. Sherlock also uses Gasser's semantic network package which comprises approximately 1,300 lines of code.

7.9. SUMMARY

Sherlock's strategy for evaluating a link and providing feedback is represented in the production system. The tutoring strategy can be summarized as follows:

1. determine the relationship between the two icons which were linked by spreading activation from the concepts associated with each of the icons;
2. determine whether Sherlock would have made the same gm-link by determining whether the production system would lead to the same gm-link given the current activations in the semantic network;
3. if not, ask the learner why he made the link;
4. use the learner's answer to classify the learner's plan as an instance of one of the known plans;
5. use the learner's answer to determine whether the learner is interpreting the icons differently from Sherlock by determining if there is a mismatch between the learner's beliefs and Sherlock's beliefs, and if so evaluate again using the new interpretation;
6. provide appropriate feedback based on;
    a. whether Sherlock would have made the same gm-link;
    b. the plan which Sherlock believes the learner was using;
    c. the mismatch between the learner's beliefs and Sherlock's beliefs.

Chapter 8 will present the results of an evaluation study designed to determine how effective this strategy is in accurately diagnosing a learner's cognitions.

Chapter 8

Evaluation of Sherlock

8.0. INTRODUCTION

The topic of evaluation in artificial intelligence (AI) research is just beginning to receive attention. DARPA has supported research at UCLA focussed on exploring evaluation approaches to AI systems (see for example Baker & Lindheim, 1988; Baker, Lindheim, & Skrzypek, 1988; Dyer & Read, 1988; Skrzypek, Mesrobian, & Gunger 1988). Others in the AI community are also beginning to describe paradigms for evaluation (Slagle & Wick, 1988; Rettig & Bates, 1988). Slagle and Wick describe evaluation efforts measuring the effectiveness of an expert system application to solve particular problems. Rettig and Bates similarly discuss criteria for choosing off-the-shelf natural language systems for particular applications.

However, the topic is not without controversy. An article in the latest issue of *AI Magazine* (Cohen & Howe, 1988), for example, argues that evaluation as it is understood in the social sciences is inappropriate to AI research.

Part of the controversy may be semantic and hinges on the differences between research and development processes. The goal of research is to develop or demonstrate new knowledge. Research is an attempt to define and test a theory. The outcome of research is intended to be generalizable with a major emphasis on understanding a phenomenon. While understanding a phenomenon might lead to a more effective solution to a real-world problem, immediate application is not the goal of research.

The goal of development is to solve some real-world problem. A real-world problem is an existing situation that the developers want to change (Cronbach & Suppes; 1969). Examples of development goals are to design a program to teach children to read, or to develop a new database to meet someone's information need. Evaluation is a part of the development process. The purpose of evaluation is to assess the effects of the developed intervention.

The semantic problem occurs when AI research is characterized by phrases such as, "...a tutoring system which helps novice programmers to learn to program" (Johnson & Soloway, 1983). At first reading, this statement implies that Johnson and Soloway are going to build a system designed to solve a problem; teaching novices to program. In fact, Baker (Baker et. al, 1985) was only able to make sense of Johnson and Soloway's activities by recognizing that the development goal—the tutoring system—was really only intended to constrain the problem space by defining a theoretical problem.

To confuse matters further, AI research usually leads to building something, a computer program. The existence of an artifact, rather than just a theory, enhances the perception that the goal is a product to solve the real-world problem. The computer program, however, may be intended only as a prototype to demonstrate the theory, not solve the real-world problem.

This confusion between development and research goals is reflected in the Cohen and Howe article mentioned above. Cohen and Howe propose evaluation as a process to assess the quality of AI research not development. Their proposal consists of criteria for each phase of AI research, assessing such attributes as potential contribution to existing knowledge and generalizability of findings. What they call "evaluation" processes are standards common to all science-based fields for assessing quality (see, for example Kaplan, 1964).

The work on Sherlock can be viewed from both research and development perspectives. The long term goal of Sherlock is to teach people to be better learners. Thus, at some point an evaluation of Sherlock's ability to achieve this goal will be appropriate. But the present implementation of Sherlock is only a prototype, intended to demonstrate a solution to one theoretical problem which must first be solved. That theoretical problem is the need to understand a learner's current misunderstandings before attempting to correct them.

Figure 8.1 illustrates the relationship between the research goal (Sherlock's ability to diagnose a learner's misunderstandings) and the development goal (possible outcomes of Sherlock's use as a tutor).

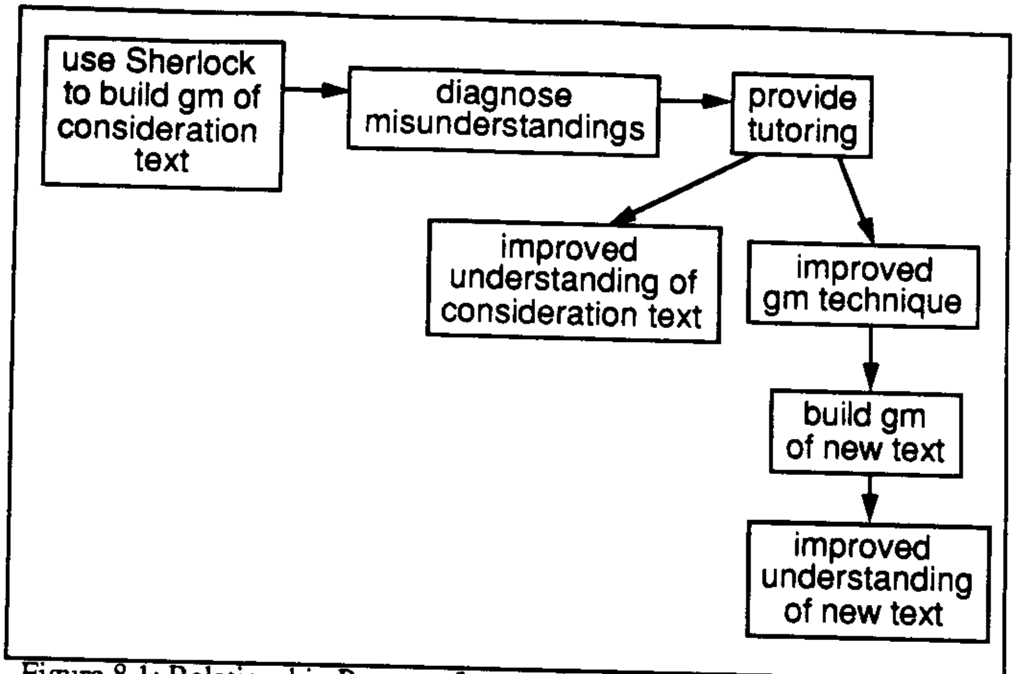

use Sherlock to build gm of consideration text
diagnose misunderstandings
provide tutoring
improved understanding of consideration text
improved gm technique
build gm of new text
improved understanding of new text

Figure 8.1: Relationship Between Intervening Variables and Final Outcomes

There are two desired outcomes from an interaction with Sherlock:

1. improved understanding of the consideration text;
2. improved ability to use graphic mapping as a learning strategy.

The first outcome could be evaluated by testing the learners' understanding of contract law following a session with Sherlock, perhaps compared to a control group which studied the text using an alternate study technique or graphic mapping without Sherlock.

The second outcome could be evaluated by testing the learners' understanding of a new text which was studied using graphic mapping without Sherlock, compared to a control group which studied the new text, but had no exposure to Sherlock.

Both of these outcomes, however, are contingent on Sherlock's ability to provide effective tutoring. Since the tutoring component of Sherlock has not been fully implemented at this time, an outcome evaluation is inappropriate. Sherlock's ability to provide effective tutoring is contingent on its ability to correctly diagnose the learner's misunderstandings. Since the diagnosis component has been fully implemented, this intervening variable can be evaluated.

To evaluate the accuracy of Sherlock's diagnosis a study was conducted in the summer of 1988. Four subjects were video taped during a two-hour session using Sherlock. This study built upon the earlier empirical work discussed in Chapter 2.

Recall that in the first pilot study, subjects were observed building graphic maps without think-aloud protocols. The goal of the first study was to explore the possibility of inferring a learner's cognitive structures based on the graphic map he produced. These inferences were to be confirmed using a posttest on the material being mapped.

The results of the first study indicated that it was impossible to infer a learner's cognitive structures based on a finished graphic map. The study did help define the three aspects of a learner's cognitive structure which had to be modelled in order to understand graphic mapping

In the second pilot study, subjects were again observed building graphic maps. This time they were asked to think aloud during the process, and explain the reasons behind each gm-link made. The goal of the second study was to define a cognitive model for the task of constructing a graphic map. Sherlock was built based on this model.

In the first two studies, subjects used Sherlock only as a graphic interface to aid them in constructing a graphic map. There was no interaction with Sherlock as an intelligent tutoring system. In this third study, subjects were again asked to think aloud while building a graphic map. At this point Sherlock was capable of analyzing the links that the subject made and interacting with the subject regarding his reason's for making the link. The goal was to compare Sherlock's analysis with the experimenter's interpretations, based on the think-aloud protocols.

8.1. METHODOLOGY

8.1.1. SUBJECTS

The four subjects used in this final study were chosen to represent a range of ages, educational backgrounds, and computer experience. None of the subjects, however, had any real knowledge of contract law. The following is a brief description of each subject;

Becky	A 23 year old college graduate with no previous computer experience.
Janey	A 31 year old college graduate with a teaching credential. She had a little computer experience.
Jennifer	A 20 year old undergraduate with a lot of experience using computers.
Stan	A 40 year old Ph.D. with a great deal of experience using and programming computers.

8.1.2. SESSIONS

Subjects were given the same training and text as that provided in the first and second pilot study. After the subject made a gm-link, Sherlock would determine whether the link was appropriate. If the link was determined to be inappropriate, Sherlock would ascertain the confusion which led to the link. Finally, Sherlock gave some feedback based on its analysis of the link.

As in the second study, subjects were instructed to think aloud while building their map. This protocol now included their thoughts while answering any questions the computer would ask of them. In addition they were asked to react to any feedback Sherlock provided. For each feedback item two things were ascertained:

1. Did the feedback accurately reflect what the learner believed?
2. If the feedback was an accurate reflection, was the subject willing to accept it?

> You seem to believe
> that all contracts are written agreements
> I do not agree

Figure 8.2: Sample Feedback

To illustrate, if the computer provided the feedback in Figure 8.2, the experimenter asked the subject the following questions:

1. "Did you believe that all contracts are written agreements?"
2. "Are you willing to accept Sherlock's correction, or are you going to stick to your belief that all contracts are written agreements?"

Subjects were given 90 minutes to make as many links as they could.

8.1.3. ANALYSIS

As a first step, I inferred the subjects' beliefs looking at their actions and words. Inferences were made without looking at Sherlock's feedback or the transcript of Sherlock's processing. There is no claim that the beliefs inferred actually reflect the subjects' beliefs; only that they represent one human tutor's best guess as to what the subjects believe. In this chapter, Sherlock's diagnosis will be referred to as correct if it agrees with these inferred beliefs.

Next, I compared the inferred belief to Sherlock's diagnosis. First I determined if the feedback provided after the link was consistent with the inferred beliefs. If the feedback was not consistent, a transcript of Sherlock's processing of the link was examined. An attempt was made to determine why Sherlock's diagnosis did not match the beliefs which had been attributed to the subject. In some instances the diagnosis had actually matched, but the feedback to the subject was either wrong or non-existent. Since the goal at this point is to evaluate the program's diagnostic ability, these were counted as matches.

As a final step, I looked at the impact of the feedback as a separate issue.

Annotated transcripts for each of the four subjects are included in Appendix C.

8.2. RESULTS

A total of 70 links were made by the subjects (Table 8.1). Of these, 11 were aborted before Sherlock's final analysis. These 11 links were aborted because the subject decided that he or she was doing something wrong. Included in the analysis are the 59 links that the subjects allowed Sherlock to analyze.

	completed	aborted	total
Becky	14	3	17
Janey	10	0	10
Jennifer	19	5	24
Stan	16	3	19
Total	59	11	70

Table 8.1: Breakdown of Links

For each completed link, four aspects of Sherlock's evaluation were analyzed:

1. Sherlock's evaluation of the link as a whole;
2. Sherlock's evaluation of the subject's knowledge of the text;
3. Sherlock's evaluation of the subject's use of the graphic mapping strategy;
4. Sherlock's evaluation of the subject's interpretation of the icons.

For each of these four aspects, links were placed in one of four categories:

1. Sherlock correctly determined that the subject was correct;
2. Sherlock incorrectly determined that something correct was wrong;
3. Sherlock correctly determined that the subject was wrong;
4. Sherlock incorrectly determined that something wrong was correct.

For each of the four aspects, an additional question was answered for any links which were placed in the third category:

> Was Sherlock able to correctly diagnose why the subject was wrong?

8.2.1. EVALUATING SHERLOCK'S DIAGNOSIS

8.2.1.1. OVERALL ACCURACY

Of the 59 completed links, Sherlock determined that 33 (55.93%) were appropriate. There were actually 32 appropriate links. I classified a link as appropriate if it reflected a correct understanding of the text and the graphic mapping strategy. Of the 26 links that Sherlock labelled as wrong, 25 were actually wrong. Thus Sherlock's determination of whether a link was right or wrong was correct for 96.61% of the links (Table 8.2).

For the 25 links which Sherlock correctly determined were wrong, the diagnosis matched only 56% of the time.

This percentage of match means that Sherlock was able to correctly determine that a link was right or wrong and provide a correct diagnosis for a total of 77.97% of the links.

	# of links	% of total
good link correctly identified	32	54.24
good link incorrectly seen as bad	1	1.69
bad link correctly identified		
reason found	14	23.73
reason not found	11	18.64
bad link incorrectly seen as good	1	1.69
Total	59	100

Table 8.2: Overall Accuracy of Links

8.2.1.2. ACCURACY OF PLAN DIAGNOSIS

Sherlock sees each link as being the result of just one plan, and that plan as being either good or bad. Sherlock's decision on whether the plan used in making a link was good or bad was correct for 83% of the links. This figure is lower than the corresponding rate for links as a whole due to the much higher false negative rate: 14% versus 2% (see Table 8.3). Sherlock's accuracy rate for correctly determining if the plan was good or bad and diagnosing the problem was 81%.

	# of links	% of total
good plan correctly identified	43	72.88
good plan incorrectly seen as bad	2	3.39
bad plan correctly identified		
reason found	5	8.47
reason not found	1	1.69
bad plan incorrectly seen as good	8	13.56
Total	59	100.00

Table 8.3: Accuracy of Plan Diagnosis

8.2.1.3. ACCURACY OF FACT DIAGNOSIS

Each link is the result of a number of declarative facts which the subject believes. For the sake of analysis the whole set of fact beliefs pertaining to a particular link is considered to be either good or bad. The set is considered good if there are no relevant false beliefs that the subject believes and no relevant true beliefs that the subject does not believe. Relevant here means that the fact affected the link made. A false belief is only relevant if believing it led the subject to make a bad link. A true belief is only relevant if believing it would have led the subject to make a better link.

	# of links	% of total
good facts correctly identified	37	62.71
good facts incorrectly seen as bad	7	11.86
bad facts correctly identified	13	22.03
bad facts not noted	2	3.39
Total	59	100.00

Table 8.4: Accuracy of Fact Diagnosis

According to Table 8.4, Sherlock's assessment of whether a set of fact beliefs was good or bad was accurate for 84.7% of the links. Since Sherlock was given no credit for determining that facts were part of the problem, but not identifying which facts were the problem, there is no separate statistic as to why a set of facts is bad.

It might be noted that plans were bad about as often as fact sets (24% and 25% respectively).

8.2.1.4. ACCURACY OF INTERPRETING ICONS

For analysis purposes, links were categorized as resulting from the use of a non-default icon interpretation if one or both of the icons linked had been interpreted by the subject in a non-default manner. In fact, no link was made between two icons which were both interpreted in a non-default manner. Of the 59 links made, less than 20% involved the use of a non-default interpretation. Of these, only 54% were correctly diagnosed by Sherlock. Problems in icon interpretations only accounted for diagnosis problems in 8.5% of the links made.

	# of links	% of total
non-default used and found	6	10.17
non-default used but not found	5	8.47
default used and found	47	79.66
default used, non-default found	1	1.69
Total	59	100.00

Table 8.5: Accuracy of Interpreting Icons

According to Table 8.5, there were five instances in which a non-default interpretation of an icon was used. These include;

(Janey, link #4)	interpreting "condition" as CONTRACT:MTRANS1 -- because MTRANS1 states the condition part (terms) of the contract;
(Stan, link #6)	interpreting "contract" as CONTRACT:RESULT -- this interpretation should have been anticipated;
(Stan, link #9, 10, & 13)	interpreting "consideration forbearance" as referring to CONSIDERATION or FORBEARANCE -- believed that they were equivalent.

Each of these three non-default interpretations were unanticipated. That is, they involved interpretations which Sherlock had not been programmed to consider. Since Sherlock currently has no ability to consider unanticipated alternate interpretations, it is not surprising that Sherlock did not find this.

8.2.1.5. Is Sherlock Accurate Enough?

Accuracy is important in a tutoring system; wrong feedback can have negative impact on learning. Telling a learner he is wrong when he is right can lead the learner to a loss of self confidence. Telling a learner he is right when he is wrong can reinforce incorrect beliefs.

Sherlock was able to correctly determine if a link was right or wrong over 96% of the time. The question is: Is this accurate enough? One way to answer this is to compare Sherlock to other forms of instruction.

A human tutor will certainly be more accurate that any machine-based tutor. In terms of automated tutoring, the highest accuracy can be achieved with multiple-choice CAI. If the questions and distractor choices in a multiple-choice format are carefully written, close to 100% accuracy can be reached. The problem is that multiple-choice responses are the least indicative of what a learner understands.

Open-ended responses can be much more useful for ascertaining what a learner understands. But if even single-word responses are allowed in CAI the accuracy dramatically drops because it is difficult to anticipate every potentially correct response.

Sherlock offers a compromise in that there is a finite range of responses possible. That range, however, is quite large. With 25 icons there are 3600 possible learner actions[1]. It would be possible, but very difficult, to record in advance which of the 3600 possible actions is correct. Add the qualifier that certain responses are only acceptable if the learner is using particular interpretations, and it becomes impossible to completely anticipate correct responses.

Considering the range of responses allowed, Sherlock's performance is certainly comparable to any automated tutoring approach that currently exists.

---

[1] Each learner action is a possible combination of from-icon, gm-link type, and to-icon. Thus there are 25 (number of possible from-icons) * 6 (number of gm-link types) * 24 (number of possible to-icons) possible combinations.

Sherlock was less successful at determining why an action was wrong. Comparing this ability to other forms of tutoring is more difficult. In automated tutoring, only intelligent tutoring systems (ITS) make any claim for diagnosing the cause of a misunderstanding. And in ITS research, evaluations have only been done on a system's ability to identify errors. No system has been evaluated for its accuracy in identifying the cause of an error.

8.2.1.6. IMPLICATIONS FOR SHERLOCK'S MODEL

To assess the implications of these statistics for Sherlock's model we look at the causes for these errors in diagnosis.

As was indicated above, there were only two instances in which Sherlock made an error in determining whether a learner's action was right or wrong.

In the first instance the learner was right, but Sherlock thought she was wrong (Janey, link #4). This error is due to the fact that the learner is using an icon interpretation which had not been anticipated — the use of "condition" to represent conditions of a contract. It is understandable that Sherlock labeled this gm-link as wrong, since it is questionable whether this is an appropriate interpretation for the icon "condition." It is clear from the think-aloud protocols, however, that the learner's action reflected a correct understanding of the text and the graphic mapping strategy which was the definition of a correct gm-link.

In the second instance the learner was wrong, but Sherlock thought he was right (Stan, link #9). This error was due to the fact that the learner made the right move for the wrong reasons. When a student does what the tutor would have done, the tutor assumes that there is no problem. In Sherlock's case if a combination of strategy, facts and interpretations can be found which would have led to a link, that link is considered correct. But it is, of course, possible that the learner used a different and wrong combination to make the link.

In this instance Stan used a unique interpretation for one of the icons and accidentally made the link in the opposite direction from the one he intended. Sherlock would have made the same link, in the direction the link was actually made, using the default interpretation for the icon. Therefore Sherlock considered the link correct despite the fact that the link was actually the result of two learner problems

The only way to reduce the chance of diagnosing a combination of mistakes as correct is to ask the learner to explain his reasoning for every link, even if the link seems completely clear. Judging from the negative reaction to the "Why did you do that?" menu (i.e.;,"Oh no, not again") that would seem undesirable.

There is a trade-off in tutoring between insuring an accurate learner model but boring the learner, or allowing for more mistakes in diagnosis but increasing learner motivation. Since wrong factors led to a correct link in only one case out of 59 links, it seems that Sherlock can permit this increased chance of errors rather than alienating learners by requiring that they justify every link.

Most of the errors in diagnosing the reason a wrong link was wrong were due to problems discussed in previous chapters. These include:

1. difficulty classifying the rule the subject used:
    a. unable to choose among equally viable options;
    b. settling on a weak rule without sufficient evidence;
2. inability to consider unanticipated icon interpretations;
3. difficulty determining what facts the subject believes because of how facts are defined.

The first two problems listed above reflect problems a human tutor also has in trying to diagnose the reasons for a learner's error. Thus, while they represent short-comings in Sherlock's model, they are at least consistent with human performance.

The third problem, however, is much more of a problem for Sherlock than it would be for a human tutor. While learners will always have trouble being conscious of the reasons they chose a particular action, this is one case where the lack of a natural language interface in Sherlock is a handicap. The human tutor can be more flexible in how facts are presented to the learner, and in interpreting the learner's answers. And of course the human tutor has the option of simply asking the learner "Why did you take that action?" and then understanding the learner's natural language response.

8.2.2. EVALUATING SHERLOCK'S FEEDBACK

As was indicated, the quality of Sherlock's feedback to the learner was not considered in the above evaluation. In this section I examine feedback as a separate issue and discuss feedback problems which were uncovered in this evaluation.

A feedback problem is defined as a case in which the proper diagnosis was made, but not clearly communicated to the learner. There were two types of feedback problems in this study. The first involved a lack of feedback and was due to two bugs in the tutoring strategies. The second involved confusing feedback and was a result of the approach used in Sherlock.

8.2.2.1. MISSING FEEDBACK

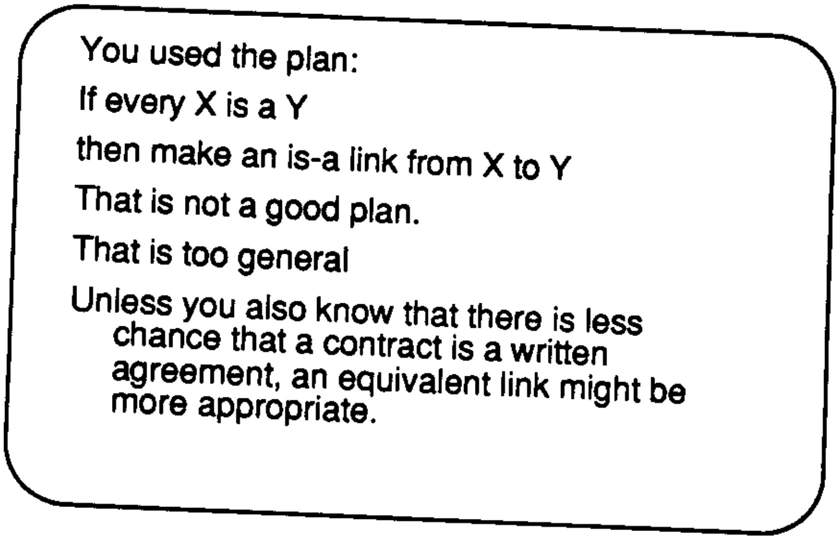
The missing feedback bug occurred when a link had been initially questioned, and then determined to be appropriate. This happened when the answers to the "Why did you do that?" question led Sherlock to conclude that the subject had used an alternate interpretation of one of the icons. If that interpretation meant that the facts were correct, and the plan had already been found to be good, Sherlock appropriately determined that the link was good.

Unfortunately a bug in the tutoring strategy productions prevented Sherlock from communicating this to the learner (i.e., Becky link, #11). The solution is, of course, trivial (i.e., fix the bug in Sherlock).

8.2.2.2. MISLEADING FEEDBACK WHEN PLANS & FACTS ARE TREATED SEPARATELY

A second feedback problem is tied to Sherlock's approach to diagnosis. Although the original diagnosis is made by looking at the interaction between plans and facts, feedback for plans and facts are each generated separately. Feedback for facts does take into account the plan that the learner used and the plan that Sherlock would have used. The plans are considered because there are a potentially large number of facts which could be commented on. To decide which facts are relevant, Sherlock looks only at the wrong facts which led to a bad learner link, or the missing facts which might have led to the plan Sherlock used.

When considering plan feedback, however, the facts are not considered because there is only one plan. In addition, this plan is bad or good regardless of whether the facts which supported it are correct or not. Unfortunately commenting on the plan without considering facts can be confusing to the learner. A good example of the potential confusion occurred in response to Jennifer's link #13 (Figure 8.3)



You used the plan:

If every X is a Y

then make an is-a link from X to Y

That is not a good plan.

That is too general

Unless you also know that there is less chance that a contract is a written agreement, an equivalent link might be more appropriate.

Figure 8.3: Confusing Plan Feedback

This feedback was chosen because Sherlock decided that she has used a buggy plan. This is part of Sherlock's strategy. It was an accurate diagnosis of the plan which was used. The second part of the feedback is generated by Sherlock's knowledge of why the plan is bad. To make it more understandable, this part of the feedback is applied specifically to the icons which had been linked, instead of using variables.

This feedback implies that knowing how likely it is that a contract is a written agreement would help you decide whether an is-a link or an equiv is more appropriate. This would only be true if every written agreement were a contract, and that is false. Actually neither IS-A or EQUIV are appropriate for these icons.

Everything in this feedback is true. And following this, Sherlock does point out that the fact "Every written agreement is a contract" is false. But at this point it appears that Sherlock is agreeing that every agreement is a contract and suggesting that an EQUIV link be made.

The problem would be minimized if the plan feedback used only variables. Thus, there is less suggestion that the feedback applies to the facts.

8.2.2.3. EFFECTIVENESS OF FEEDBACK

As mentioned above, subjects were asked questions regarding the effect of the feedback on them. Although the present feedback was not intended as effective intervention, it is interesting to note these results. With very few exceptions subjects did not believe Sherlock's diagnosis, and thus did not change their beliefs as a result of the feedback. This seemed to be due to several factors:

1. The diagnosis was incorrect almost half the time. Even though Sherlock had been correct in rejecting a link, if the reasons given for rejecting the link were wrong, the subject had trouble accepting the overall rejection.
2. Subjects had been told that Sherlock was an experimental program. Early CAI researchers had been concerned about false feedback from automated tutors due to the aura of infallibility which people often ascribe to computers. Subjects seem to be more aware now of the limitations of a computer. Being aware of computer limitations and the experimental nature of the program, subjects seemed quite willing to reject Sherlock's opinion when it differed from their own.
3. The present feedback merely states that there is a problem. There is little attempt to explain why it is a problem or to help the learner build up the necessary understandings.

Despite reporting that they did not believe the feedback, subjects often changed their thinking as a result of answering Sherlock's questions. These changes are reflected in a majority of the 11 links which were aborted during the questioning process. This would suggest that a Socratic approach, which helps the learner question his own thinking, holds much promise.

8.3. SUMMARY

There is an advantage to multiple choice responses in a conventional CAI system. The learner is seldom told he is wrong when he is right, which might lead the learner to a loss of self confidence. The learner is seldom told he is right when he is wrong, which might reinforce incorrect beliefs. It has been asserted that the advantage in ITS is that a wider range of responses can be recognized and the tutor can say why something is wrong. These are important features in a learning system, but the trade-off is an increased chance of telling the learner he is wrong when he is right, or right when he is wrong.

Sherlock was able to correctly determine if a link was right or wrong over 96% of the time. This percentage indicates that whatever diagnostic power is achieved, it is achieved with very little loss in overall accuracy.

Sherlock's ability to actually diagnose **why** something was wrong was far more modest. Only a little over half of the wrong links were diagnosed correctly. It is difficult to compare this statistic to that of other approaches to intelligent tutoring systems as few provide similar statistics.

What is most encouraging is that the problems do not invalidate the model. Most problems were the result of superficial bugs or could be fixed by enhancements to Sherlock. The problems which did not suggest solutions within the existing model were of a type which would cause problems for a human tutor. That is, people seem no more effective at diagnosing what a person is thinking without (and sometimes with) the benefit of full think aloud protocols.

Chapter 9

Overview of Intelligent Tutoring System Research

9.0 INTRODUCTION

This chapter discusses four major projects which span the 18 year history of work in intelligent tutoring systems. These four projects are chosen to represent four distinct approaches to the problem of building an intelligent tutoring system. As each approach is considered, it will be contrasted with Sherlock.

9.1 SCHOLAR

One of the earliest prototypes for an intelligent tutoring system was Carbonell's SCHOLAR program (1970). SCHOLAR was designed to teach geography offering the options of student-initiated, tutor-initiated, or mixed-initiative dialogs. Figure 9.1 contains a sample dialog between SCHOLAR and a learner.

What distinguished SCHOLAR from earlier computer-assisted instruction (CAI) was that it was knowledge-based rather than program-based. Questions were generated, and responses evaluated, on the basis of a representation of the domain. This is in contrast to traditional CAI in which each question, and the criteria for evaluating the response to that question, are explicitly indicated by the programmer.

SCHOLAR	Approx what is the area of Brazil?
student	2,500,000 square miles.
SCHOLAR	Wrong. Please indicate if the following statement is correct: the area of Paraguay is approx 47,432 square miles.
student	What is the area of Brazil?
SCHOLAR	The area of Brazil is approx 3,600,000 square miles. Now answer the question you didn't answer before.
student	False
SCHOLAR	Rephrase your statement. I can't understand it.
student	Incorrect.
SCHOLAR	Very good. The area of Paraguay is approx 157,000 square miles. The language in Brazil is French. Right or wrong?
student	Wrong.
SCHOLAR:	You are correct. The language in Brazil is Portugese. ...

Figure 9.1: SCHOLAR Dialog

SCHOLAR's representation of the domain consisted of a semantic network (Figure 9.1).

An overlay approach was used for learner modelling. The network was assumed to be a functional representation of the ideal student, and each new student was assumed to have this cognitive structure until evidence indicated otherwise.

Sherlock's approach to declarative knowledge is similar to SCHOLAR's. Both use a semantic network to represent ideal knowledge, and suggest the use of the network for representing the learner model. SCHOLAR and Sherlock both make an attempt to categorize a misconception that was evident in a wrong response. The following taxonomy was suggested by Carbonell for this purpose:

1. Missing information

2. Misfiled fact

3. Wrong entry
4. Lack of a concept
5. Wrong superordinate
6. Overgeneralization error
7. Failure to draw some superordinate inference
8. Failure to draw some negative inference

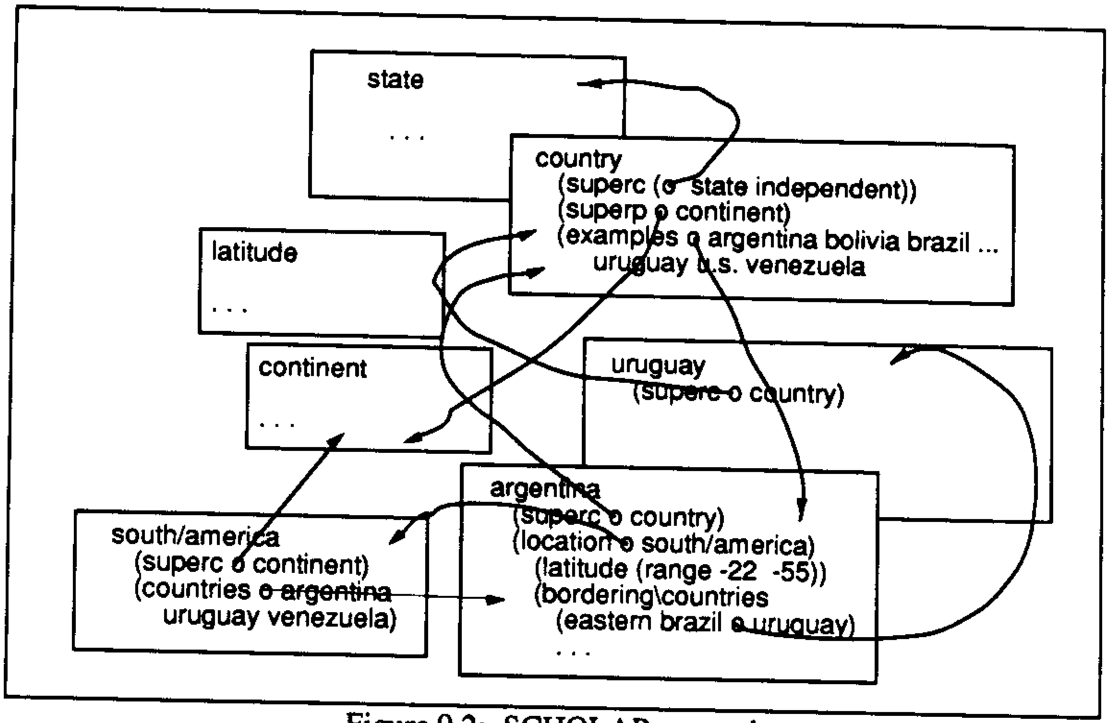


Figure 9.2: SCHOLAR network

Sherlock makes an explicit attempt to identify only three of these misconceptions:

1. missing information;
2. wrong entry;
3. overgeneralization error.

Carbonell suggested using the semantic network to represent the learner model by modifying the network as new information about the learner became available. This modelling of the learner would be accomplished by modifying or deleting nodes and links in the representation of ideal knowledge.

The problem with this approach to learner modelling is that the same representation is used to both determine if a response is correct and to represent a learner's misconceptions. Some means must be provided to differentiate between the learner's knowledge and correct knowledge. Otherwise a learner's misconception may be used to validate an incorrect learner response at some later point.

Learner modelling was not implemented in SCHOLAR. The only long-term modelling Sherlock does in the semantic network is the representation of a learner's interpretation of icons. The problem of differentiating between the learner's model and correct knowledge is avoided because the interpretation of icons is neutral. Since no icon interpretation is considered wrong, it is advantageous that the network uses this aspect of the learner's conceptions to validate responses.

For a more complete modelling of the learner, however, Sherlock will have to solve the problems left by Carbonell. Nodes and links in the semantic network must be marked as representing either:

1. the tutor's knowledge that the learner does not share;
2. knowledge that the tutor and learner share;
3. the learner's knowledge that the tutor does not share.

Despite the fact that Carbonell introduced the use of a semantic network in an intelligent tutoring systems over 25 years ago, semantic networks have not been popular in tutoring systems since then. There are two reasons for this:

1. It is cumbersome to represent procedural knowledge in a semantic network (Wenger, 1987);
2. Generating inferences based on inheritance in a semantic network is complicated (Woods, 1975; Brachman & Levesque, 1985).

Sherlock attempts to solve these problems with two features which most clearly distinguish Sherlock from SCHOLAR

1. modelling of procedural knowledge in a separate representation;
2. a connectionist approach to spreading activation.

9.2 THE IMPORTANCE OF PROCEDURAL KNOWLEDGE

SCHOLAR was concerned only with the learner's declarative knowledge. This would seem appropriate to the task of learning facts about geography. However Carbonell recognized that learners should not only know certain facts but should also have strategies for making guesses about knowledge they do not have (Carbonell & Collins, 1973).

For example, if the learner wanted to know if country X exports coffee, but has no explicit fact either stating that country X does export coffee or country X does not export coffee, what should he conclude? Normally we will not have explicit negative facts, but does the lack of a positive fact indicate that country X does not export coffee? One strategy is to determine the likelihood that the positive fact would be known if it was true. If, for example, we have facts about exports for country X and facts about other countries exporting coffee, it is more likely that we would have such a fact for country X exporting coffee if it was true.

Carbonell and Collins discussed extensions to SCHOLAR which would allow the system to make these kinds of inferences about its own knowledge. But these were never implemented. SCHOLAR can only identify the cause of learner errors in terms of the learner's declarative knowledge.

Intelligent tutoring systems since SCHOLAR have emphasized procedural learning. Production systems and goal-plan hierarchies have been found to be better suited than semantic networks for representing the target knowledge in procedural learning tasks. This has enabled more recent systems to handle the type of strategic knowledge which SCHOLAR could not handle. Unfortunately this only reverses the problem, as it is just as cumbersome to represent declarative knowledge in a production system.

Sherlock is able to use a semantic network because procedural knowledge is represented in a separate representation.

9.3 GENERATING INFERENCES FROM A SEMANTIC NETWORK

The idea of a semantic network originated with the work of Quillian (1967). Carbonell's SCHOLAR was one of the first important applications of Quillian's semantic network. Quillian used a semantic network to simulate the human ability to compare and contrast arbitrary English words, similar to Sherlock's use of the semantic network. Similar to Sherlock, Quillian's network determined the relationship between two concepts using a form of spreading activation to find the intersection, or common path, between the two concepts.

Although Quillian's networks were successful at finding the relationship between concepts, they were less successful as a general understanding model. One reason for the limited success was that Quillian's network, and most early semantic networks, lacked a defined semantics for the links (Brachman, 1979). There was, for example, no distinction between general inheritance links and domain-specific links such as SCHOLAR's location link (see Figure 9.2). As a result there was little direction to the search, and thus combinatoric problems quickly made search unreasonable.

More recent work on semantic networks has resulted in defined link semantics and a better definition of the processes which operate on a network. Sherlock is based largely on Fahlman's NETL (1979). Fahlman's network is organized around what he calls "virtual copies." When a sub-type is created it carries with it a virtual copy of any properties of the parent node. An inherited property is explicitly represented when some modification on the property is necessary. Fahlman also introduced the idea of roles, which are full-fledged nodes, as used in Sherlock.

Fahlman used a small set of primitive links which controlled the passing of markers. Gasser (1988) combined spreading activation with marker passing in his implementation of CHIE. Built on Fahlman's concept, Gasser added the idea of links which consisted of weighted connections in each direction and the basic rules for spreading activation which are implemented in Sherlock.

9.4 WEST

Burton and Brown (1976) explored the idea of a computer-based coach in WEST. WEST is an arithmetic game in which the learner plays against the computer or another opponent. The goal of their project was to construct an environment in which the learner is free to explore and learn from mistakes. This type of "discovery learning" has been shown, in many circumstances, to be more effective than didactic training (Guthrie, 1967; Wittrock, 1963; Wittrock & Twelker, 1964). By discovering relationships on his own, the learner is more likely both to remember the relationships. A more important goal of discovery learning is that by learning to learn on his own, the learner has a greater chance of discovering new relationships in the future.

The problem with discovery learning is that the learner can spend a lot of time in non-productive searching. It is possible that the learner will not discover the relationships for which the discovery learning task was designed. It has also been shown that the increased effectiveness in future self-discovery is contingent on the learner's success, not just the process of discovery learning (Anthony, 1973).

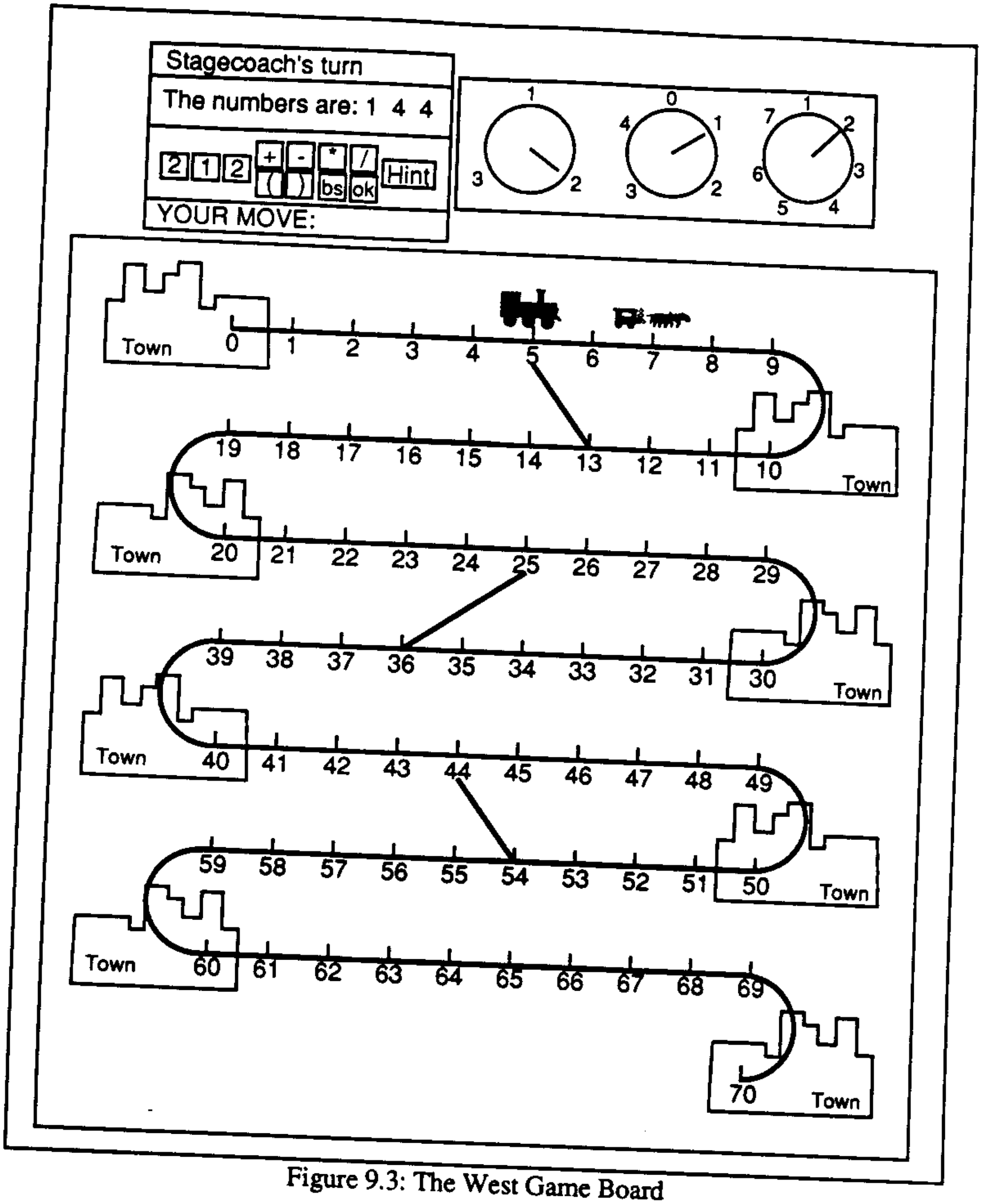
To solve this problem, WEST incorporated an intelligent coaching module to reduce non-productive searching (Burton & Brown, 1979; Goldstein, 1979). To provide a balance between open-discovery and guided-learning, the coaching module employed a set of pedagogic heuristics. These would take into account such factors as the level of the learner, how many times a mistake had been made, and the relevance of the mistake at a particular point in the game.

As a coached discovery learning environment, WEST serves as a proto-type for Sherlock. The major difference in philosophy involves the amount of coaching which is deemed desirable. Because WEST was primarily a game, its designers felt that any intervention should be kept to a minimum, so as not to hamper the enjoyment of the game. In Sherlock, whenever a possible misunderstanding is discovered, the coach intervenes.

The WEST game involves using randomly generated integers in an arithmetic expression. The result of the arithmetic expression determines how many spaces the player's piece moves on the game board (Figure 9.3).

The first player to reach the end of the board wins. The rules of the game require that the learner consider a number of possible strategies in order to produce the most desirable move. These rules include:

1. If you land on a town (there is a town every 10 spaces), you automatically jump ahead to the next town.
2. If you land on your opponent's space when he is not on a town, your opponent has to go back to the last town.
3. If you land at the top of a short-cut, you automatically jump to the bottom of the short-cut.



Stagecoach's turn

The numbers are: 1 4 4

YOUR MOVE:

Figure 9.3: The West Game Board

For each move, the learner must decide which of the following strategies is most effective:

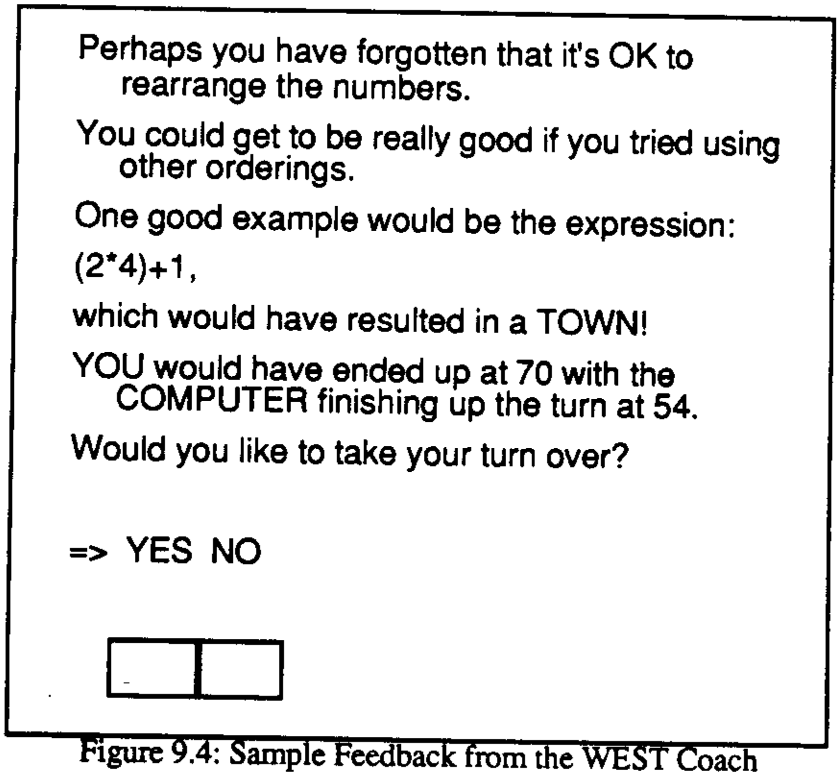
1. trying to make the largest number possible;
2. trying to land on a town;
3. trying to land on his opponent's space;

4. trying to land at the top of a short-cut;

Making the optimal move involves picking the right strategy and being able to manipulate the arithmetic expression to implement the strategy. The arithmetic strategies which can be used to implement the best move include:

1. use of parentheses to affect order of operations;
2. use of zero;
3. creating negative numbers;
4. rearranging numbers;
5. multiplying by zero

The coach in WEST watches each move the learner makes. The coach determines whether the learner's move was optimal, and if not, why the optimal move was not made. Once the determination is made, the coach decides whether it should intervene with advice. Figure 9.4 contains sample feedback provided by the coaching module in WEST.



Perhaps you have forgotten that it's OK to rearrange the numbers.

You could get to be really good if you tried using other orderings.

One good example would be the expression:

(2*4)+1,

which would have resulted in a TOWN!

YOU would have ended up at 70 with the COMPUTER finishing up the turn at 54.

Would you like to take your turn over?

=> YES NO

Figure 9.4: Sample Feedback from the WEST Coach

Burton and Brown make a distinction between two approaches to modelling strategic knowledge:

- "glass box" or articulate model. It is called articulate because it solves a problem as a human would, and can explain each step in human terms;
- "black box" or non-articulate model. Solves problem is the most efficient manner possible, not necessarily the same way a human would. This model is not able to explain why it does what it does.

Their approach is to combine the best of both models in order to diagnose a learner's move. The "black box" expert is used to efficiently determine the range of optimal moves. If the learner has not made the same move, the "glass box" expert is used to explain why the optimal move was not made. The "glass box" expert does not have to be able to compute an entire move, only to work backwards from a wrong move to determine why it was wrong.

Diagnosis involves comparing each of the learner's moves with what the non-articulate expert module would do. Each possible move is associated with skills which are necessary to make that move. It is assumed that a learner who does not make an available good move is missing these associated skills.

Sherlock also makes this distinction between articulate and non-articulate strategic knowledge. The two types of knowledge are also used in a similar manner:

1. Sherlock uses the knowledge necessary to perform the task, represented in the production system ("black box" model), to determine if the learner has made a good link;
2. Sherlock uses the knowledge necessary to explain the task, represented in the semantic network ("glass box" model), to help determine why a better link was not made.

The major difference is in the implementation of the articulate model. The WEST articulate model consists only of the issues which would be involved in any given move. The system does not model how knowledge of those issues interacts with each other, and the learner's other knowledge, to determine a move. The system therefore has no way of knowing which of the issues is really responsible for a bad move.

For example, assume that the learner's piece is on a town, and the three integers are: 1,3, and 0. WEST determines that the optimal move is:

$$0 * (1 + 3) = 0$$

so that the piece lands on the same town and can jump ahead to the next town. If the learner does not make this move, WEST assumes that the learner lacks knowledge of the strategy "multiply by zero results in zero." But there are two other possibilities:

1. The learner knew that multiplying by zero resulted in zero, but did not realize that would be the optimal move in this case;
2. The learner knew that multiplying by zero resulted in zero, but did not realize that a zero move meant that you "landed" on the same spot.

In Sherlock the articulate model is implemented as a part of the declarative knowledge used for the task. Sherlock is therefore able to model how issues interact with other knowledge to lead to a particular move.

9.5 ANDERSON'S TUTORS

The Anderson group at Carnegie-Mellon University has produced intelligent tutoring systems in the domains of LISP programming and geometry (Anderson, Boyle, Farrell & Rieser, 1984; Anderson, Farrell & Saurers, 1984; Anderson, Boyle & Reiser, 1984). The Anderson tutors are unique in ITS research in that they are complete systems. Rather than being proto-types intended to demonstrate an approach to one or two aspects of the tutoring problem, they are functional tutors which are actually used to tutor. They have thus been able to demonstrate effectiveness in teaching.

One claim of the developers is that their tutors are based on a comprehensive model of learning: Anderson's ACT* model (1983). The basic framework for this model is illustrated in Figure 9.5.

A unique feature of Anderson's model is the separation of procedural and declarative knowledge. Each kind of knowledge is stored in a system best suited for the use of that kind of knowledge. A production system represents procedural knowledge. The production system operates on declarative memory which is represented by an associative network.

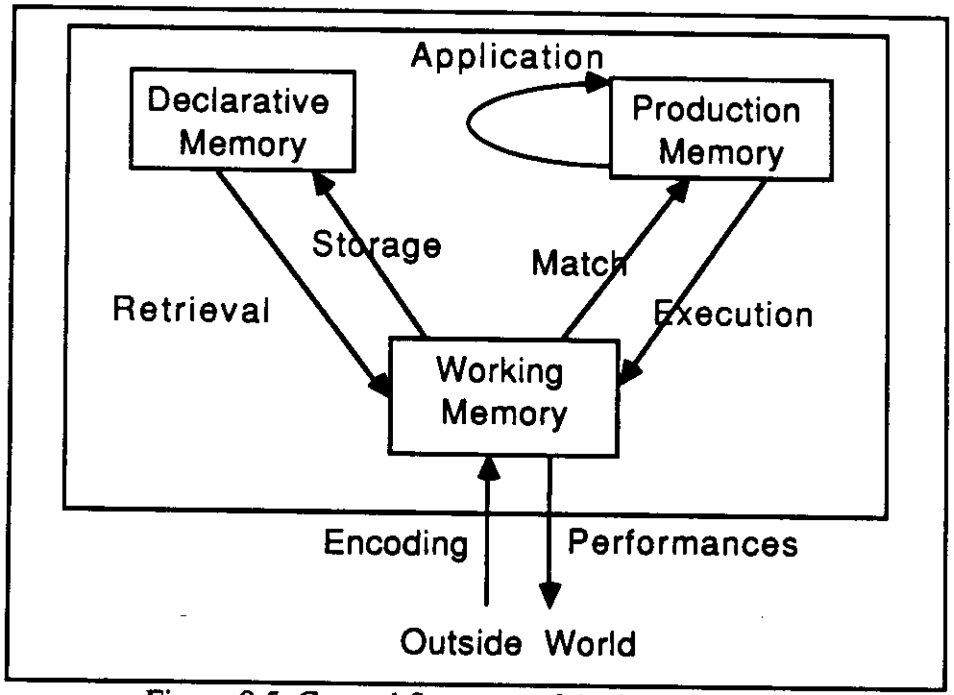


Figure 9.5: General Structure of Anderson's Model

One reason for this separation was the observed differences in human acquisition of declarative and procedural knowledge. It seems relatively easy for humans to learn a new fact. A comedian recently claimed that television stations in the south sign off by saying:

> It is Ten O'clock...
> Do you know what time it is?

This is only funny because we assume that anyone hearing the first line will obviously know the answer to the second line. Although we are often unable to retrieve a fact after any length of time, we can learn the fact just by hearing it.

Learning procedural knowledge, on the other hand, appears to be more difficult. Anderson hypothesizes that when we are first told how to do something, we store it in declarative memory. In this form it is difficult to access, and thus carrying out actions which are stored in declarative memory is slow and stilted. We practice a skill in order for it to become a part of our procedural memory.

Sherlock's model of learning is influenced strongly by ACT*. The distinction between procedural and declarative knowledge, the design of the representation system for each kind of knowledge and their interaction all follow the model specified by Anderson in ACT*.

It might seem that since both Sherlock and the Anderson tutors are based on ACT*, that the systems would be similar. The Anderson tutors, however, have no representation for declarative knowledge component of the ACT* model shown in Figure 9.5. There is thus no attempt to model the acquisition of declarative knowledge or the interaction between declarative knowledge and procedural knowledge.

The procedural knowledge component, similar to Sherlock, is represented as a set of if-then rules to guide behavior. Sample productions are contained in Figure 9.6.

P1: IF the goal is to code the second member of a list
THEN use CADR and set a subgoal
to code the list.

P2: IF the goal is to obtain all the elements which have a relation to any member of a list
THEN use MAPCONC and set as subgoals

1. To code a funtion that will return all the elements that have a relation to the argument.
2. To code the list.

Figure 9.6: Sample Productions in Anderson's Lisp Tutor

By specifying productions at a low enough level, these systems can pin-point the specific piece of a process which is missing, and thus provide tutoring. The Anderson tutors and Sherlock both evaluate a learner's action by determining if it matches the action portion of an appropriate production. The two systems, however, use very different methods of determining appropriate productions.

In the graphic mapping task, a wide range of appropriate productions is possible at any given point. To determine which productions are appropriate, Sherlock must use the declarative knowledge of the text. The Anderson tutors bypass the need for using declarative knowledge by representing a specific strategy which assumes a complete and correct knowledge of the declarative facts needed to accomplish the task. It is assumed that the learner is following the same strategy the tutor would use. At every point in the lesson, the tutor is expecting a specific production to be used. Diagnosing the cause of an inappropriate action is simple: assume that the learner does not have the correct production which the tutor would have used.

If the learner does not use the correct production in the Anderson tutors, he is stopped and corrected. The learner is not allowed to pursue a wrong path in the hope that he will discover it is wrong on his own. This approach is based on a pedagogic belief that pursuing a non-productive path will teach non-productive behavior. This approach also avoids most of the difficulties associated with tracking a learner's intentions by ensuring that the learner does not engage in a strategy which is different from the tutor's strategy.

Sherlock also stops the learner as soon it is determined that he is making a mistake. The difference is that more of an effort is made by Sherlock to determine if the learner is pursuing a strategy which is different, but still appropriate. Sherlock does this by:

1. representing non-optimal but appropriate rules as well as optimal rules;
2. using spreading activation to classify the learner's plan despite possibly incomplete or inaccurate information;
3. representing alternate interpretations for the icons.

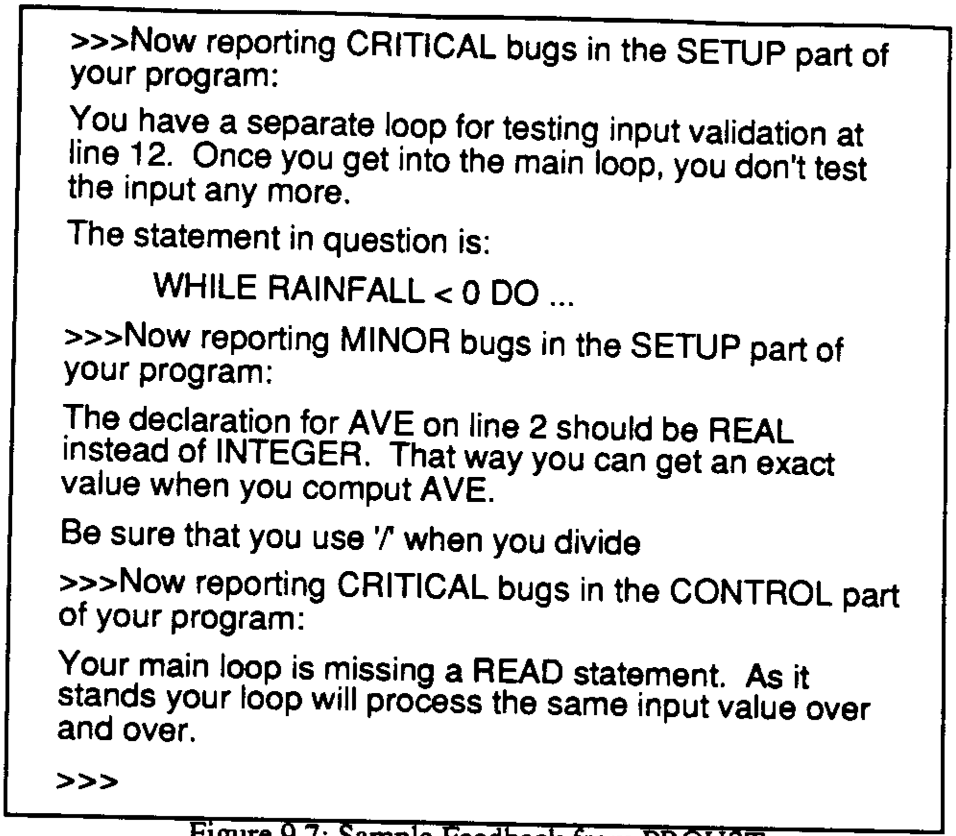
9.6 PROUST

A representation of goals and plans is used to infer the cognitions behind a learner's actions in PROUST (Johnson, 1985; Johnson & Soloway, 1983). The domain is programming in Pascal. Both this program and Anderson's Lisp tutor are designed for the same purpose: to provide an on-line aid to students completing programming assignments. The difference is that rather than looking over the learner's shoulder while he programs, PROUST waits until the learner is done to look at the program.

Sample feedback from PROUST is indicated in Figure 9.7.

PROUST's learner model is based on the goals and plans used to implement a program. PROUST is given a statement of the assignment including the major goals which are implicit in the assignment. Its expert model is able to generate the sub-goals which could accomplish these goals, and the possible plans which would accomplish each sub-goal. The plans are then matched against the learner's program to try and identify which plans the learner was attempting to use. PROUST also has transformation rules which enable it to recognize variations in the way a plan might be implemented.

Once PROUST has identified the learner's plans, it can identify where the learner has gone wrong in implementing the plan and offer advice. In this way, rather than imposing a particular programming style, PROUST begins by figuring out what approach the learner is trying to use. Thus any tutoring provided by PROUST is aimed at helping the learner fulfill his own plan.



```
>>>Now reporting CRITICAL bugs in the SETUP part of
your program:

You have a separate loop for testing input validation at
line 12. Once you get into the main loop, you don't test
the input any more.

The statement in question is:

      WHILE RAINFALL < 0 DO ...

>>>Now reporting MINOR bugs in the SETUP part of
your program:

The declaration for AVE on line 2 should be REAL
instead of INTEGER. That way you can get an exact
value when you comput AVE.

Be sure that you use '/' when you divide

>>>Now reporting CRITICAL bugs in the CONTROL part
of your program:

Your main loop is missing a READ statement. As it
stands your loop will process the same input value over
and over.

>>>
```

Figure 9.7: Sample Feedback from PROUST

PROUST is similar to Sherlock in that it begins by trying to reconstruct the learner's intentions before attempting to evaluate what the learner's actions. The approach used to reconstruct the learner's intentions is also similar; the available information is used to match the learner's plan with the plans which the tutor knows about. The difference is that PROUST is able to obtain the information needed from the program which the learner has written, Sherlock must ask the learner additional questions to obtain sufficient information.

PROUST's major limitation is rooted in the complexity of allowing the learner to pursue individualistic plans. The expert model must have a great deal of information about each kind of goal that would appear in a problem, and most of the legitimate plans for accomplishing those goals. For that reason, the expert model has currently been generalized to handle only two assignments, both of these of a similar type. Much work remains to generalize the PROUST approach to handle a wider range of assignments.

Sherlock has been tested with only one text fragment, the consideration text. But since the strategy needed to make gm-links is simpler than the strategy needed to write a computer program, it is theoretically easier to represent a strategy in Sherlock which will generalize to new texts. In addition, the new knowledge needed for a new text is represented separately from the strategy in the semantic network.

A major difference between PROUST and Sherlock is in the relationship between the diagnosis and the tutoring which is provided based on that diagnosis. PROUST's goal is to identify bugs in the learner's program. The intention-based approach is used only as an intelligent means of identifying bugs by identifying the differences between a learner's plan and his implementation of that plan. All tutoring consists of pointing these bugs out to the learner.

Sherlock's goal is to identify the reason a mistake was made. An intention-based approach is used in Sherlock to identify what aspect of the learner's knowledge led to the error. Tutoring in Sherlock is focused on clarifying the misconception which led to the error rather than pointing out the error.

A comprehensive effort was made to evaluate PROUST's performance (Johnson, 1986). It was found that PROUST could provide a complete analysis for over 70% of the programs which it analyzed, and in those cases found over 90% of the bugs. PROUST was able to provide a partial analysis of another 20-25% of the programs, and find approximately 40% of the bugs in those programs. PROUST could not analyze 5% of the programs submitted to it. Taking all categories into account, PROUST correctly recognized 75% of the bugs in all programs submitted to it.

It should be noted that the evaluations of PROUST were done with a much larger sample than the evaluation done on Sherlock. Since the large sample size provides statistical significance to the PROUST figures, lacking in the Sherlock figures, a direct comparison would be inappropriate.

The PROUST evaluations also indicated a number of false alarms (correct statements labeled as bugs). In the first study of 155 programs, for example, there were 46 false alarms. Twenty-six of the programs had one or more false alarms. There was only one false alarm in the Sherlock evaluation of 59 links. Again, however, direct comparisons across such different tasks and systems is not possible.

What is missing from the PROUST evaluation is an assessment of the accuracy of the plan diagnoses which led to the bug reports. While an accurate plan diagnosis certainly contributes to accurate error identification, accurate error identification does not necessarily mean that the plan was diagnosed correctly. This omission reflects the difference in goals between PROUST and Sherlock noted above. Johnson states, "The bottom-line issue in evaluating the work that has gone into PROUST is whether or not it has resulted in an effective tool for finding novice bugs" (Johnson, 1986, p. 225).

An outside evaluation of Proust was conducted by the Center for the Study of Evaluation (Baker, Aschbacher, Feifer, Bradley & Herman, 1985). The CSE study evaluated two aspects of Proust:

1. the quality of instructional strategies and principles incorporated into PROUST;
2. the learning outcomes for students.

In terms of the quality of instructional content, PROUST was found to suffer from two deficits also shared to some extent by Sherlock. First, PROUST is an incomplete tutoring system concentrating on diagnosis. Thus there has been no attempt to refine the feedback provided to learners. Second, there is no cumulative modelling of the learner. Each time the learner submits a program to PROUST it is treated as a one-time event. Thus PROUST cannot build on tutoring that may have been provided during an earlier iteration.

Another problem which the study cited is the lack of interaction with the learner. PROUST evaluates a program as a whole; there is no attempt to get any further information from the learner or confirm any aspect of the diagnosis while evaluating the program. Sherlock provides this interaction by asking the learner to identify the reasons that a questionable link was made. This information is used by Sherlock to both narrow down the diagnostic search space and confirm the diagnosis.

In terms of learning outcomes, an attempt was made to assess the effect of PROUST on the learner's ability to program. One class of 69 students was given the opportunity to use PROUST with the two homework assignments for which PROUST was designed. A control group of 72 students had no access to PROUST. Programming ability following the homework assignments was measured using the mid-term and final scores. Neither the experimental or control group scored high on the exams (53% on the final), and there was no significant difference between the groups.

Two possible explanations for the lack of demonstrable main effects are cited by the CSE report:

1. a number of confounding effects including the fact that both control and experimental groups had access to TAs for help during programming assignments;
2. exposure to PROUST for only one or two homework assignments may not be "sufficiently strong instructionally to exhibit effects." (Baker, et. al., 1985).

9.7 SUMMARY

The four intelligent tutoring systems discussed in this chapter have served as the foundation for Sherlock. SCHOLAR and WEST created the field of intelligent tutoring system research; PROUST and the Anderson tutors are more recent projects demonstrating some of the capabilities of the technology.

In SCHOLAR, Carbonell introduced the idea of a computer-based tutor which relied on a representation of the knowledge rather than a script. In WEST, Burton and Brown introduced the use of a computer-based coach in a discovery learning environment. These systems serve as proto-types for the potential of intelligent tutoring systems.

The more recent systems have begun to fulfill some of the promises of the early work. The Anderson tutors and PROUST have been used in educational settings and have demonstrated that they are effective educational aids. The Anderson tutors, however, are successful educational tools because they limit the range of learner actions. While PROUST allows a much greater range of learner actions, it is less accurate and difficult to generalize.

Sherlock incorporates the best elements of each system. Sherlock is a more complete implementation of the ACT* model on which the Anderson tutors are based. By modelling both procedural and declarative knowledge, Sherlock is able to recognize a learner's individual approach to the task while providing accurate diagnosis.

Chapter 10

Future Work and Conclusions

10.1. LIMITATIONS AND FUTURE WORK

This section discusses some possible future directions for the work begun in Sherlock.

10.1.1. COMMUNICATING WITH THE LEARNER ABOUT FACTS

One weakness in the present version of Sherlock is its limited ability to communicate about facts. Part of this weakness is inherent in any tutoring. If it were always possible to clearly and unambiguously state facts to learners, a tool such as Sherlock would not be necessary to aid in the comprehension of text.

A small sample of 15 subjects (separate from the previously reported studies) was surveyed to determine how people would state common relationships. People were asked, for example, "How would you express the relationship between robin and bird?" Several possibilities were offered, as well as the option of filling in their own. There was no clear consensus on how to express any of the facts in natural language.

At present there is no solution to the problem of communicating facts about relationships. Several future steps are possible, however, that would enhance Sherlock's ability to ascertain what facts the learner believes.

10.1.1.1. STATING FACTS

More effort is needed is to attempt to clearly and less ambiguously articulate the facts. Because there is no one correct way to express a relationship in natural language, however, there will always be some ambiguity. One alternative is for Sherlock to offer a version of the survey mentioned above, on-line, before the learner began constructing his graphic map. The expression chosen for each relationship could then be used as a paradigm for stating each of relationships during the session.

10.1.1.2. UNDERSTANDING THE LEARNER'S ANSWER

One approach is to treat the interpretation of facts as another possible source of confusion, similar to the current treatment of icon interpretations. All known interpretations of a fact could be represented. When trying to understand the answers to a question, Sherlock would try the alternatives to see if any of these better explain the learner's actions.

10.1.1.3. RECOGNIZING INCONSISTENT ANSWERS

It would be nice in tutoring if the tutor knew when the learner was not accurately stating his own beliefs. Human tutors are at least capable of recognizing when a learner is giving inconsistent answers; Sherlock is not.

For example, learners using Sherlock will occasionally indicate two contradictory reasons for making a gm-link. Although people seem to be capable of maintaining contradictory beliefs, it is also possible that these learners are just confused, not really thinking about their answers, or interpreting one or both of the statements in such a way that they are not inconsistent.

Whatever the case, it would be useful for Sherlock to notice that two apparently inconsistent facts are chosen. Sherlock could then take steps to determine which of the above cases was true, and take appropriate action.

Recognizing inconsistent facts should not be difficult, as the facts are a part of the semantic network. Currently this property is used to determine the learner's implied beliefs. For example if the learner has indicated a belief that "Every block is red" then Sherlock infers that the learner also believes that, "It is possible for a block to be red." Sherlock can infer some negative implications, for example, if the learner believes "Every block is red" he will not believe that "Every block is blue."

Once the rules needed to make these negative inferences are better represented, it is a small step to make sure that there is no overlap between the things we infer that the learner does not believe, and the things the learner says that he does believe.

10.1.2. CLASSIFYING UNKNOWN PLANS

It has been indicated that Sherlock has a tendency to classify a learner's plan as an instance of one of the known plans, even though it might be an unknown plan. One way to allow Sherlock to classify rules as unknown is by adding rules to the semantic network which lead to each of the gm-link types, but have no antecedents.

If such a rule had a stronger than normal link to the consequent, and no antecedents, it would match in situations where there are no strong antecedents matched in any of the rules with the same consequent. Sherlock could then try and infer the antecedents for that rule by looking at the facts that the learner believes. Sherlock could then at least suggest the possibility that it is a buggy rule. It might also compare this to the closest good rule, and point out the differences in antecedents.

10.1.3. PROVIDING INTELLIGENT TUTORING

The feedback offered by the current version of Sherlock is only a first step towards intelligent tutoring. As was indicated by the evaluation study reported in Chapter 8, just telling the learner about a mistake had little impact.

It is much more effective to help the learner recognize his own mistakes and discover for himself more appropriate actions. Automatically generating a self-learning experience, however, is a non-trivial task. No intelligent tutoring system has yet been able to provide this type of tutoring.

One possible approach is to use Sherlock's diagnostic information and knowledge representations to facilitate directed analogical reasoning (Burstein, 1985).

This approach could be used when Sherlock has determined that a learner does not understand concept X and Sherlock knows of an analogous concept A which is part of a more familiar domain. An analogous relationship could defined in Sherlock's network as a set of nodes with links identical to the links connecting the nodes in the target concept (Figure 10.1).

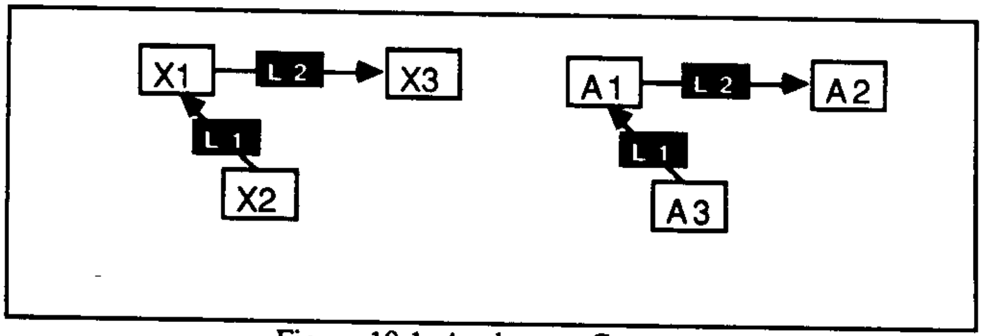


Figure 10.1: Analogous Concepts

The steps to providing the learner with directed analogical reasoning are:

1. Ascertain that the learner understands the relevant relationships in concept A.
2. Indicate that concept A is analogous to concept X.
3. Help the learner map relationships from A to X.

Consider, for example, that a learner has made the construction pictured in Figure 10.2.

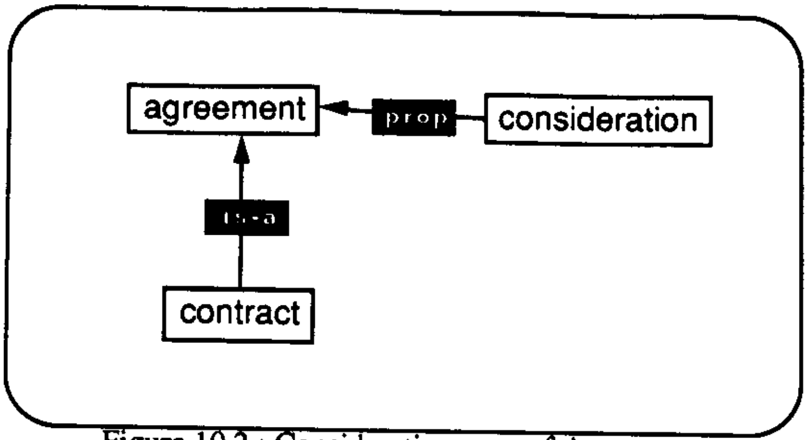


Figure 10.2 : Consideration prop of Agreement

Assume that Sherlock has determined that the learner does not correctly understand the relationship between these concepts. To automatically generate a remedial exercise Sherlock would look for nodes in a familiar domain which have the same s-links between them as AGREEMENT, CONTRACT, and CONSIDERATION. In other words, Sherlock would look for a node which is a role of a second node, which is in turn a specialization of a third node (Figure 10.3).

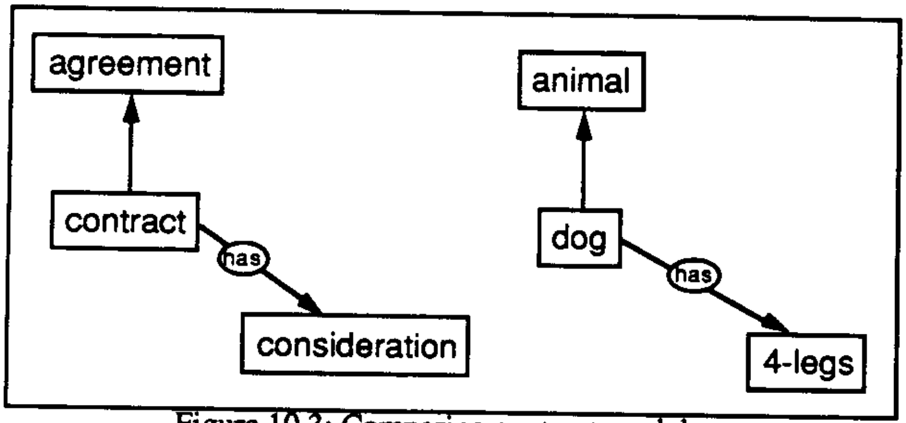


Figure 10.3: Comparing contracts and dogs

The learner would be presented with a new window containing icons corresponding to each of the nodes in the analogous relationship (Figure 10.4). A prompt would instruct the learner to link these icons to show their relationship to each other. If the learner makes appropriate links in his construction, Sherlock would assume that he understands the relationship in the analogy (step 1 above). If the learner were not able to produce an appropriate construction, this would indicate that the problem has to do with the learner's strategies for picking appropriate links and thus would provide tutoring for his graphic mapping strategy.

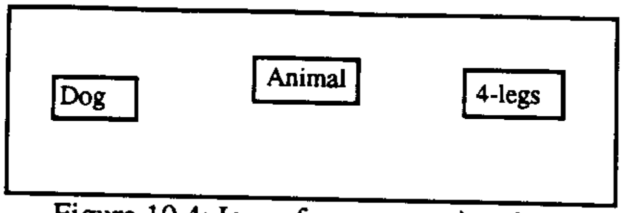


Figure 10.4: Icons for representing dogs

If the learner does understand the analogous relationship, Sherlock would then indicate that the relationship between agreement, contract and consideration is similar to the relationship between animal, dog, and 4-legs (step 2). Telling the learner that the two relationships are analogous may seem like an indirect way to give the learner the correct answer, and in a sense it is. But the correct answer is actually the internal cognitive structure a tutor has for the relationship between agreement, contract, and consideration. Giving this correct answer to the learner is very difficult.

Presenting the learner with Sherlock's representation for the relationship (as illustrated on the left side of Figure 10.3) would have little meaning. There is little chance that Figure 10.3 actually corresponds to a human being's cognitive structure. In representing the analogous relationship the learner is activating portions of his own internal cognitive structure which correspond to the correct relationship. What we are trying to do is point to this structure in the learner's mind and say, "That is the correct answer."

To use this "correct answer," the learner must be able to map the concepts in the analogy to the present domain (step 3). Sherlock would flash one of the icons in the analogous relationship and ask the learner to click on the icon in the original construction which corresponds to this icon. If the subject had any problem with identifying the corresponding icon, Sherlock would present the analogy at a lower level. For example, "The relationship between dog and 4-legs is similar to the relationship between contract and consideration."

10.1.4. KNOWLEDGE ACQUISITION

One advantage of ITS over CAI is that because knowledge in the system is represented explicitly, it should be easier to change parts of the knowledge to adapt the tutor to new situations. In Sherlock's case this could mean adapting the system to tutor learners building a graphic map for a new text in a different domain. All that would be necessary is representing the new text and domain. The tutoring strategies, graphic mapping strategies and the processes which operate on the text knowledge all remain the same.

This extensibility is important considering the development which goes into a system such as Sherlock. It would be ideal if a subject matter specialist and/or teacher could adapt Sherlock to work with any new text. The reality, however, is that representing a new text and domain is a formidable task, especially for someone without training and experience in knowledge engineering.

It was mentioned in Chapter 1, at some time in the future a natural language understanding system could be used to automatically parse a text into Sherlock's internal representation. In the meantime, another approach is to provide an intelligent interface to facilitate the representation of knowledge by non-knowledge engineers.

This knowledge acquisition interface could use the graphic mapping interface used by learners. The user would begin by creating icons to represent concepts in the new domain. The user would then use gm-links to represent the relationships between the concepts. Rather than evaluating these gm-links, the knowledge acquisition interface would use Sherlock's graphic mapping strategy to infer an internal representation for the gm-links. The interface would ask any necessary questions to disambiguate the users constructions. In this way the user would teach Sherlock what it needs to know about the new domain.

10.2. SUMMARY AND IMPLICATIONS

Carbonell's work with SCHOLAR suggested exciting contributions to education. The goal was a computer program that understood the domain it was teaching, knew how to teach, and had a model of what the learner already knew. This computer program could then generate its own tutoring based on its knowledge. This tutoring would be individually tailored to each learner and able to respond to the learner's direction. A computer program such as this would certainly benefit any educational enterprise.

Work since SCHOLAR has moved closer to this goal. There has been a move away from learning facts and towards learning processes through the representation of procedural knowledge. Systems have offered a more learner directed environment (WEST), enhanced ability to work with a learner's individual approach to a problem (PROUST), and a better use of research in learning and instruction (Anderson tutors).

The complexity of representing and using knowledge, however, has shown Carbonell's prediction of automatic diagnosis in the near future to be unrealistic (Wenger, 1987, p. 32). And in terms of actually teaching, we have yet to see an intelligent tutoring system which can offer intelligent tutoring in an educational setting.

Similarly, learning-strategy research offers the possibility of dramatic improvement in our ability to educate. The goal of learning-strategy research is to teach unsuccessful learners to be successful learners. If this is possible, we can enhance an individual's chance of success in all future educational endeavors.

A major challenge in utilizing learning strategy research is training the learner to use a learning strategy. Successfully using a learning strategy requires that the learner actively integrate his existing knowledge with new knowledge. It is thus important that any training be personalized so as to account for the learner's existing knowledge. It is also essential that the learner know why something is wrong if he is to be able to generalize the strategy to future tasks where the tutor is not available.

Sherlock is an attempt to build on the work in intelligent tutoring systems to move a step closer to Carbonell's promise. In the process it is also an attempt to offer a solution to the problem of offering personalized training in the use of a learning strategy.

Sherlock is the first intelligent tutoring system to attempt to offer a combination of the major achievements of the systems which I have listed. Sherlock emphasizes learning of process through the modelling of procedural knowledge. But this is not done at the expense of factual knowledge. Both the procedural knowledge and the declarative knowledge needed for a task are represented. Each type of knowledge is represented in a manner most appropriate to that type of knowledge. Separate processes work on each type of knowledge, and together model the task being taught.

Sherlock offers a learner-directed discovery learning environment similar to WEST. But Sherlock is not restricted to passively watching the learner and attempting to infer his intentions. Sherlock has facilities for asking the learner to explain his actions, and then using that information to more accurately infer the cognitions which led to the action.

Sherlock is based on the same cognitive model as the Anderson tutors, but is a more complete implementation of that model. Sherlock is able to model a learner's action as the result of the interaction between that learner's beliefs about the domain and his strategy for dealing with the domain.

Sherlock is also the first ITS to use a localist, spreading activation network as part of its representation. Spreading activation gives Sherlock two important capabilities;

1. Sherlock can recognize even unanticipated relationships between concepts in order to determine what kind of gm-link can be made between any two concepts.
2. Sherlock can classify a learner's plan as an instance of one of Sherlock's known plans, thus expanding the system's ability to infer a learner's intentions.

An example of the ability to recognize unanticipated relationships can be seen in Jennifer's second attempt shown in Figure 10.5 (repeated from Chapter 1).

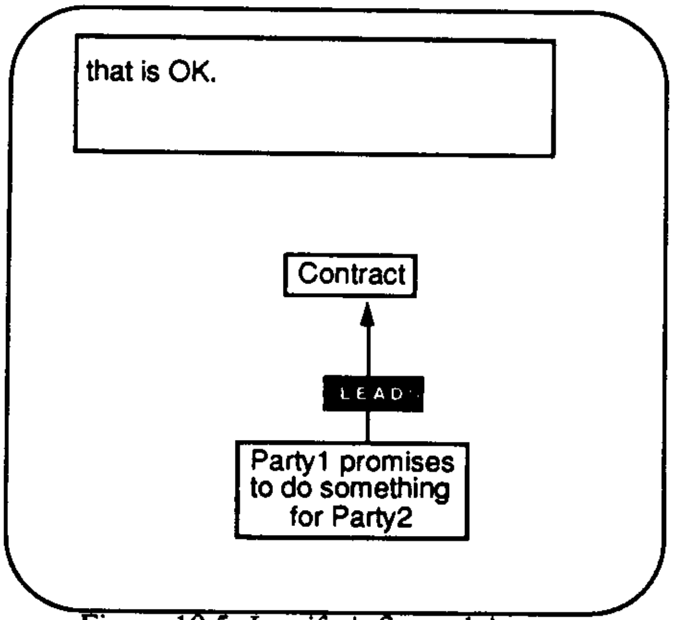


Figure 10.5: Jennifer's Second Attempt

I had been expecting the subject to make a PART gm-link at this point, and considered the subject's gm-link to be wrong. PARTY1 promising to do something for PARTY2 leads to PARTY2 doing something in return, but I had never thought of it as leading to the contract as a whole. The gm-link in Figure 10.5 can thus be considered unanticipated.

Sherlock's feedback, however, indicated that the subjects gm-link was "OK." Sherlock considered the gm-link to be correct because PARTY2 doing something in return is a necessary component of a contract, and thus anything that is a necessary condition of PARTY2'S act (i.e., PARTY1'S promise) is also a necessary condition of a contract. Although this particular relationship was unanticipated, Sherlock was able to find it using spreading activation.

Sherlock's ability to classify a learner's plan that does not match a known plan was shown in Chapter 7. In this case Sherlock classified the learner's plan (shown in Figure 10.6) as a buggy instance of Sherlock's IS-A-RULE.

learner's buggy rule

```
IF      X-is-Y
and     Y-unlik-X
THEN    make an EQUIV gm-link from X to Y
```

Figure 10.6: Buggy Instance of IS-A-RULE

By classifying the rule, Sherlock is able to focus on the differences between the learner's rule and the non-buggy rule of which it is an instance. In this case only the consequent is different — IS-A-RULE's consequent states that an IS-A gm-link is appropriate. Thus Sherlock informed the learner that an IS-A gm-link is more appropriate given these antecedents.

All of the above factors combine to enable Sherlock to evaluate a learner's responses with an accuracy rate as high, or higher, than any other automated tutoring approach with the exception of multiple-choice CAI. In addition, the evaluation of Sherlock has gone further than other intelligent tutoring system research by comparing the model that leads to the system's diagnosis with the learner's think-aloud protocols.

The essential model used in Sherlock is adaptable to tutoring any task in which both the learner's declarative knowledge and the learner's procedural knowledge are important to successful accomplishment of the task. The type of knowledge the production system would need from the semantic network would change with each task, but the model of separate knowledge interacting would remain the same.

Sherlock has demonstrated that it is possible to understand unanticipated responses and diagnose the cause of misunderstanding without degrading the system's ability to make a bottom-line determination of whether a learner response is correct. The challenge now is to use this diagnostic information to deliver truly intelligent tutoring.

References

Anderson, J.R. (1983). *The architecture of cognition*. Cambridge, MA: Harvard University Press.

Anderson, J.R., Farrell, R. & Saurers, R. (1984). Learning to program in LISP. *Cognitive Science, 8*, 87-129.

Anderson, J.R. Boyle, C.F., Farrell, R., & Reiser, B. (1984) Cognitive principles in the design of computer tutors. *Proceedings of the Cognitive Science Society Conference*, 2-8.

Anderson, J.R., Boyle, C.F., & Reiser, B. (1984) An interactive computer-based tutor for LISP. *Proceedings of AAAI-84*, 106-109.

Anderson, T.H. (1979). Study skills and learning strategies. In H.F. O'Neil and C.D. Spielberger (Eds.). *Cognitive and affective learning strategies*. New York Academic Press.

Anthony, W.S. (1973). Learning to discover rules by discovery. *Journal of Educational Psychology, 64*, 325-328.

Baker, E.L., & Lindheim, E.L. (1988). *A contrast between computer and human language understanding*. Paper presented at the annual meeting of the American Educational Research Association, New Orleans, LA.

Baker, E.L., Lindheim, E.L., & Skrzypek, J (1988). *Directly comparing computer and human performance in language understanding and visual understanding*. Paper presented at the annual meeting of the American Educational Research Association, New Orleans, LA.

Baker, E.L., Aschbacher, P., Feifer, R.G., Bradley, C., & Herman, J. (1985). *Intelligent computer-assisted instruction study* (JPL contract #956881). University of California, Los Angeles, Graduate School of Education, Center for the Study of Evaluation.

Bower, G.H. (1970). Imagery as a relational organizer in associative learning. *Journal of Verbal Learning and Verbal Behavior, 9*, 529-533.

Brachman, R.J. (1979). On the epistomological status of semantic networks. Associative networks: In Findler, N.V. (ed.) *Representation and use of knowledge by computers*. New York: Academic Press.

Brachman, R.J. & Levesque, H.J. (Eds.) (1985). *Readings in knowledge representation*. Los Altos, CA: Morgan Kaufmann Publishers, Inc.

Breuker, J.A. (1984). A theoretical framework for spatial learning strategies. In C.D. Holley, & D.F. Dansereau (Eds.), *Spatial learning strategies techniques, applications, and related issues*. Orlando, FL: Academic Press.

Bruggen, J.M. van (1981). *Graphentheorie en schema's* [Theory of graphs and schematizations] (COWO Report). Amsterdam: University of Amsterdam.

Burstein, M.H. (1985, May). *Learning by reasoning from multiple analgogies*. Ph.D. Dissertation, Yale University.

Burton, R., & Brown, J.S. (1976). A tutoring and student modeling paradigm for gaming environments. In: R. Coleman, and P. Jr. Lorton, (Eds.). *Computer Science and Education*. ACM SIGCSE Bulletin, *8*, 236-246.

Burton, R. & Brown, J.S. (1979). An investigation of computer coaching for informal learning activities. *International Journal of Man-Machine Studies, 11*, 51-77.

Carbonell, J.R. (1970). AI in CAI: An artificial-intelligence approach to computer-assisted instruction. *Man Machine Systems*, MMS-11, *4*, 190-202.

Carbonell, J.R., & Collins, A. (1973). Natural semantics in artificial intelligence. *Proceedings of the Third International Joint Conference on Artificial Intelligence*, Stanford, California, 344-351.

Camstra, B. (1979). *Empirical research with "learning by schematizing."* Paper presented at the EARDHE International Conference, Klangenfurt.

Camstra, B. & Bruggen J.M. van (1984). Schematizing: the empirical evidence. In C.D. Holley, & D.F. Dansereau (Eds.), *Spatial learning strategies techniques, applications, and related issues*. Orlando, FL: Academic Press.

Cohen, P.R. & Howe, A.E. (1988). How evaluation guides AI research. *AI Magazine, 9* (4), 35-43.

Cronbach, L.J., & Suppes, P. (Eds.) (1969). *Research for tomorrow's schools: Disciplined inquiry for education*. New York: Macmillan.

Dansereau, D.F. (1978). The development of a learning strategies curriculum. In H.F. O'Neil, Jr., (Ed.), *Learning Strategies*. New York: Academic Press.

Dansereau, D.F., Collins, K.W., McDonald, B.A., Holley, C.D., Garland, J.C., Diekhoff, G.M., & Evans, S.H. (1979a). Development and evaluation of a learning strategy training program. *Journal of Educational Psychology. 71*, 64-73.

Dansereau, D.F., McDonald, B.A., Collins, K.W., Garland, J.C.,Holley, C.D., Diekhoff, G.M., & Evans, S.H. (1979b). Evaluation of a learning strategy system. In H.F. O'Neil, Jr., & C.D. Speilberger (Eds.), *Cognitive and affective learning strategies*. New York: Academic Press.

Dyer, M.G. (1983). *In-depth understanding*. Cambridge, MA.: MIT Press.

Dyer, M.G., & Read, W. (1988). *A source book approach to evaluating articicial intelligence systems*. Paper presented at the annual meeting of the American Educational Research Association, New Orleans, LA.

Ericson, K.A., & Simon, H.A. (1980). Verbal reports as data. *Psychological Review, 87*(3), 215-251.

Fahlman, S.E. (1979). *NETL: A System for representing and using real-world knowledge*. Cambridge, MA.: MIT Press.

Feldman, J.A., & Ballard, D.H. (1982). Connectionist models and their properties. *Cognitive Science, 6,* 205-254.

Feifer, R.G., Dyer, M.G., Baker, E.L., Flowers, M., & Read, W. (1985). *Teaching inferencing strategies* (Technical Report UCLA-AI-85-5). University of California, Los Angeles, Computer Science Department, Artificial Intelligence Laboratory.

Feifer, R.G., Dyer, M.G., Baker, E.L., Flowers, M., & Read, W. (1986). *SHERLOCK: A tutoring system for graphic learning strategies* (Technical Report UCLA-AI-86-5). University of California, Los Angeles, Computer Science Department, Artificial Intelligence Laboratory.

Feifer, R.G., Dyer, M.G., Baker, E.L., & Flowers, M. (1987). *SHERLOCK: Identifying tasks in graphic mapping* Paper presented at the Third International Conference on Artificial Intelligence and Education, Pittsburgh, PA.

Feifer, R.G., Dyer, M.G., & Baker, E.L. (1988). Learning procedural and declarative knowledge. *Proceedings of the Intelligent Tutoring Systems Conference - 88,* Montreal, Canada.

Fisk, M. & Mietus, N.J. (1977). *Applied business law*. Cincinnati, OH: South-Western Publishing Company.

Frase, L.T. (1969). Structural analysis of the knowledge that results from thinking about text. *Journal of Educational Psychology,* Monograph *60.*

Gasser, M.E. (1988). *A connectionist model of sentence generation in a first and second language*. Doctoral disseration. (Technical Report UCLA-AI-88-13). University of California, Los Angeles, Computer Science Department, Artificial Intelligence Laboratory..

Goetz, E.T. (1984). The role of spatial strategies in processing and remembering text: a cognitive-information-processing analysis. In C.D. Holley, & D.F. Dansereau (Eds.), *Spatial learning strategies techniques, applications, and related issues*. Orlando, FL: Academic Press.

Goldstein, I.P. (1979). The genetic graph: A representation for the evolution of procedural knowledge. *International Journal of Man-Machine Studies , 11*, 51-77.

Guthrie, J.T. (1967). Expository instruction versus a discovery method. *Journal of Educational Psychology, 58*, 45-59.

Holley, C.D. & Dansereau, D.F. (1984a). The development of spatial learning strategies. In C.D. Holley, & D.F. Dansereau (Eds.), *Spatial learning strategies techniques, applications, and related issues*. Orlando, FL: Academic Press.

Holley, C.D. & Dansereau, D.F. (1984b). Networking: The technique and the empirical evidence. In C.D. Holley, & D.F. Dansereau (Eds.), *Spatial learning strategies techniques, applications, and related issues*. Orlando, FL: Academic Press.

Johnson, W.L. (1986). *Intention-based diagnosis of novice programming errors*. Doctoral dissertation, Yale University, New Haven, CT.

Johnson, W.L. (1985, May). *Intention-based diagnosis of errors in novice programs* (U/CSD/technical report #395). New Haven: Yale University.

Johnson, W.L. & Soloway, E. (1983, August). *PROUST: Knowledge-based program understanding*. New Haven: Yale University. (U/CSD/PP, #285).

Kaplan, A. (1964). *The conduct of inquiry*. Scranton, PA.: Chandler Publishing Co.

Lange, T.E. & Dyer, M.G. (1988). Dynamic, non-local role bindings and inferencing in a localist network for natural language understanding. *Proceedings of the IEEE Conference on Neural Information Processing Systems - Natural and Synthetic* (NIPS-88), Denver, CO.

Long, G.L. (1976). The development and assessment of a cognitive process based learning strategy training program for enhancing prose comprehension and retention. Doctoral dissertation, Texas Christian University.

Lytinen, S.L. & Moon, C.E. (1988) Learning a second language. *Proceedings AAAI-88 Seventh National Conference on Artificial Intelligence*, 222-226.

Mirande, J.A.M. (1984). Schematizing: Technique and applications. In C.D. Holley, & D.F. Dansereau (Eds.), *spatial learning strategies techniques, applications, and related issues*. Orlando, FL: Academic Press.

Neeches, R., Langley, P., & Klahr, D. (1987). *Production system models of learning*. Cambridge, MA: MIT Press.

Newell, A. (1967). *Studies in problem solving: subject 3 on the crypt-arithmetic task Donald + Gerald = Robert*. Technical Report. Pittsburgh, PA: Carnegie Institute of Technology, Center for the Study of Information Processing.

Novak, J.D., Gowin, B., & Johnsen, G.T. (1983). The use of concept mapping and knowledge vee mapping with junior high school science students. *Science Education, 67*, 625-645.

Paivio, A. (1971). *Imagery and verbal processes*. New York: Holt.

Piaget, J. (1929). *The child's conception of the world* (translated by Joan and Andrew Tomlinson). London: Kegan Paul, Trench, Taubner, & Company.

Quillian, M.R. (1967). Word concepts: A theory and simulation of some basic semantic capabilities. *Behavioral Science, 12*, 410-430.

Reese, J., Adams, N. & Meehan, J. (1984) *The T manual.* New Haven, CT: Yale University, Computer Science Dept.

Rettig M, & Bates M. (1988 July). How to choose natural language software. *AI Expert*, 41-49.

Schank, R.C. (1975). *Conceptual information processing.* Amsterdam, North Holland.

Schank, R.C., & Abelson, R. (1977). *Scripts, plans, goals and understanding.* Hillsdale, NJ: Lawrence Erlbaum Assoicates.

Skrzypek, J., Mesrobian, E., & Gunger, D. (1988). *Defining general purpose machine vision metrics for evaluation.* University of California, Los Angeles, Computer Science Department, Machine Perception Lab.

Slagle J., & Wick, M. (1988 Winter). A method for evaluation candidate expert system applications. *AI Magazine*, *9*(4), 44-53.

Sleeman, D. & Brown, J.S. (1982). *Intelligent tutoring systems.* New York: Academic Press Inc.

Sumida, R.A., Dyer, M.G., & Flowers, M. (1988). Integrating marker passing and connectionism for handling conceptual and structural ambiguities. *Proceedings of the Tenth Annual Conference of the Cognitive Science Society,* Montreal, Canada.

Vaughan, J.L. (1984). Concept structuring: The technique and empirical evidence. In C.D. Holley, & D.F. Dansereau (Eds.), *spatial learning strategies techniques, applications, and related issues.* Orlando, FL: Academic Press.

Weissman, C. (1967). *LISP 1.5 Primer.* Belmont, CA.: Dickenson.

Wenger, E. (1987). *Artificial intelligence and tutoring systems.* Los Altos, CA: Morgan Kaufmann Publishers, Inc.

West, L.H.T. (1985, May). *Concept mapping.* Paper presented in symposium: Perspectives on Cognitive Structure and Conceptual Change. AERA Annual Meeting, Chicago, IL.

Wittrock, M.C. (1963). Verbal stimuli in concept formation: Learning by discovery. *Journal of Educational Psychology, 67*, 484-489.

Wittrock, M.C., Marks, C.B., & Doctorow, M.J. (1975). Reading as a generative process. *Journal of Educational Psychology, 67*, 484-489.

Wittrock, M.C. & Twelker, P.A. (1964). Prompting and feedback in the learning, retention, and transfer of concepts. *British Journal of Educational Psychology, 34*, 10-18.

Woods, W.A. (1975). What's in a link? In: Bobrow, D. & Collins, A. (Eds.) *Representation and understanding: Studies in cognitive science*. New York: Academic Press,

Appendix A

Post Test
First Pilot Study

The following is a posttest administered to subjects in the first pilot study. The test consists of four sets of questions. Each set was designed to ascertain the subject's cognitive structures for a particular relationship.

The concepts being test are indicated in italicized text. These comments were not a part of the original posttest.

Sherlock Post Test

Please do not guess! Answer only if you are reasonably sure.

Name:

The following four questions pertain to the relationship between the concepts promise, consideration, and definite.

Which of the following statements is most accurate:
- a all promises are definite
- b all definite promises are consideration
- c all promises are definite and a type of consideration
- d A definite promise is a type of consideration

Which of the following could be consideration in an enforceable contract:
- a Sarah promises to return five library books this Friday
- b Bob promises to be good
- c Jim promises to try harder in the future
- d both a and c

Which of the following could be a promise:
- a "I will sell you ten apples for $1."
- b "I will try harder in class."
- c "I will be good."
- d all of the above

If something is a promise, then it must be:
- a lawful
- b a type of consideration
- c definite
- d none of the above

The following four questions pertain to the relationship between the concepts agreement and contract.

Which of the following is most accurate:
- a all agreements are contracts
- b all contracts are agreements
- c some contracts are agreements
- d both a and b

Which of the following is most accurate:
- a a contract is the same thing as an agreement
- b a contract and an agreement are very similar
- c a contract is one kind of agreement
- d an agreement is one kind of contract

If Bob and Sue have a contract, then they :

a must have an agreement
b must not have an agreement
c might have an agreement
d none of the above

According to the paragraph, what is the difference between an agreement and a contract?

a There is no difference
b in a contract the second party does something in return; this is not true in an agreement
c in an agreement the second party does something in return; this is not true in a contract
d none of the above

The following four questions pertain to the concept consideration

Which of the following is most accurate:

a There are several kinds of forbearance, one of which is not doing something that you have the right to do.
b Not doing something you have the right to do is the definition of forbearance.
c An indirect effect of a forbearance is not being able to do something that you have the right to do.
d none of the above

If something is not an act, a promise, or a forbearance...then:

a it cannot be a contractual consideration
b it can be another type of contractual consideration
c it has nothing to do with whether or not it is a contractual consideration
d none of the above

If the second party in a contract performs an act, then the act is a consideration if:

a it is a legal act
b if the act is not required by law
c both a and b
d none of the above

All agreements have:

a consideration
b two parties doing something
c something which is agreed on
d none of the above

The following four questions pertain to the overall relationship between the concepts in the text.

The thing which is agreed on in a contract is:

a that the first party promised to do something for the second
b that the second party does something for the first
c both a and b
d none of the above

John promised to send Mary to Europe this summer if she was good. Which of the following is the most accurate description of what they had:

a an agreement
b a contract
c an enforceable contract
d none of the above

Bob promised Jack that he would try harder. Which of the following is the most accurate description of what they had:

a an agreement
b a contract
c an enforceable contract
d none of the above

Jack said he would pay Sam $5 if he stayed away from the dance. Which of the following is the most accurate description of what they had:

a an agreement
b a contract
c an enforceable contract
d none of the above

If Mary and Susan have an enforceable contract, that it would also be true to say that they have:

a an agreement
b a contract
c both a and b
d none of the above

Appendix B

Data from Second Pilot Study

B.0. INTRODUCTION

20 UCLA undergraduates were video taped during 90 minute session with the Sherlock interface. In addition subjects were asked for suggestion which would like to see in an automated tutor for the Sherlock task. The data from this pilot study is divided into two sections: First, the suggestions the subjects had for future development; second, the problems they encountered in making links.

B.1. SUGGESTIONS

B.1.1. GENERAL REACTIONS

Answers to the question, "What general reactions do you have to what you have just done?"

> Too hard to go back and figure out what you meant by something. It is too confusing. It's easier to do things in step.
>
> It is interesting, but I don't understand why you are doing it.

> I definitely understand the concepts better now having made the map.
>
> Makes you think. Usually when I study I don't think.
>
> You need to read text over again, because otherwise you lose sight of the whole from dealing with just parts.

B.1.2. TUTORING SUGGESTIONS

We explained to the subjects that future plans included implementing an automated tutor to aid in the construction of graphic maps. The following responses were in answer to the question "What aspects of today's session do you think we should try and include in the automated tutor?"

> I like when something I did wrong was questioned, but not telling me the right answer
>
> I liked suggestions for things to link to the network. They were things I might not have thought of.
>
> The analogies were helpful.
>
> Too hard to go back and figure out what you meant by something [when you try and talk about mistakes I made some time ago]. It is too confusing. It's easier to do things in step [say something at the time that I make the mistake].
>
> Give new ideas.
>
> Make suggestions.
>
> Put up help icon when it thinks that I am in trouble [so that I have the option of getting more help].
>
> Clarify difference between PROP and PART [links].
>
> Let me know in the beginning if something is wrong. Let me try again several times. Let me know **why** something is wrong.
>
> Liked the use of analogies to show me why something was wrong.

B.1.3. DESIRED ADDITIONS

We asked subjects what they felt was missing in the interface. More specifically, "What extra links or icons would have assisted you in representing the consideration text?"

LINKS

> similar (not exactly equal, but close)
>
> partial definition

could be

or

ICONS

would have liked specific examples

conduct

thing

B.2. PROBLEMS

Below is a list of the links (or lack of links) which were considered mistakes during the sessions. They are classified as belonging to one of the three possible sources of confusion: text understanding, strategy usage, or icon interpretation. Following each link are the underlying thoughts (summarized from the protocols) which led to the link. Full dialogs are included in some of the places where an attempt was made to clarify the misunderstanding.

B.2.1. UNDERSTANDING

"AGREEMENT" EQUIV "ONE-PARTY-PROMISES" (MAYBE OK)

understanding problem. thought they were the same thing

"CONSIDERATION" PROP "AGREEMENT"

Past beliefs were causing interference. She believed that the only difference between a contract and an agreement is that the contract is written. This apparently caused her to mis-read the first paragraph, assuming that everything was referring to agreements.

t: Does that mean that all agreements have the attribute consideration?

s: I suppose all agreements would have consideration.

t: You agreed to do this study. Is there consideration?

s: I understand.

t: Would you like to reconsider that link?

s: I cannot think of anything I like better.

t: Consider the 3 icons: dog, animal, has 4 legs. What kind of a link would you put between animal and dog?

s: That would be a specialization link. From dog to animal.

t: What would you link has 4 legs up to?

s: To dog. And I suppose that would be a property link because dog because dog has 4 legs.

t: Now, would it be wrong to link 4 legs to animal?

s: Yes.

t: Why?

s: Because we don't have 4 legs, and I suppose we are animals. A chicken doesn't have 4 legs.

t: Now you have on the screen that contract is a specialization of agreement.

s: Oh, I see. we agreed that all agreements do not have consideration, just a certain type has it.

t: What would you call the type of agreement which has consideration?

s: I don't know. I don't know what I would call it. I guess I always thought that all agreements had consideration. Now they supposedly don't. It is an agreement by both parties I suppose, an exchange.

t: Let me ask you: would you say that all contracts have consideration?

s: Hmmm? I suppose they usually do, do they all? According to the text it says usually, that is not an absolute. So I suppose not all contracts are agreements, but that's another problem. No that is right, a contract is a type of agreement. So should I make consideration a property of contract?

t: What is the major point of the text?

s: It is taking an agreement and its breaking it into all of its parts. Seems like its trying to tell you about something you take for granted.

t: Explanation of paragraph.

s: Woops, so it is connected up to the wrong one. I suppose I did not understand it correctly.

"AGREEMENT" IS-A "CONTRACT"

Did not understand. thought that agreement was a sub-type of contract.

NO LINK BETWEEN "CONSIDERATION" AND "WHAT-IS DONE-IN-RETURN"

t: What link should go here?

s: Confusion.

t: Define consideration.

s: (after reading) They are the same thing. It is the definition. they are equivalent.

"FORBEARANCE" PART "CONSIDERATION"

Parsing problem. She was reacting to text "consideration may also consist of forbearance."

B.2.2. STRATEGY

"CONTRACT" EQUIV "AGREEMENT"

t:: Why did you make that link?

s: A contract is a written agreement.

t: Would you say that all agreements are contracts?

s: No, just written ones?

(This reflects an understanding problem, but should not have affected this link.)

t: Would you say that all contracts are agreements?

s: Yes, so it's like geometry. Therefore it would be some kind of subset. Maybe it is a part link? No, because all fingers are not hands. I know, it is a specialization link.

"NOT-ALREADY-REQ" EQUIV "CONSIDERATION"

Thought that equiv could be used even if it was just one part of a definition.

"PROMISE" EQUIV "CONSIDERATION"

Consideration can be one of 2 things. So consideration and promise are the same thing. Wait...that can't be because mathematically if a and b are equivalent to c, then a is equivalent to b, and promise cant be equivalent to act. Take out EQUIV. I will use PROP because they can both be the same thing.

"DEFINITE" EQUIV "PROMISE"

The promise is restricted to a promise which is definite (this was done by Ron while there was no PROP link).

L did the same thing. If x is a promise it must be definite. I will make it EQUIV. (She definitely understood text.)

"ACT" IS-A "PROMISE"

An act can be a promise. Was trying to say that consideration can be a promise or an act and thought this was a good first step. Consideration IS-A promise was the next step. At this point was very confused as to what it all meant.

"CONSIDERATION" IS-A "AGREEMENT"

As a result of an agreement you make a consideration. So consideration is part of the agreement. So I think it is a specialization. Consideration is a sub-class of agreement. like in an outline. A robin is one of many birds; consideration is one of many things which are in an agreement.

"CONSIDERATION" IS-A "PROMISE"

Consideration is a type of promise or it is a type of act.

"NOT-ALREADY-PROHIBITED" LEAD "PROHIBITED"

Consideration can make something not already prohibited, prohibited.

"CONSIDERATION" LEADS "ACT"

(And similarly for "promise" and "forbearance.")

Consideration will lead to an act a promise or a forbearance.

Gave ice cream analogy. Represented correctly, and felt the relationship was the same as in contract.

t: do you want to change it?

s: I still think it could lead to... but it doesn't always lead to all 3.

"PARTY1" LEADS "ONE-PARTY-PROMISES"

Knew that they were related but could not find the perfect link. The thoughts of party1 lead to the act. Party1 causes the action.

"PROMISE" LEADS "MARY-RETURNED "-(THIS COULD BE OK)

Mary fulfilled her promise. so promise would lead to Mary's act. Because promise means you have to fulfill your promise. (but link is going the wrong way) (r)

NO LINK BETWEEN "NOT-ALREADY-RECQ-BY-LAW" AND "REQUIRED-BY-LAW"

Understands but does not know how to say.

"NOT-ALREADY-REQUIRED" NOT "NOT-ALREADY-PROHIBITED."

Both are properties of consideration, but not at the same time.

"PROMISE" PROP "CONSIDERATION"

Understood. But this is better than EQUIV

B.2.3. ICONS

"ACT" EQUIV "WHAT IS DONE IN RETURN"

It says in the paper that they are the same thing.

"CONTRACT" EQUIV "PROMISE"

A contract is a promise. A contract is a promise in which someone does something in return. She was not real clear in her thinking on this.

THING AGREED TO

The thing which one person asks another to do in return for a promise.`

B.2.4. UNKNOWN SOURCE OF PROBLEM

"REQUIRED" PROP "ACT"

"MARY-DID-SOMETHING-IN-RETURN" IS-A "ONE-PARTY-PROMISES"

I assume that is something she promised to do. It is a specific example of a promise.

"NOT-REQUIRED" PART "CONSIDERATION"

B.3. LINKS

Below are the explanations given by subjects for each of the links. These explanations were later used in constructing the strategy Sherlock would use in building a graphic map.

B.3.1. PART

PHYSICAL PART

PART COULD ALWAYS BE A PROP

NECESSARY COMPONENT

X MAY BE A-OR B, SO A IS PART OF X (PART OF DEFINITION)

B.3.2. EQUIV

X IS DEFINITION OF Y

X MEANS Y

X IS Y AND Y IS X, A DOUBLE ISA

X IS PART OF THE DEFINITION OF Y

Was responding to the key word definition on the key sheet. Said that she did not really understand what equiv had to do with definition but would use it because there was not a better link.

X IS A RESTRICTION ON Y

Y must be equiv to a Y which has char of X.

B.3.3. IS-A

SUBSETS OF X

THEY ARE SUBSETS OF THE SAME THING

a is an x

b is an x

then a isa b

JUST ONE OF MANY THINGS WHICH COULD BE X

An example.

AN ACT WHICH HAS THE PROPERTY

There was no property link so Ron used is-a to say that consideration is a type of lawful act. Interpreted "lawful" as referring to lawful-act.

B.3.4. LEADS

WILL LEAD TO

THE THING THAT IS X WILL LEAD TO Y

MAY BE PART OF BUT NOT ALWAYS

B.3.5. NOT

THEY ARE DIFFERENT

OR

B.3.6. PROP

X IS A CHARACTERISTIC OF Y

IN ORDER FOR Y TO BE WHAT IT IS, IT MUST BE X

MUST BE

X HAS TO BE Y

Y SHOULD BE X

IT MUST BE SO I GUESS IT IS A PROP LINK.

B.4. INTERVENTIONS

These are descriptions of some of the interventions which were used when it was determined that a subject was having a misunderstanding.

B.4.1. TRYING TO FIGURE OUT HOW TO SAY THAT X CAN BE A, B OR C

Used three times. Successful every time.

> t: Ice cream icons. how would you link chocolate and ice cream?
>
> s: I suppose specialization.
>
> t: Do you think that that is at all analogous to what you are doing here?
>
> s: yeh, I suppose. ok. I just don't want to have a lot of the same kinds (all isa links). and if i do it like an outline, it will be all isa's. I guess that is ok.

recognized that act and forbearance had the same relationship.

B.4.2. EQUIV INSTEAD OF ISA

> t: Would you say that x is always a y?

s: No, but it could be.

t:: Would you say that y is always an x?

s: No.

t: Would equiv link be the best link?

s: I guess not. But we have no links that it may be. Ah-ha, we have specialization! Changed it to act isa consideration.

Appendix C

Annotated Protocols Final Study

Appendix C contains the full think-aloud protocols for the subjects described in Chapter 9. The following are the conventions used to identify the source for each paragraph:

Actual subject protocols will appear in the right column in the standard font.

Experimenter dialogue which was said during the study appears in the right column, idented, and in italics.

Comments presented to the subject by Sherlock appear in the right column in a non-serif font.

Description of visual information from the video tape appear in the left column in italics

Comments and analysis of the protocols appear in the left column in the standard font.

Numbers appear before each of the possible reasons presented to the subject when the program asks, "Why did you do that?" These numbers do not appear in the actual menu presented by Sherlock to the subject. They are included here only to clarify to which reason the experimenter or subject are referring.

SUBJECT: BECKY

LINK #1 8:OO

Moves CONTRACT over to middle top of screen.

I'm moving "contract" over because I want to get the most general word on top. Since I'm defining a contract, I'll start with "contract."

places the icons for written agreement and oral agreement beneath "contract".

Now contract is usually an agreement, so I will put oral and written

Oral agreement is a type of so that's isa link

LINK: "ORAL AGREEMENT" ISA "CONTRACT"

WHY DID YOU MAKE THAT LINK?

1. every oral agreement is a contract
2. it is not as likely that a contract is a oral agreement
3. oral agreement might be a contract
4. it is not as likely that an oral agreement is a contract
5. every contract is a oral agreement

No. 3 picked. No 2 picked. 2 and 3 were only choices picked. Done clicked.

PLAN FEEDBACK

No plan feedback because these are sufficient reasons for an IS-A link.

FACT FEEDBACK

This is an incorrect statement on Sherlock's part. Mistake was made because of the way that Sherlock defines this fact

You seem to believe that
an oral agreement might be a contract.
It is not as likely that a contract is an oral agreement.
I do not agree.

But it says "might be" So that's true. I still believe both of those are true.

would you like to remove that link?

No.

EVALUATION

Sherlock correctly rejected the link and was able to correctly diagnose the problem. There was the problem with rejecting a true fact, though. Also it would have been nice if Sherlock recognized this general problem and could comment on it.

LINK #2 8:23PM

It probably will say the same thing for written, but . . .

LINK: "WRITTEN AGREEMENT" ISA "CONTRACT"

I'm thinking that it's probably going to say that it doesn't agree., but I'm going to try it anyway.

why is that?

Because an agreement can be either a written agreement or an oral agreement, and according to this sheet a contract is usually an agreement. Which means oral or written.

WHY DID YOU MAKE THAT LINK?

1. every written agreement is a contract
2. it is not as likely that a contract is a written agreement
3. a written agreement might be a contract
4. it is not as likely that a written agreement is a contract
5. every contract is a written agreement

No. 3 picked.

Holding hands up in the air to represent concepts.

(Re: #2) So "contract" is here. "Written agreement" is here. "Written agreement" is larger. This is what confuses me. That concept is hard for me to comprehend. Since last time I did that one, I'll go for the other one.

picks 4

(Re: #4) I don't agree with that, but last time I said the other one, and it said that was wrong. So.

click done button.

PLAN FEEDBACK
With only #3 picked it should have been seen as a plan problem now.

FACT FEEDBACK

You seem to believe that
written agreement might be a contract.
I do not agree.

Why didn't it put both of them?

because the other one it agreed with.

It says, one party says to another . . . Well, I think you're right, then, computer because this text says "Says." but when you say something, it also write it, but . . . it doesn't say "one party writes to another." I'm going to say, you're right computer.

Just click on OK

would you like to remove that link?

Yes. It doesn't say anything here about having a written, so I will take it off. So, yes. Remove that link.. Maybe there's another one.

EVALUATION
Sherlock recognized the link as bad but was not able to diagnose the problem correctly.

LINK #3 8:30 PM

Written agreement "can be" a part of . . . but there's no "can be". So I'll put this aside. Put it back over there.

Now, contract is usually an agreement, so I'll make those equal

LINK: "AGREEMENT" EQUIV "CONTRACT"

contract is usually an agreement. So, . . because contract is usually an agreement. Well, actually it's not an equivalent. I changed my mind. Because it doesn't have to be. Because I happen to know that a contract is an agreement

WHY DID YOU MAKE THAT LINK?

1. the likelihood of a contract being a agreement is equal to the reverse.
2. every contract is a agreement
3. every agreement is a contract
4. contract and agreement refer to the same concept
5. the likelihood of a contract being an agreement is not equal to the reverse
6. contract is another name for agreement

(Re: #6) It's not another name. Contract is usually an agreement. That "usually" word is throwing me off.

(Re: #2) No.

(Re: #3) No, because it could be a written agreement which is not a contract., according to this paper. I could save the whole problem and abort this. I'm going to abort it.

Because the text says "usually," that means sometimes it's not. Well if I can go beyond the text to what I know, then I would say that No. 1 is true.

Pick No. 1, pick No. 2

No, I don't like that whole thing, so I'm going to abort it.

abort button clicked.

would you like to remove that link?

Yes.

EVALUATION
aborted

LINK #3B 8:34 PM

Maybe it's another kind of link. A contract is a type of agreement. Yeah That's better.

Tell me why you decided that.

Because it says a contract is usually an agreement. I don't think that a contract . . . an agreement is necessarily a contract. Because I could be talking with somebody and agree them, but we're not making a contract. If I make a contract with them, that means, according to this, that it would be an agreement.

LINK: "CONTRACT" IS-A "AGREEMENT"

very good. That is what I would have done.

EVALUATION
Link made for the right reasons and accepted by Sherlock.

LINK #4 8:35 PM

moves the "promise" icon over to the far right of "contract"

party1 promises to do something for party2 isa promise. So that's a promise. that means they're making a promise. So contract is a type of promise.

LINK: "CONTRACT" IS-A "PROMISE"

OK. According to this, contract is an agreement and meaning if you do this, I'll do that. and it says here that the think which one person asks another to do in return for a promise indicating that, if you do this I'll do that, is a promise.

I don't understand that.

Because it says here, "A contract is usually an agreement in which one person says I'll do this and you'll do that" and then the next sentence here is that "the thing which one person asks another to do in return for a promise" . . . So they're assuming that by saying, if you do this, I'll do that, they're assuming that that's the promise.

WHY DID YOU MAKE THAT LINK?

1. every contract is a promise
2. it is not as likely that a promise is a contract
3. a contract might be a promise
4. it is not as likely that a contract is a promise
5. every promise is a contract
6 there is no chance that a promise is a contract
7. There is no chance that a contract is a promise.

8:37 pm Picked no 3, picked no. 4.

Wait, are those contradictory? No. The only reason it's less likely that a contract is a promise, it's less likely that the reverse. But it still might be that a contract might be a promise.

Final choices 3 and 4 done button clicked.

PLAN FEEDBACK

Does not seem like a good reason for an IS-A link.

Sherlock incorrectly decided that she was using r-irrelevant. (bug)

FACT FEEDBACK

you seem to believe that:
a contract might be a promise.
I do not agree.

> *you said that's true. You believe that, right?*

Right.

> *do you think the computer's wrong?*

No.

> *You think the computer's right?*

Yes.

would you like to remove that link?

Yes

The link is removed. She takes "promise"

Oh, wait a minute. It liked just that one there. So, I'll leave it there. and make it a link.

EVALUATION

Sherlock recognized the link as bad but was not able to diagnose the problem correctly.

The plan was much more responsible than the facts.

LINK #4B 8:48

LINK: "CONTRACT" IS-A "PROMISE"

Interesting notion: the link is OK, but the facts that I gave to support it were a problem.

That's what you did before though

Yeh, but see before, I pressed both of those and it liked one of them but not the other one, so this time I'll just press the one.

WHY DID YOU MAKE THAT LINK?

1. every contract is a promise
2. it is not as likely that a promise is a contract
3. a contract might be a promise
4. it is not as likely that a contract is a promise
5. every promise is a contract
6. there is no chance that a promise is a contract
7. There is no chance that a contract is a promise.

Oh, wait a minute. That's not right. the thing another person asks a person to do in return ________________ the promise that is not the contract but the thing which one person asks another to do in return.

[reads]

Oh, wait, then they have no rela . . . that's like three relationships away. Promise, the thing which one person asks another to do in return for a promise . . . Promise is out here. that not

clicks the abort button.

would you like to remove the link?

Yes.

EVALUATION

aborted

LINK #5 8:51 PM

Party1 promises to do something for party2? Maybe. Party1 fulfills promise

the thing another person is asked to do in return a promise is consideration. Is there a consideration?

she points to "party1 promises to do something for party2"

That equals consideration.

LINK: "PARTY1 PROMISES TO DO SOMETHING FOR PARTY2" EQUIV "CONSIDERATION"

WHY DID YOU MAKE THAT LINK?

1. the likelihood of consideration being a party promise is equal to the reverse
2. every consideration is a party promises
3. every party promises is a consideration
4. consideration and party promise refer to the same concept
5 the likelihood of a consideration being a party promise is not equal to the reverse.
6. consideration is another name for party promises
7. it is not as likely that a consideration is a party promises

Picks #1

(Re: #1) has to be true. If they're equal, they're reversible.

picked no 2, no. 3, no. 4, no. 6 final choices are 1, 2, 3, 4, and 6. clicks done button.

PLAN FEEDBACK

No feedback on plan ok because these would have been good reasons to make an EQUIV link.

FACT FEEDBACK

you seem to believe that:
every consideration is a party promise.
The likelihood of a consideration being a party promise is equal to the reverse.
I do not agree.

I'm trying to say that, I think they're equal . . . but there are . . . wait. Consideration is a party promise. Maybe not. Wait. IF you do this for me, I shall do that for you. The thing which one person asks another to do in return for a promise is the consideration. So actually consideration may be the giving of another promise, or it may be an act. Consideration may also consist . . . I don't agree with that. I agree that I was wrong.

Ok How about the second statement?

The likelihood of a consideration being a party promise is equal to the reverse. . . I agree that that's wrong. I agree with the computer.

Were you aware that:
it's not as likely that a consideration is a party promise?
Would this change your choice of link?

were you aware of that fact?

Oh, I hate these "nots".

It's less likely

It's less likely that a consideration is a party promise. Now I'm aware of that. I didn't think about it before.

would you like to remove that link?

Yes. I want to change the link.

link removed

EVALUATION
Sherlock correctly rejected the link and was able to correctly diagnose the problem.

LINK #6 8:58 PM

The thing ___________ in return for a promise is a consideration. The consideration MAY BE giving of another promise or it may be an act. So it could be a couple of things.

So a consideration can be a party1 promise to do something for party2 or it can be something else. So this (points to "Party1 promises") can be. So is an attribute of . . .? Is just the same as . . . ___________ oh maybe it's that. Maybe it leads to.

The thing which one person asks another to do in return for a promise ___________ consideration may be the giving of another promise. try that one. Leads.

8:58 pm

No, maybe its a type of. It's an example of a consideration. Yeah. Cause there is a connection. A connection between those. I'm just not sure what's the right one.

pointing to PARTY1 PROMISES

therefore this is a type of. Can be a type of. But you don't have a lead for a "can be a type of" Try "is an example of"

LINK: "PARTY1 PROMISES" ISA "CONSIDERATION"

_____ it can be a type of.

what does that mean?

That means it's not necessarily. But according to this, it may be either a promise or an act. If this is a promise, it's possible . . . a consideration can either be this or this. So it's possible that . . . it's like consideration can result from A or B. So A or B can be . . . the result of . .. is a . . . "can be" is an example of [???]. "Can be an example of" is what I want.

Very good. That is what I would have done.

Without knowing what the actual contract is, then it's isa. But if I knew what the contract was, it might have been an act.

EVALUATION
Link made for the right reasons and accepted by Sherlock.

LINK #7 9:02 PM

Party2 does something in return. Something done in return for a promise. Party1 promises to do something for party2. Party2 does something in return. That's "is a result". Results in. Leads to. I guess. Party1 promises to do something for party2 leads to party2 doing something in return.

When it say Party 2 does something in return, am I assuming that they're saying . . . well. I'm assuming that they're saying in return for

something else. When I'm going to do my next lead.
see, that's what's more important.

9.04 pm

Something done in return for a promise.

A promise is a part of a contract. Maybe that works? these 2 things: Party2 does something in return, and Something done in return for a promise, is a consideration. I think. Something done in return for a promise. That's the consideration.

LINK: "SOMETHING DONE IN RETURN FOR A PROMISE" EQUIV "CONSIDERATION"

Something done. . . the thing which one person asks another ________ is the consideration.

very good. That's what I would have done.

what does the computer say?

It agrees. And I agree with the computer.

EVALUATION
Link made for the right reasons and accepted by Sherlock.

LINK #8 9:08 PM

So that again, act can be. May be an act. Consideration may be the giving of another promise or it may be an act. SO the giving of another promise, something done in re . . . the giving of another promise. Do they have the word "promise?"

Correctly identifies that there is some relationship between them. Now must determine what the relationship is.
click forbearance

So act and promise have some relationship to consideration. Consideration may also consists of forbearance, so forbearance

Interpreting the word "consist" literally as being a part.

and consideration have some sort of . . . consideration may consist of forbearance, so consist of is a part of.

LINK: "FORBEARANCE" PART "CONSIDERATION"

WHY DID YOU MAKE THAT LINK?

1. forbearance is a thing
2. a consideration usually has a forbearance
3. forbearance is a portion of something

Chose no. 3.

(Re: #3) Because it says a consideration may consist of forbearance.

click done

Well, now, forbearance . . . consideration may also consist of forbearance. Consist of means a part of. Wait a minute. Consideration may also consist of forbearance ____________ consideration. . . . is larger than forbearance. Yes, so forbearance is part of, well that's OK.

It won't like it, but

PLAN FEEDBACK
good plan

FACT FEEDBACK

You seem to believe that:
forbearance is a portion of something.
I do not agree.

Yeah, I agree with the computer because of the wording again. It's not a portion. It's something else. So I agree with that. I mean, it sounds . . . the way it's worded here, forbearance is a portion of something—and it's more consideration is a portion of something.

Consideration can be a part of forbearance, but it's not necessarily. Can be, but it doesn't have to be. And this says that it is.

Were you aware that:
every forbearance is a consideration
It is not as likely that a consideration is a forbearance.
Would this change your choice of links?

1. No I wasn't aware of that.

2. No I wasn't aware of the second fact either.

> *regarding the first fact, do you believe it?*

No. It still doesn't change my answer though.

would you like to remove that link?

link removed

Yes, but not for those reasons.

________ promise and act. I want to do something with those.

EVALUATION

Sherlock correctly rejected the link and was able to correctly diagnose the problem.

LINK #9 9:16 PM

So the oral agreement is party1 promising to do something for party2. Now what kind of link is that? I'm going to cancel that link I started to make.

So consideration is a relationship . . . so a relationship is a characteristic of relationship

LINK: "RELATIONSHIP" PART "CONSIDERATION"

A relationship is a characteristic . . . what did I press, "part" . . . but I don't think it's a part. It's more a characteristic of. There is a relationship in the consideration. A consideration, you know, some sort of relationship going on in the consideration. A is related to B in some way which . . . or B is related to A. So that's the relationship. And that whole thing is called the consideration.

WHY DID YOU MAKE THAT LINK?

1. relationship is a thing
2. a consideration usually has a relationship
3. relationship is a portion of something
4. there is no chance that a consideration is a relationship
5 there is no chance that a relationship is a consideration

picks no. 2, click done.
pointing to fact #2

What exactly does that statement mean to you—that consideration usually has a relationship?

That one part of . . . a consideration made up of 2 parts, A and B, A and B are related in some way.

PLAN FEEDBACK

FACT FEEDBACK

You seem to believe that a consideration usually has a relationship. I do not agree.

I think the computer's wrong.

would you like to remove that link?

Yes. I'm going to change it to property. I want to say that relationship is a characteristics of consideration.

link removed

EVALUATION
Sherlock correctly rejected the link and was able to correctly diagnose the problem. Might not be exact because there are several levels of problems going on in her thinking.

LINK #9B 9:25 PM

LINK: "RELATIONSHIP" PROP "CONSIDERATION"

WHY DID YOU MAKE THAT LINK?

1. A consideration usually has a relationship
2. A relationship is usually true for a consideration
3. Every consideration is a relationship
4. There is no chance that a consideration is a relationship
5. There is no chance that a relationship is a consideration

Why did I do that? I did that because relationship is a characteristic of consideration.

No. 1 and No. 2 seem to say the same thing to me.

Picks no. 1, so final is just No. 1.

PLAN FEEDBACK
Plan was reasonable given the fact that she agrees with

FACT FEEDBACK

You seem to believe that:
consideration usually has a relationship.
I do not agree.

Would you like to remove that link?

No.

EVALUATION
Sherlock correctly rejected the link and was able to correctly diagnose the problem.

LINK #10 9:28 PM

One cluster contains CONTRACT, AGREEMENT, and ORAL-AGREEMENT. The other cluster contains CONSIDERATION, RELATIONSHIP, PARTY1-PROMISES, and SOMETHING-DONE-IN-RETURN.

You know what you might want to try and do now, you have two separate clusters Why don't you try and link up those clusters by linking something in one of them with something in the other?

Party1 promises something to party2. That's a contract. Aha. Contract: party1 promises to do something for party2.
One of these is a contract

pointing to PARTY1 PROMISES and SOMETHING DONE IN RETURN

Party1 promises to do something for party2—that's the contract. Oh, wait, but it's only a contract if party2 does something in return. There's a stipulation there.

So that's a part of a contract. Part of. That's what I want.

LINK: "PARTY1 PROMISES" PART "CONTRACT"

very good. That is what i would have done.

EVALUATION
Link made for the right reasons
and accepted by Sherlock.

LINK #11

Now there's another part to it. Something that in return . . . Party1 has also promised . . . that'd be a part of the contract. _________ Party1 does something for party2. If you do this for me, I shall do that for you.

Something done in return for a promise. That's a part of a contract.

LINK: "SOMETHING DONE IN RETURN FOR A PROMISE" PART "CONTRACT"

Yeah. Party1 promises to do something for party2. Is part of it. Party2 has to do something for party1. Something done in return.

no feedback

> *It didn't give any feedback. Why do you think it didn't give you any feedback?*

Cause it's wrong.

> *Do you think that's why>*

Yes. But . . . What should I do now?

EVALUATION
Link made for the right reasons
and accepted by Sherlock.
No feedback because of network
change bug.

LINK #12 9:33 PM

something done in return for a promise is contingent on party1 promising to do something for party2.

Maybe it's leads to. Something done in return for a promise. Party1 promises to do something for party2 leads to something done in return. That leads to that. So this (party1 promises) leads to "something done in return". That leads to . . . something done in return.

LINK: "PARTY1 PROMISES" LEADS "SOMETHING DONE IN RETURN"

This results in that. If party1 promises to do something for party2, then, and if they carry it out, then something's going to be done in return.

***check feedback

EVALUATION

Link made for the right reasons and accepted by Sherlock. But no feedback.

LINK #13 9:35 PM

Consideration must be a legal act, so legal act is a characteristic of consideration. That means it has to be a characteristic of, because there's an "and" there.

LINK: "LEGAL ACT" PROP "CONSIDERATION"

WHY DID YOU MAKE THAT LINK?

1. A consideration usually has a legal act
2. a legal act is usually true for a consideration
3. every consideration is a legal act.

Consideration must (a) be legal . . . be lawful. Maybe legal act doesn't mean lawful. But it's not necessarily an act. It can be a promise or it can be an act. But you don't have the word "lawful". Legal obligation . . . maybe that's a better word. Legal act may be . . . consideration must be lawful. Now a legal act is lawful, but to be lawful is not necessarily an act. Consideration must be lawful. A legal obligation? Obligation? No. Lawful. Legal Act. Property? Is it property of? is it just like? Leads to? Is a type of? it could be part of.

That's that "can be" again.

So I'm going to abort the whole thing.

why are you aborting?

Cause it just __________ [warned?] me. And there's no "can be".

would you like to remove the link?

Know what I'm saying, cause a legal act is lawful, but a consideration is not necessarily an act. It can be an act or A promise.

Say that again.

Because consideration can be an act or a promise. And this is just an act. So this isn't necessarily . . . so this can be a property, but it's not necessarily a property.

OK, so you want to remove that link then? Is that what you're saying?

Yes. Like if it say Lawful, that would be better cause that could be either an act or a promise.

link removed

EVALUATION
aborted

LINK #14 9:39 PM

Party1 promises to do something for party2. Party2 does something in return. So this (PARTY1) leads to that (PARTY2). So party1 promise is contingent on party2 does something in return. It is contingent upon it. That's the link. Provided you do this (PARTY2), I'll do that (PARTY). Oh, Party2 does something in return leads to party1 promises to do something . . .No. Party2 does something in return that's contingent upon party1 promising to do something for party2. So it's either leads or an isa. It's got to be a leads. This leads to that. This is a leads.

LINK: "PARTY1 PROMISES" LEADS "PARTY2 DOES SOMETHING IN RETURN"

Because this is contingent on that.

what does that mean, "contingent"?

I mean like, if this is done, that will be done.

that is OK

Oh, no wait. Yeah. If this is done, that will be done.

EVALUATION
Link made for the right reasons
and accepted by Sherlock.

SUBJECT: JANEY

LINK 1

not clear whether believes that they are really equal, or just unclear on what equal would mean.

I am going to link contract to agreement as equal

It is true that if you make a contract you make an agreement, but that is not a good reason to make equiv link.

contract and agreement are the same thing. if you are make a contract with someone you are making an agreement for something.

LINK: "CONTRACT" EQUIV "AGREEMENT"

WHY DID YOU MAKE THAT LINK?

1. The likelihood of a Agreement being a Contract is equal to the reverse
2. every agreement is a contract
3. every contract is a agreement
4. agreement and contract refer to the same concept
5. the likelihood of a agreement being a contract is not equal to the reverse
6. agreement is another name for contract
7. It is not as likely that a agreement is a contract

In part of this I am applying it in general and in some of it I am applying it to this (specific situation).

an agreement is a contract and vica-versa in this specific situation

chose 1,4 ,5 & 7

her answers are confused. half indicate she believes they are equal (1&4), half indicate bad rule (5&7).

PLAN FEEDBACK

It would be nice if Sherlock would take into account that these sets of facts were contradictory. It would seem that the problem is more a matter of not having thought clearly about it, rather that believing or using this rule.

you seem to have used the plan:
If
it is not as likely that a Y is a X
the likelihood of a Y being a X is NOT equal to the reverse
the likelihood of a Y being a X is = to the reverse
then
make a EQUIV link from X to Y.
I do not recognize that plan.

The computer does not recognize my plan? I am not sure that I recognize my plan.
It doesn't look like it was very equivalent
It seems to be what I was thinking, but now that I look at it I don't agree with it.

FACT FEEDBACK

Were you aware that
every "Contract" is a "Agreement" ?

No, I was not aware of that.

do you believe the computer

yes. but i don't believe that every agreement is a contract. I think that that is what i was addressing before.
would you like to remove that link?
yes
now I don't have to link those 2 do I? Because I don't want to link those 2 now.

Unfortunately she is so frustrated at this point that she does not want to try and use this information to try another link here. She returns to this relationship in link #7 at my suggestion, but seems to forget everything that happened here.

EVALUATION

Sherlock correctly rejected the link and was able to correctly diagnose the problem.
Feedback good, convinced subject to remove link. did not know what to replace it with though.

LINK 2

Party1 promises I am going to move that over, I am going to link that as a part of the contract

LINK: PARTY1 PROM PART CONTRACT

When party1 promises to do something for party2 that is the start of the contract. Its part of the contract.

very good that is what I would have done.
oooh!

EVALUATION
Link made for the right reasons
and accepted by Sherlock.

LINK 3

now I am getting the hang of it.

Move over party2 does something in return (return).

I am going to link that to contract because it is the other end of the contract.

LINK: "RETURN" PART "CONTRACT"

so they are both part of a contract. They are the 2 parts of a contract.

very good that is what I would have done

EVALUATION
Link made for the right reasons
and accepted by Sherlock.

4:33

LINK 4

what I want to say is that "party1 fulfills promise" (fulfills) leads to... well ok we will use fulfills. Now

move over condition next to fulfills.
Interpreting condition as a role of contract.

Its part of the condition. Fulfills that a condition for the contract.

LINK: PARTY1-FULFILLS PART CONDITION

Party1 fulfills the promise that is a part of the condition of the contract. A contract has conditions under which they are carried out. Party 1 fulfills the conditions. its part of the contract.

WHY DID YOU MAKE THAT LINK?

1. Party1-Fulfills is a thing
2. a condition usually has a Party1-fulfills
3. Party1-fulfills is a portion of something
4. there is no chance that a condition is a Party1-fulfills
5. there is no chance that a Party1-fulfills is a condition

(Re: #1) I guess it can be. yes.
(Re: #2) YES

Picked 1 & 2
Her answers are reasonable given her interpretation of CONDITION.

FACT FEEDBACK
Sherlock is not seeing condition as a role of contract

You seem to believe that:
A condition usually has a fulfill promise
fulfill promise is a thing
I do not agree
I do believe that a condition usually has what you are calling a promise.
I think that the computer is wrong. I think condition does usually have a fulfill promise
Fulfill promise is a thing, that I don't really understand.

tell me what it means for a condition to have a fulfil promise;

(could not explain)

There is a condition of something in a contract, and that is what we are talking about, yes?

did not remove link

EVALUATION
Sherlock recognized the link as bad but was not able to diagnose the problem correctly.
There was a re-interpretation of the icon. Sherlock had no way of knowing that this is how she was using CONDITION.

LINK #5 4:44

Let's go with forbearance. I was first looking at these icons here, instead I am going to look at the links to try and link something together from these different links I have to choose from.

move over "consideration act" (ca) and "legal act" ("legal act"))

Now they are equivalent according to this. So I am going to use an equivalent link.

LINK: "CONSIDERATION ACT" EQUIV "LEGAL ACT"

Assumes the inverse without any evidence

according to this paragraph a consideration act is a lawful act. So a lawful act is a legal act. So I am going the links them as equivalents because they are both legal acts.

WHY DID YOU MAKE THAT LINK?

1 The likelihood of a "legal act" being a "consideration act" is equal to the reverse
2 every "legal act" is a "consideration act"
3 every "consideration act" is a "legal act"
4 "legal act" and "consideration act" refer to the same concept
5 the likelihood of a "legal act" being a "consideration act" is not equal to the revers
6 "legal act" is another name for "consideration act"
7 It is not as likely that a "legal act" is a "consideration act"

To me this is in general or specifically related to this. Because in general I wouldn't say that a "consideration act" is a "legal act". In general. But this is telling me that it is. Why are you looking at me funny? So I am answering these question in regards to the material here? yes.

(check #1)

(Re: #1) because it says here that a "consideration act" must be lawful. If it is lawful it must be equal and vica versa.
(Re:#2) I don't believe that it is, but ... I am not going to go with that, I don't believe that it is. But again, related to this, this says that it is. That is true according to the text.
(Re: #3) No, no. I am contradicting myself, I know that I am. But again ...

pointing to statement #2

Show me in the text where it says that every legal act is a consideration act

"Consideration must be lawful and require conduct that is not already required by law."

and that says that every "legal act" is a "consideration act"?

She seems to have trouble separating a fact like
every X is a Y from
every Y is a X. This is also evident in her reaction to the fact feedback in link 1.

well, it says that it must be lawful. That to me means that it must be a legal act. "Consideration MUST be lawful."

that is not the statement I am asking you about

oh, "every "legal act" is a "consideration act"" ok, no. but every "consideration act" is a "legal act", according to this.

Now she seems to understand that the 2 facts can be separate. How could Sherlock have diagnosed that? How could Sherlock have helped her to see her confusion?
(check #3)

(Re: #4) no. It says that a "consideration act" must be a "legal act", but it doesn't say that they are absolutely the same thing

read over several times)

(Re: #5)..so a "consideration act" would be a legal act but not necessarily a "legal act" a "consideration act".
(Re: #7) no, its not as likely as a "consideration act" being a "legal act".

(did not click it)

Am I done?

do you agree with that statement (#7) or not

putting different meanings on the facts.

No, I do not agree with that statement. Because from the text it says that a "consideration act" is a "legal act", but it doesn't necessarily say that a "legal act" is a "consideration act". So that's why I don't agree with it.

what does the statement say?

Oh yeah...you're right, I do agree with it.

When she is asked to think about them has no problem.
(checked #7)
(removed check on #1)

final choices 3 & 7
these are correct facts. But should have led to an is-a link.

PLAN FEEDBACK

You used the plan:
if
every x is a y
it is not as likely that a y is a x
then
make an EQUIV link from x to y
that is not a good plan

Sounds like algebra, and I am just terrible at algebra.

Do you think that that is the plan you used?

you're asking me?

(read it again)

I guess that is what it came it that I used, but it doesn't really make good sense. That is the plan I used, but it doesn't seem like it is a very good plan or that it is going to be equivalent.

At times like this I would make an IS-A link.

It means where I made the equivalent link? huh. Let's check that out.

EVALUATION

Feedback was perfect but she would not have received this feedback if I had not prompted her on the questions.

LINK #5B

trying to use Sherlock's suggestion to use an is-a link. Reads the definition of is-a and realizes she now has to decide what direction it goes in.

referring to list of rules

So, "legal act" we will... "consideration act" would be... would be a subset of a "legal act", or vice versa. Or do I have to decide that? I guess I have to decide that. Huh, OK. I would say that a... Well, according to this a "consideration act" is a type of a "legal act". So I am going to connect "consideration act"

LINK: "CONSIDERATION ACT" IS-A "LEGAL ACT"

Now tell me why you did that.

Well according to this a "consideration act" is a "legal act". It is one type, it is a type of a "legal act". So it's got to be legal. So... it's some type, or kink of "legal act".

Very good, that is what I would have done.

EVALUATION

Link made for the right reasons and accepted by Sherlock.

LINK #6 5:05PM

consideration-forbearance (cf), I am going to move that and I am going to link it to... leads to an act (t).

LINK: "CONSIDERATION-FORBEARANCE" IS-A "ACT"

consideration and forbearance are an examples of acts.

That might be true, but ... who cares?

I think, yes, it might be true also, but "who cares?" I guess it's not really important that its an act but it was ...

a link you found.

right, exactly, I mean you're right, who cares. But it seemed like a fairly easy link to make. OK, we're rolling.

EVALUATION

Link made for the right reasons and accepted by Sherlock. Feedback was appropriate.

LINK #7 5:09

I am focussing on "promise" right now. hmmm, well.
Let me focus on something else other than promise. And now I am actually looking at the icons instead of the links. So I am kind of switching back and forth in my approach.

go back to "contract" and "agreement" and try to link them

"agreement" is...a part of a "contract".

LINK: "AGREEMENT" PART "CONTRACT"

Interpreting AGREEMENT as an act.

An agreement is a main component of a contract. You are agreeing, 2 parties or whatever, are agreeing. And that's what makes up a contract.

WHY DID YOU MAKE THAT LINK?

1 "agreement" is a thing
2 "contract" usually has "agreement"
3 "agreement" is a portion of something

(clicked on # 1)

(Re: #1) yes an agreement is a thing. It's some type of thing you're agreeing to do.

clicked on number 2

(Re: #2) yes, that's what it is, it's an agreement between the parties.

comments as if agreement was a relationship.

(Re: #3) I don't really understand that. So I am not going to choose it.

1 & 2 chosen

PLAN FEEDBACK

no feedback

answers were consistent with PART link so no problem with link.

what does that mean? I think that means that ... I don't know what it means. How come it didn't give me any feedback? I think that it really doesn't care about that link. Before when it didn't care it told me that it didn't care. So I can't figure out why it didn't give me any feedback. Maybe its not a real important link.
It bothers me not to get any feedback because I was looking for another "very good."

FACT FEEDBACK

Sherlock first interpreted icons as AGREEMENT and CONTRACT. Link was not understood with this interpretation. Only one out of the 3 facts agreed (both Sherlock and the subject thought that #3 was false). Sherlock then tried interpreting "Contract" as MAKE-CONTRACT (the act of making a contract). Using this interpretation all of the facts agreed and the link was a good one.

A bug in the tutoring strategies failed to reconsider positive feedback at this time.

EVALUATION

Link made for the right reasons and accepted by Sherlock. But Sherlock gave no feedback.

LINK #8 5:16

Using PROMISE as a promise which is a consideration. Sherlock understands this.

If the consideration is a promise it must also be definite. so.. I am going to link promise and consideration. Promise is a type of consideration. Let me try that.

LINK: PROMISE IS-A CONSIDERATION

Very good! That is what I would have done. you're surprised. Why?

EVALUATION

Link made for the right reasons and accepted by Sherlock.

LINK #9 5:18

Icon interpretations consistent with Sherlock. Good plan. But based on prior belief that moral and legal are the same thing.

Let's go with "moral obligation" and let's try "legal obligation" Let's go with a LEADS-TO. Well I am thinking if something is moral it is most likely that it is legal. Let's go with the same, let's go with equivalent. I think moral and legal are somewhat the same.

LINK: "MORAL OBLIGATION" EQUIV "LEGAL OBLIGATION"

This one is a little shaky, I don't know about this one, but I will try it.

WHY DID YOU MAKE THAT LINK?

1 The likelihood of a "Legal obligation" being a "Moral obligation" is equal to the reverse
2 Every "Legal obligation" is a "Moral obligation"
3 Every "Moral obligation" is a "Legal obligation"
4 "Legal obligation" and "Moral obligation" refer to the same concept
5 The likelihood of a "Legal obligation" being a "Moral obligation" is NOT equal to the reverse
6 "Legal obligation" is another name for "Moral obligation"
7 There is no chance that a "Legal obligation" is a "Moral obligation"
8 There is no chance that a "Moral obligation" is a "Legal obligation"

Oh, I knew that was going to happen.

(picked)

(Re: #1) yes, I believe it is.

(Re: #2) no, I don't think everyone is, but I think most of them are.

(Re: #4) I think it is a matter of opinion, but... I better say no.

read several times

(Re: #5) No, I think it is equal to the reverse. well.

picked #7

(Re: #7) no, there is a chance.

please read them carefully

unpicked #7

final choice 1 only

PLAN FEEDBACK

no feedback

why no feedback?

Sherlock decided that the plan used was r-same-equiv. Facts 1,2, & 3 are the antecedents for this rule. She only chose #1. But r-same-equiv is the only rule which has #1 as an antecedent. Since the link she made matched the consequent for the rule, Sherlock decided that this was the best match.

Possible Solutions:

1. Have a buggy production which says that if the only thing you know is that there is an equal chance in each direction make an equiv link. This would have been the production matched, and then Sherlock would have known what to say.
2. Have Sherlock construct buggy productions on the fly by noticing that the matched production really did not have enough antecedents selected.

do you still like that link?

no. for 2 reasons. I trust the computer. When all of those questions come up it makes me think twice about it. And the second reason is that it didn't give me any feedback. So I don't think it was the best choice.

(removes link)

FACT FEEDBACK

No feedback on facts

When link was first made Sherlock assumed that "Moral Obligation" represented MORAL-OBLIGATION and "Legal Obligation" represented EXIST-LEGAL-OBLIGATION (the fact that a legal obligation exists). Using these interpretations Sherlock agreed on 6 out of the 8 facts. The two disagreements were facts # 7 & 8 above. Sherlock believed that there was no chance in either direction, but she had not indicated that she agreed with these.

Then Sherlock tried again interpreting "Moral Obligation" as EXIST-MORAL-OBLIGATION. Using this interpretation there is a small chance that the existence of a moral obligation implies the existence of a legal obligation. Thus Sherlock considered facts 7 & 8 to be false. There is now a perfect match between the learner's answers and Sherlock's, so this interpretation is used. Since there is no disagreement on facts, there is no feedback on facts.

EVALUATION

Link was bad because of a bad plan. Sherlock recognized that the plan was bad, but failed to recognize why the plan was the problem.

LINK #10 5:28

I am going to try and link consideration act and legal act to contract. But I am not going to do that because I don't really see the link.

Wait, do you think that they are related?

Well, I think they are related in regards to this.

Then do your best.

(consideration act has is-a link to legal act. both are near contract)

I am going to have to say property link because these things, consideration act and legal act, are characteristic of, or they are attributes of the contract. You need these things in order to have a contract. These are parts of. They are properties of whatever the contract is.

clicks on CONSIDERATION then ACT LEGAL ACT then is about to click on contract but link menu comes up.

oh, I thought I could link both of them.

No, you cannot link 3 things to another

So, I have to link just consideration act?

Whichever you want, but you can only link one..

LINK: CONSIDERATION ACT PROP LEGAL ACT

Ran out of tape at this point.

SUBJECT: JENNIFER

LINK 1 1:07 PM

Thoughts do not seem to lead to conclusion.

Think how . . . what would be the best one to do first. ________ Start . . . since the text deals with contract, I guess I'll start with the word contract. is .an agreement. It can be an oral agreement or a written agreement. Hmmm. It can be equal to an agreement. I'm going to say Is an agreement. So oral agreement isa contract.

LINK: "ORAL AGREEMENT" IS-A "CONTRACT"

because a contract . . . can be an oral agreement.

WHY DID YOU MAKE THAT LINK?

1. every oral agreement is a contract.
2. it is not as likely that a contract is an oral agreement
3. an oral agreement might be a contract
4. It is not as likely that an oral agreement is a contract
5. Every contract is an oral agreement

(Re: #1) No.
(Re: #2) NO it might be a contract

Picks no 3

(Re: #3) Yeah

(Re: #5) no

final choices 3 only.
This fact is true.

PLAN FEEDBACK

No feedback but it says Would you like to remove the link.

Yeah, because I also question

I think I will ____________

and put . . . change it. I guess an oral agreement is equivalent to a contract.

same bug as link s-4

EVALUATION

Sherlock recognized the link as bad but was not able to diagnose the problem correctly.

LINK 1B

LINK: ORAL AGREEMENT EQUIV CONTRACT

WHY DID YOU MAKE THAT LINK?

Or maybe it's a type of contract? Leads to a contract?

tell me what you're thinking

I think . . . __________ it's like. It's just like a contract.

It's a type of contract. That's not . . .. It's not like __________

it could lead to a contract. An oral agreement could lead to a contract

link aborted

LINK: #1C

LINK: ORAL AGREEMENT LEADS TO CONTRACT

First statement does not lead to the second. Plan is not good anyway.

A type of contract can be an oral agreement so if you make an oral agreement, you've made a contract.

WHY DID YOU DO THAT?

1. doing oral agreement will cause contract
2. if there is an oral agreement there will be a contract
3. a contract usually has an oral agreement
4. oral agreement can be a condition of contract
5. Oral agreement is usually true for a contract

picked no 1

(Re: #1) Yeah, doing an oral agreement will cause a contract.

final choices is 1 only

This isn't easy Ric.

PLAN FEEDBACK

No plan feedback because if the fact were true, it would have been a good plan.

FACT FEEDBACK

You seem to believe that:
doing an oral agreement will cause a
contract.
I do not agree.

Yeah. an oral agreement will make a contract. I think the computer's making a mistake.

EVALUATION

Sherlock recognized the link as bad but was not able to diagnose the problem correctly.

Could not diagnose it because she read facts to fit her thinking.

LINK 2 1:13 PM

Moved over the icon "party1 promises to do something for party2"

This is an example of a contract.

If this was true it would be a good reason for an is-a link.

LINK: "PARTY1 PROMISES TO DO SOMETHING FOR PARTY2" IS-A "CONTRACT"

WHY DID YOU MAKE THAT LINK?

1. Every party promise is a contract
2. It is not as likely that a contract is a party promise
3 A party promise might be a contract
4. It is not as likely that a party promise is a contract
5. Every contract is a party promise
6. A party promise is a portion of something
7. A contract usually has a party promise

Seems to be saying that she agrees with #1, but does not pick it.

(Re: #1) Every party promise is a contract. Because, well, I mean, if someone . . . cause a contract has to be __________ informally say well I'll do this and you'll do that, but it's not a contract.

Is she deciding then that she really should make a part link?

(Re: #2) No, because it is a part. That's what it is.

Selected No 3

(Re: #3) Yeah . . .

Selected No 5

(Re: #5) Yeah. Because party1 promises party2 something. It's a contract. A contract is a promise.
s
(Re: #6) Yeah, but it doesn't have anything to do with . ..

She defines "usually" as often but not always

(Re: #7) No, it always has a party promise.

So, Every party . . . . is a contract. Every party . . . it is not

Final choices 3 and 5

It's probably going to tell me that it thinks it's wrong. It could be an equivalent to because

[reads from paper]

What the paper says. it could also be equivalent.

PLAN FEEDBACK

You seem to have used the plan:
If
every Y is an X
X might be a Y
then
make an isa link from X to the Y.
I do not recognize that plan.

Would have been nice if Sherlock had realized that this was a backward IS-A plan.

Yeah. It is the plan I used. Yeah, it should recognize this plan. No, I guess I don't think it's a good plan.

Because isa kinda means that A is B and B is A and if you have the them might being, then it's not always.

FACT FEEDBACK

Are you aware that
a contract usually has a party promise
A party promise is a portion of something.
Would this change your choice of link?

1. Yes I was aware of that.

2. Oh, it's probably a portion of the contract

I didn't think like that.

> *OK, you were sort of aware of those facts before?*

Yeah.

> *So hearing them now would not change your choice of link.*

Yeah, it would.

> *How would it change it?*

Ummm. I'd probably use the partof

> *Ok, well, if you were aware of it before,*

Well, no. I thought I was aware of the first one, but the second one . . . I wasn't aware . . . Yeah, I wasn't aware.

> *Oh, I see. OK Do you believe it?*

A portion of something? A portion of the contract.

> *Yeah. So you were aware of it on some level, but just never thought of it. in those terms.*

Yeah

> *Now that you're think in those terms you're thinking maybe it could be*

maybe it can be

> *A part link?*

Would you like to remove the link?

Yes.

link removed.

EVALUATION

Sherlock correctly rejected the link and was able to correctly diagnose the problem. Impressive because it was result of faulty facts and bad plan, and Sherlock found both.

LINK 2B 1:26 PM

She was hinting that she was going to use a part link, now suddenly she uses a leads link.

I might try the leadsto. Party1 promising to do something for party2 leads to the contract.

LINK: "PARTY1 PROMISES TO DO SOMETHING FOR PARTY2" LEADS "CONTRACT"

__________ promising something to party2 _______ a contract . . . and that agreement lead to a contract between them.

that is OK.

So I'm kind of right. __________ for party 2.

EVALUATION

Link may have been made for the right reasons and accepted by Sherlock.

Sherlock determined that the antecedent for a LEADS link (X is a condition of Y) was true. The way it determines if X is a condition of Y is to determine if X is a child of a role of Y, and a descendent of CONDITION. CONS1 is a role of Y and is a descendent of CONDITION. In this case X is a condition for CONS2.

Since X is a condition for a part of Y, Sherlock determines that it could be seen as leading to Y.

LINK 3 1:27 PM

she's moving (SOMETHING DONE IN RETURN FOR A PROMISE) and (CONSIDERATION PROMISE) and (CONSIDERATION) starts to make a link from CONSIDERATION to SOMETHING DONE IN RETURN FOR A PROMISE. gets the menu and then starts reading over the kind of links that are available. Reads definition property

party1 promises to do something for party2 the thing which one person asks another person to do in return for a promise is consideration.

I think I'll use an isa link one more time.

LINK: "CONSIDERATION" IS-A "SOMETHING DONE IN RETURN FOR A PROMISE"

WHY DID YOU MAKE THAT LINK?

1. every consideration is a return for a promise
2. it is not as likely that a return for a promise is a consideration
3. consideration might be a return for a promise
4 it is not as likely that a consideration is a return for a promise
5. every return for a promise is a consideration
6. return for a promise and consideration refer to the same concept

Picked no 5

Can I change the link already?

chose the abort to get rid of this link. Why do you want to abort?

Is equal to. I think it defines something done in return for a promise they're equal. Because it's not. It says the thing which one person asks another to do in return for a promise is the consideration.

link aborted

EVALUATION
aborted

LINK 3B 1:32 PM

LINK: "CONSIDERATION" EQUIV "SOMETHING DONE IN RETURN FOR A PROMISE"

what specifically cued you, do you think?

When it said, when it asked the question "a consideration and a promise refer to the same thing" When you would think about it they are. You can either call it a consideration or you can call it a return for a promise. And you're talking about the exact same thing.

very good. That is what I would have done.

EVALUATION
Link made for the right reasons
and accepted by Sherlock.

LINK 4 1:32 PM

Sound thinking.

And consideration can be another promise or it can be an act, so . . .they're not equal because a consideration act is not a consideration, because a consideration can also be a consideration promise. So a consideration act is a type of consideration. So I guess I'll try the isa link again. ___________

LINK: CONSIDERATION ACT ISA CONSIDERATION

very good. That is what I would have done.

EVALUATION
Link made for the right reasons
and accepted by Sherlock.

LINK 5 1:33 PM

moved consideration promise over near the consideration icon

same thing for consideration promise.

LINK: "CONSIDERATION PROMISE" IS-A "CONSIDERATION"

very good. That is what I would have done.

Yea!

EVALUATION
Link made for the right reasons and accepted by Sherlock.

LINK 6 **1:34 PM**

Now we're going to see if we can link the act and the promise. I think a NOT link would be good. They're not the same. It's not an equivalent link, and it's not a property link. and one doesn't lead to the other. and they're not part of each other.

LINK: "CONSIDERATION ACT" NOT "CONSIDERATION PROMISE"

that is OK.

EVALUATION
Link made for the right reasons and accepted by Sherlock. Feedback was perfect.

LINK 7

Consideration may also consist of forbearance. Ohh. Forbearance. That is refraining from doing what one has a right to do. So, consideration . . . where's forbearance?

moves forbearance over to the other part of the screen near consideration act

and forbearance is a consideration

Oh wait, yeah, because forbearance can be a promise and an act, but we won't worry about that for now

LINK: "FORBEARANCE" IS-A "CONSIDERATION"

Feedback is more neutral this time because she use "forbearance" instead of "consideration-forbearance."

that might be true but, who cares?

[laughter] Well, maybe it's trying to tell me something.

Unfortunately she misunderstands the more neutral feedback. She probably did not see the "consideration-forbearance" icon and sees no difference between this and the previous 2 links.

maybe there's no link between them. I'm going to go on.

EVALUATION
Link made for the right reasons and accepted by Sherlock.

LINK 8 1:37 PM

moved "not going what one has the right to do" over near "forbearance"

Now forbearance is . . . not doing what one has the right to do. That is, I think they're the same. They're equal. You can use both terms.

LINK: "NOT DOING WHAT ONE HAS THE RIGHT TO DO" PART "FORBEARANCE"

look out something's not right

WHY DID YOU MAKE THAT LINK?

computer asks why did you do that. Subject says abort. Computer asks would you like to remove that link? Subject says yes.

LINK: #8B

LINK: "NOT DOING WHAT ONE HAS THE RIGHT TO DO" EQUIV "FORBEARANCE"

very good. That is what I would have done.

EVALUATION
Link made for the right reasons and accepted by Sherlock.

LINK 9 1:40 PM

Now, consideration must be lawful. IF the consideration is a promise, it must also be definite. OK, let's start at the bottom and move up. See if there's definite in here.

She is pointing at the CONSIDERATION FORBEARANCE icon.

What's this icon, consideration forbearance?

she's pointing to the other place where she used consideration

Maybe they aren't connected.

There's no "definite" so I won't do that. I'm trying to figure out where this consideration forbearance __________ [square?] goes. I guess something done in return for a promise can be consideration and forbearance. I mean, I don't know why they'd have those squares . .. but I don't like that square

moves consideration forbearance over near consideration
looking at CONSIDERATION FORBEARANCE.
moves the icon back away again.

It doesn't seem necessary, so I'll put it back over there.

consideration act can be an act.

moves over act underneath icon for consideration act. trying out all the possibilities for act and consideration act.

AN act is an attribute of, an act is a characteristic of. It can be an attribute, it could be equivalent.

OK. I think it's . . . .An act is an example of a . . . I think it's a property because an act is a characteristic of a consideration act

LINK: "ACT" PROP "CONSIDERATION ACT"

it's going to ask me why.

WHY DID YOU MAKE THAT LINK?

1. a consideration act usually has an act
2. act is usually true for consideration act
3. every consideration act isa act
4. consideration act and act refer to the same concept

1. It always has an act.

> *oh, wait, no, go ahead. it's just that if something always has, then it usually has. Right?*

But usually is like a might . . .

Would you say Ric usually wears clothes when he comes to UCLA?

But there, that leaves you the possibility that one day Ric might not wear clothes. But I think if he always wear clothes, there's no way Ric will every show up at school without any clothes on.

2.

3. Yes

picks no 3

4. Oh, I'm getting where I was before.

she says done with only choice No 3.
Her answer is consistent with an IS-A link which is what Sherlock would have done.

There can also be no links between them, right?

PLAN FEEDBACK

You used the plan:
If
every Y is an X
then
make a property link from X to Y.
that is not a good plan.

Her thinking matched one of the buggy productions so Sherlock was able to explain the problem clearly.

If all Ys have the properties of X, maybe Y isa X.

OK, that makes sense, though. That all Ys have properties of X. Then maybe Y is X. But I can't figure out what . . . is this Y and this X? Which is X and which is Y? Here.

looking at the plan, it gives you, how could you have possibly figured that out?

Every Y . . . . I don't think that's the plan I used. I didn't say that every Y is an X. I said that every X is a Y.

No, it can't be that either. Looking at that plan, I think it got it wrong, but I don't really remember what i said. If I did check that one, I was wrong.

FACT FEEDBACK

Were you aware that:

For some reason Sherlock is interpreting "act" as CONSIDERATION-ACT.

consideration act and act are different names
for the same concept?
Would you change your choice?

Yeah, Yeah.

where you aware of that?

No. Or was I?

would you like to remove that link.

No.

do you think it's true?

Do I think it's true? _____ ok, we'll try . . .

Yes, she does think it is true. she has removed the link

EVALUATION

Sherlock correctly rejected the link and was able to correctly diagnose the problem.

Actually the plan problem was correctly diagnosed, but the fact which came from the icon interpretation might confuse her.

LINK 9B 1:50 PM

LINK: "ACT" IS-A "CONSIDERATION ACT"

Because consideration act . . . maybe there's nothing more to it than just the act. Consideration may be . . . or it may be an act.

WHY DID YOU MAKE THAT LINK?

1. every act is a consideration act
2. it is not as likely that a consideration act is an act
3. act might be a consideration act
4 it is not as likely that act is a consideration act
5 every consideration act is an act
6 consideration act and act refer to the same concept.

before you answer the facts, look at the link you made and tell me in your words why you made that link.

An act is what happens when you have a consideration act. I mean, it's that . . . in the contract they , what they do is they give you the consideration and then say we'll have a consideration act and ______ can do is act. Do an act. He's not going to promise anything . . .he's going to sell a car or do something. And so, if a consideration act is doing something, then doing the act. An act is a consideration. Or maybe . . . they're equal.

Yeah.

why would they be equal?

She may be saying this now because of the wrong fact Sherlock just gave her.

Because you can say . . . you could say act or you could say consideration.
A definition. An act is a definition of consideration act. A promise may be . . . To define consideration act as an act. Yes. OK.

she aborted the question menu. It asks if she wants to abort the link. she says yes,

I'm going to try that.

EVALUATION
aborted

LINK 9C 1:52

did I try that before??

LINK: "ACT" EQUIV "CONSIDERATION ACT"

Sherlock liked it because it matched the icon interpretation it had for her. Is that bad?
Yes, but did she learn a good thing?

very good. That is what I would have done.

Oh, it liked that. That means promise is the same thing. Because. After All that I had to learn something.

EVALUATION
Not clear whether we should consider her link as appropriate or not.
Will count it as correct diagnosis.

LINK 10 1:53 PM
moves the promise icon underneath consideration promise

LINK: "PROMISE" EQUIV "CONSIDERATION PROMISE"

very good that is what I would have done.

EVALUATION
Not clear whether we should consider her link as appropriate or not.
Will count it as correct diagnosis.

LINK 11 1:54 PM

OK, a consideration must be a legal act. So a legal act is a property of consideration.

LINK: "LEGAL ACT" PROP "CONSIDERATION"

WHY DID YOU MAKE THAT LINK?

1. A contract usually has a legal act
2. Legal Act is usually true for contract
3. Every contract is a legal act

(Re: 1) I don't like the word "usually"

think of it as "always"

picks no 1, picks no 3

Oh. As "always?"

(Re: 3). I don't understand that.

final choices: 1 and 3

PLAN FEEDBACK
Sherlock determined that she was using r-maybe-prop. This is probably correct, so lack of plan feedback is appropriate.

FACT FEEDBACK

You seem to believe that
a consideration usually has a legal act.
I do not agree.

Yeah, that's something I believe. But I don't think that the computer's right.

Would you like to remove that link?

No.

EVALUATION
Sherlock correctly rejected the link and was able to correctly diagnose the problem.

LINK 12 1:57 PM

learner moves over PARTY1 PROMISE node towards that construction she has on the screen.

In the contract party1 fulfills a promise. ________ the party ________ the contract.
OR

inaudible stuff here at end

So it's the portion of the contract or . .. no, it's not an example of or type of . . . leads to . . . contract leads to party1 ________ promise.

pointing down from "contract" to "party1 fulfills promise"

I can go this way, right?

. . . contract leads to . . . characteristic property

Party1 is a part of the contract. That might be a ________ [type ?] link then. Party1 fulfills the promise.

LINK: "PARTY1 FULFILLS PROMISE" PART "CONTRACT"

no feedback from the computer on this one

It liked it, I assume. Or it has no comment?

______________ enough to argue with it anymore.

EVALUATION
Link made for the right reasons and accepted by Sherlock.
But, there should have been feedback. No feedback after network change bug.

LINK 13 2:00 PM

Written agreement. what are we going to do with this one? Let's try it again.

She moves "written Agreement" over next to "oral agreement"

written agreement is a contract.

LINK: "WRITTEN AGREEMENT" ISA "CONTRACT"

I won't worry about contradicting myself on oral agreement

WHY DID YOU MAKE THAT LINK?

1. Every written agreement is a contract
2. It is not as likely that a contract is a written agreement.
3. Written agreement might be a contract
4. It is not as likely that a written agreement is a contract
5. every contract is a written agreement

Picked no 1.

(Re: 5) I think this is right. Every contract is a written No, because a contract can be oral.

final choice: 1 only.

PLAN FEEDBACK

you used the plan:
If every X is a Y
then make isa link from X to Y
that is not a good plan.
That is too general
unless you also know that there is less chance that a contract is a written agreement, an equivalent link might be more appropriate.

This feedback is appropriate on the plan level. Unfortunately, because the fact that every X is a Y is false, she may misunderstand and think that Sherlock is suggesting that she make an EQUIV link here.

Well, that is the plan I used. Is it saying that unless you know it can be something else, [then] it should be equivalent. That's what it's saying. Right? ________________

OK.

Yeah, there is less chance because a contract can be an oral agreement. I bet equivalent won't be appropriate either . . . because

FACT FEEDBACK

you seem to believe that every written is a contract. I do not agree.

Yes, I believe that. _____________ find one written agreement that wasn't a contract, than _____________ off.

OK. Do you think such a thing exists.?

No. I don't think it does. Wait . . .

OK. Go ahead and __________

OK. Written agreement.

No, ________ agreement anything because I think you could take it to court and it would hold up.

Would you like to remove that link?

No.

EVALUATION

Sherlock correctly rejected the link and was able to correctly diagnose the problem.

LINK 14 2:06 PM

she moves "Agreement" over next to :"Contract"

OK. Contract and agreement.

Contract is equivalent to agreement. We have an agreement. We have a contract. It's equivalent.

LINK: "AGREEMENT" EQUIV "CONTRACT"

WHY DID YOU MAKE THAT LINK?

1. the likelihood of a contract being an agreement is equal to the reverse.
2. every contract isa agreement
3. every agreement isa contract'
4, contract and agreement refer to the same concept
5. the likelihood of a contract being an agreement is not equal to the reverse.
6. contract is another name for agreement.

because she thinks it's true but not sure	(Re 1) Not too sure of no 1.
picks no. 2	(Re: 2) it's why I make the link
picks no 3	(Re: 3) Oh, no. here we go again.
Picks no 4. and no. 6.	(Re: 4) Contract and agreement refer to the same concept.
final choices, 2, 3, 4, and 6.	
(offering only one choice)	Why did you do that? Agreement is usually true for a contract.
	No, I don't think it applies.
Clicks Done	
This should not have been asked at this point, a bug.	Would you like to remove that link?
	No.

PLAN FEEDBACK

	You used the plan: If Y is another name for X Y and X refer to the same concept every X isa Y every Y isa X then make an equivalent link from X to Y.
It would certainly seem that Sherlock should recognize this plan. The problem is that the learner has given antecedents from a number of plans which lead to an EQUIV link, and the network was not able to settle on which one she was using.	I do not recognize that plan.
	I don't see why it wouldn't recognize it. That's obvious. No more questions?

She is asked if she want's to remove the link.

NO.

FACT FEEDBACK

Did not comment on all of the bad facts because was unable to determine what plan she was using. Sherlock only comments on bad facts which actually led to the bad link. Did not comment on correct facts which would have led to a good link, because failed to see the correct relationship. This is a bug, Sherlock would have recognized the correct link if the EQUIV link had begun with "contract."

EVALUATION

Sherlock recognized the link as bad but was not able to diagnose the problem correctly.

LINK 15 2:10 PM

Party1 fulfills a promise and party1 . . . party2 does something in return. ____________ in order. is a contract. Is a part of a contract is that party1 fulfills a promise.

Move over the icon for "Party2 dos something in return"

Hmmm. results in. OK, party1 fulfills a promise . . . .which leads to party2 dose something in return.

LINK: "PARTY1 FULFILLS A PROMISE" LEADS "PARTY2 DOES SOMETHING IN RETURN"

Oh, wait a second. But what if party2 does something first? ________

(reads from the instructions)

cancels the link she was making

cause I think it might be the other way around. Because if party2 dose something in return, party1 fulfills the promise. No . . .

Party1 fulfills a promise which leads to party2 dose something in return. Let's go back and try what we were going to try before. Leads to.

Because I think party1 fulfills the promise when party2 dos something in return. SO like with the contract, party1 says, I'll so this if you do this, and party2 does it, and that makes . . . . oh but that's backwards.

OK, we'll try it the other way.

cancels the link she was making. Clicks on "Party2 dose something in return" and clicks on "party1 dos promise". clicks a link.

EVALUATION
aborted

LINK 15B

LINK: "PARTY2 DOES SOMETHING IN RETURN" LEADSTO "PARTY1 FULFILLS A PROMISE"

That is OK.

EVALUATION
Link made for the right reasons and accepted by Sherlock. Feedback is not perfect, though.

LINK 16 2:15 PM

wait. Something done in return for a promise, party2 dose something in return for a promise. So they should be equal

clicks on "party2 dos something in return for a promise" and then on "something done in return for a promise", then is perusing the different menu choices.

LINK: "PARTY2 DOES SOMETHING IN RETURN FOR A PROMISE" EQUIV "SOMETHING DONE IN RETURN FOR A PROMISE"

WHY DID YOU MAKE THAT LINK?

1. The likelihood of a return for a promise being a party2 return is equal to the reverse.
2. every return for a promise is a party2 return
3. every party2 return is a return for a promise
4. return for a promise and party2 return refer to the same concept
5. the likelihood of a return for a promise being a party2 return is not equal to the reverse.
6. return for promise is another name for party2 return
7. it is not as likely that a return for promise is a party2 return

picks no. 1

(Re: 2) No, it could do another promise, and in a sense, that would be a return for a promise. No, wait. That doesn't make sense. No. I don't think it has anything to do with it.

pick no 3
click no. 4. Pick no. 6
final choices: 1, 3, 4, and 6

PLAN FEEDBACK

You seem to have used the plan:
If Y is another name for X
Y and X refer to the same concept
every X is a Y
the likelihood of a Y being X is equal to the reverse
then make an equiv link from X to Y.

Same problem as in link #14

I do not recognize that plan.

IT doesn't recognize it. But it's OK, because that's what all these if's lead to.

FACT FEEDBACK

were you aware that
it is not as likely that a return for promise is a party2 return.
Would that change your choice of link?

No, I wasn't aware of that. I mean, I wasn't aware of it because I didn't think it was valid.

if you believed that were true, would it change your choice of link?

Yeah, because isn't it saying that . . . as likely that a return for a promise is a party link. ________ not likely that a return for a promise is a party link? Party link? No, party2 return.

Yeah, I would, but I don't think that. I mean, I still think they're equal.

> *not that they're equal , but that the chance they're equal in both directions.*

Yes.

> *OK*

EVALUATION

Sherlock correctly rejected the link and was able to correctly diagnose the problem.

Correctly diagnoses that the fact was part of the problem. But incorrectly thought that the plan was also part of the problem.

SUBJECT: STAN

LINK #1 1:19

Thought that they were related but since he could not find the right link decides that maybe they are not related

has a rule: if x is y then x is a prop of y. misreads the definition of prop link to support this.

read definition of PROP link:

He seems to be mis-reading the definition.

I guess I will start with the fundamental which is a contract and an agreement. Because that was the first sentence of what I read. A contract is usually an agreement, so they are not equivalent. It's not of something, although I suppose an agreement can be part of contract. I guess a contract can have an agreement language and maybe something else in it.

But I guess it's more like a property, well no, an agreement is not a property of contract. That is a hard one. They can't be equivalent, they don't necessarily lead to each other... I am not really sure that the two of them go together.

So... one thing I am sure is that a contract is a legal act. A contract is a legal act. So I guess a contract is a property of a legal act.

content of one is an attribute or a defining property of the class Well.... contract is definitely in the class of legal act so I am going to define this with a property link. contract is a property of a legal act.

LINK: "CONTRACT" PROP "LEGAL ACT"

WHY DID YOU MAKE THAT LINK?

These are all the reasons Sherlock would have made a prop link here (the third fact is from a buggy production). There are fewer than normal facts because Sherlock would not have made any link here..

picked #2

His wording is accurate, but it does not match the fact.

1 a "legal act" usually has a "contract"

2 "contract" is usually true for "legal act"

3 Every "legal act" is a "contract"

(re: 1) Woops, I wanted it the other way.

(re: 2) OK, I will take the middle one

A contract is usually a legal act.

PLAN FEEDBACK

gives no plan feedback because fact #2 would have been sufficient reason to make a PROP link if it were true..

FACT FEEDBACK

You seem to believe that:
contract is usually true for legal act
I do not agree.

Not every legal act is a contract but a contract is a legal act. I am not quite following the syntax. It is accurate that I believe that. But I think that the computer is making a mistake.

Were you aware that:
Every contract is a legal act
It is not as likely that a legal act is a contract
Would that change your choice of link?

It would not change my choice, it would change the direction. Which is apparently what the computer didn't like.

Are you saying that these facts are facts that you already know?

right.

But having them stated to you in this way has changed your mind?

He is applying the rules for an is-a link to the prop link. He seems to understand the relationship between them, just has the links messed up.

If I follow the property link as it is described in the materials, the more general one is the one that's being linked to. The specific one is the one where the link originates. legal act is the more general activity. So, according to the information I have I should be starting with contract and going to legal act. A contract by definition... unless there is such a thing as a contract which is not a legal act. Now I am assuming that every contract is a legal act. So according to the information I have I am right and the computer is wrong.

At this point all the computer is saying is that these facts are true, do you agree with that?

Because he assumes that the problem is direction and not link-type he never considers another link type. Thus he assumes that the directions are wrong on direction for prop link.

yes. I would deduce one of two things. Number one, the instructions here are not accurate. I think what is happening is the direction of the link more than the link. And either the directions I have here in terms of how you determine the direction of the link are incorrect, in which case the computer is looking for the other direction.

Or else, the second possibility is that I am just too stubborn to understand what the computer is trying to say.

would you like to remove that link?

no.

EVALUATION

Sherlock correctly rejected the link and was able to correctly diagnose the problem.

The problem seems to be at a very low level though, using different words for same ideas.

LINK #2 1:28

Has a correct understanding of relationship between contract and agreement. Good use of is-a link.

a contract is an agreement, I would say that those are essentially equivalent. Let me look at the definition of equivalence. But you can have agreements which are not contracts. So I would say that contract is a specialized form of agreement. The more generalized concept is agreement so we will start with contract. Its the specialized form of agreement.

LINK: CONTRACT IS-A AGREEMENT

Very good!
That is what I would have done.

OK. I got one right.

EVALUATION

Link made for the right reasons and accepted by Sherlock.

LINK #1B 1:30

Somehow seeing this IS-A link helps him see the source of problem in link #1

actually now that I look back at the first one (indicates the link between "Contract" and "Legal act") I am going to make this one an is-a. I am going to go back to that first one and change it again. Because it really is a sub-class.

(removes prop link from "Contract" to "Legal act").

And say that a "Legal act" and "Contract"... let's see the "Legal act" is the more general category, "Contract" is a type of "Legal act"

LINK: "CONTRACT" IS-A "LEGAL ACT"

PLAN FEEDBACK

That might be true,

I would have expected Sherlock to either interpret "Contract" as CONTRACT (in which case the link would be wrong) or as MAKE-CONTRACT (in which case it would be OK). In this case it used CONTRACT. But a small amount of activation always goes to alternate interpretations from the concept node. Usually this is too small to do anything. In this case MAKE-CONTRACT has a strong link from the concept node (since it is such a viable alternative) and there is a very strong link from MAKE-CONTRACT to LEGACT. This allowed a small amount of activation to reach LEGACT. Thus the conditions for an irrelevant link were met, and so Sherlock did not check the alternate interpretations.

But... who cares?

Well, you should compliment me that I realized what my mistake was and I corrected it. It should be telling me that first of all it is about time I figure out what was wrong. But "who cares," I don't quite understand why it is saying that.

FACT FEEDBACK

EVALUATION
Link made for the right reasons and accepted by Sherlock.

LINK #3 JUNE 21 1:32 P.

move oral agreement out.
move written agreement out
Not clear that he really believes that they are equivalent. More like he is just not thinking it through completely.

Contract is an oral agreement. I guess it can be. And a written agreement is also . . . that might be different. A written agreement I guess is synonymous with a contract. Ooops, OK. ______ it again. OK. A written agreement is a contract. So I will say that's an equivalent

LINK: . WRITTEN AGREEMENT EQUIV. CONTRACT

Now that he thinks about it realizes that it is not equiv.

Although come to think of it, it does not ________ have to be a legal contract.

Now . . . computer isn't going to like this one.

WHY DID YOU DO THAT?

1 The likelihood of a contract being a written agreement is equal to the reverse.
2 Every contract is a written agreement.
3 Every written agreement is a contract
4 Contract and written agreement refer to the same concept.
5 The likelihood of a contract being a written agreement is not equal to the reverse.
6 Contract is another name for written agreement

Oh, why did you do that? I can't agree with any of these

(picks abort)

Would you like to remove that link?

I guess we'd have to say that a written agreement is a subset of contract . . . So I guess you come back to an isa.

Answers yes to Remove the equivalent link

LINK #3B

LINK: WRITTEN AGREEMENT IS A CONTRACT

WHY DID YOU MAKE THAT LINK?

1 Every Written is a contract
2 It is not as likely that a contract is a written
3 Written might be a contract'
4 It is not as likely that a written is a contract
5 Every contract is a written
6 End of choices

Pick: Written might be a contract

(re: #3) Select that one

Select No. 4

(re: #4) because most contracts are written and yet there are many forms of writing

Done

PLAN FEEDBACK
The plan seems to be the real problem. These are not good reasons to make an IS-A link. See next link for an explanation of this problem

FACT FEEDBACK

This seems to be an incorrect statement.

You seem to believe that:
Written might be a contract
I do not agree.

Sorry, computer, I can't go along with that. So I';m just going to click on OK and I want to continue.

Would you like to move that link?
Noo.

EVALUATION

Sherlock recognized the link as bad but was not able to diagnose the problem correctly.

LINK #4 1:37 PM

This is true, but there is no link for that.

OK, an oral agreement can be a contract

I'll do the same thing for oral agreement. It's a sub-clause or it's a subset or an example, although one would not know it from the material that I read here.
I'm going to try and make that an isa also. Wether the computer likes it or not.

LINK: ORAL AGREEMENT ISA CONTRACT

This is an untrue statement. seems to understand the relationship but is placing his own interpretation on what it means to be a type of..

The oral agreement . . . it's a type of contract.
An oral agreement can be a contract. It's not necessarily a contract.
If it has the legal attributes described, then it is, in fact, a contract.

WHY DID YOU MAKE THAT LINK?

1. Every oral agreement is a contract.
2. It is not as likely that a contract is an oral agreement.
3. Oral agreement might be a contract.
4. It is not as likely that an oral agreement is a contract.
5. Every contract is an oral agreement.

I'm going to go through the same process as

. . . oral is a contract, that's clearly not true. We do many oral things. Like singing . . . that's not a contract.

Well, no, they mean oral as a shorthand for oral agreement

Oh, OK.

I would say,

Is every oral a contract" ? No, if it doesn't;t have the attributes discussed on this sheet, it would not be a contract. I;m assuming contract means a legal contract, or a contract with the properties described on this sheet.

Oral might be a contract. That's clearly true. That's the third one, but let's go in order.

It is not as likely that a contract is an oral . . . As an oral is a contract . . ..

I want my mama. Where are you when I need you.

It is not as likely that a contract is an oral, as an oral is a contract. Well, both of them could be equally likely or equally unlikely. So, enough of that

OK

Picks number 3.
This is true

Oral might be a contract. That's clearly true.

It is not as likely that an oral is a contract as that a contract... I think can be equally unlikely. Or equally likely. There is no basis for saying one is more likely than the other. I would think . . .. Let me think about it again.

It is not as likely that an oral is a contract . . .as a . . . contract is an oral. I don;t know. I have no way of making that judgment. Although it should parallel what I did before with written.

It's just that most contracts are probably written but then I;'m making an inference which is not in the information provided to me.. So . . ..

OK. Every contract is an oral . . . is clearly not true.

So the only I can, I feel that's not ambiguous given the information that I know personally, from personal experience, would be Oral might be a contract. That's all I can say.

Click the done button
the only choice is number 3

PLAN FEEDBACK
Ideally Sherlock would have commented on the basis for an IS-A link.
In this case there was enough evidence (in the facts which he said led to his link) for the spreading activation to determine that he was using a weak but legitimate rule: r-irrelevant. Since this rule is weak but not buggy, there was no feedback.
There should not have been enough evidence for this rule, since he did not indicate the other antecedent, "it is less likely that Y is X." This antecedent is more important for the rule than the antecedent which was chosen.

FACT FEEDBACK
No problem with the facts

Computer gives no feedback but asks if he wants to remove the link.

If you'd still like it, go ahead and say no.

NO.

EVALUATION
Sherlock recognized the link as bad but was not able to diagnose the problem correctly.

LINK #5

He's starting the next link while the computer's thinking.

Contract is synonymous with legal obligation

Move Legal Obligation over.

I think that's equivalent to a contract

LINK: LEGAL OBLIGATION EQUIV CONTRACT.

WHY DID YOU MAKE THAT LINK?

OK. It's asking Why did you do that?

1 The likelihood of a contract being a legal obligation is equal to the reverse
2. Every contract is a legal obligation.
3. every legal obligation is a contract
4. Contract and legal obligation refer to the same concept
5. The likelihood of a contract being a legal obligation is not equal to the reverse.
6 A contract is another name for legal obligation
7. legal obligation is usually true for a contract

Alright. I see.

The likelihood of No 1 . . . you can have legal obligations . . . Well can you have a legal obligation that is not a contract? Yeah. Ok. Clearly.

OK. I am wrong.

You can just say abort if you want

OK.

Aborted the Why did you do that menu.

removes the link.
Sherlock see a legal obligation as a result of a contract. But he is using a common notion that a contract is a legal obligation.

Would you like to remove that link,
Yes, and so we're going to have to say that a contract is a subset of a legal obligation. Because you have laws and you have contracts.

LINK: #6

LINK: CONTRACT ISA LEGAL OBLIGATION

Why did you do that?

1. every contract is a legal obligation
2. It is not as likely that a legal obligation is a contract
3. A contract a might be a legal obligation
4. It is not as likely that a contract is a legal obligation
5. Every legal obligation is a contract
6. If there is a contract, there will be a legal obligation.

Pick number 1
Pick number 2.

(Re: #3) That's not true, it's always a legal obligation if it's a contract.

(Re: # 4) That's not true because It's always a legal obligation. Every legal obligation is a contract. That's not true. If there is a contract, there will be a legal obligation, that is true.

Pick no. 6, so final choices are 1, 2, and 6. Clicking the done button.
First 2 are good reasons for an IS-A link, but not literally true. Fact #6 is true, but not a good reason for an IS-A link.

PLAN FEEDBACK
Plan seems to be good except for the extra fact (#6).

FACT FEEDBACK
This could be seen as an icon problem. He is using the CONTRACT icon to represent the result of a contract.

You seem to belive that
every contract is a legal obligation,
It is not as likely that a legal obligation is a
contract.
I do not agree,.

> *So the computer is right, that those are the things you thought.*

Yep. I think the computer is making a mistake.

Would you like to remove that link?

Definitely not.

EVALUATION
Sherlock recognized the link as bad and made a reasonable diagnosis of the problem. Actually however the subject seems to be interpreting "contract" loosely as the result of having entered into a contractual relationship. If Sherlock had recognized it as an interpretation problem, the subject might have accepted the computer's opinion.

LINK #7 1:47 PM

Got lot's of isa's here. Let me find something else that's not an isa. A consideration is a part of a contract. Oh boy, OK. We'll do something else.

Move Party One icon over.

Not going to say who party one is, huh?

Moves party 2 over.

OK. Party 1 is a part of a contract.

LINK MADE: "PARTY1" PART "CONTRACT".

Party 1 is part of a contract. That's part of the language of the contract, that's part of the agreement. You have to have two parties, at least.

No feedback

It doesn't tell me anything, whether it agrees or not.

EVALUATION
The link was good and was recognized by Sherlock. Unfortunately the program failed to give feedback due to a bug. The feedback would have been "That is OK."
Sherlock changed its network to reflect that the he was using "party1" to refer to the PARTY1 role of CONTRACT. After doing this the goal to give feedback was accidentally popped.

LINK 8 1:50 PM

LINK: PARTY2 PART CONTRACT

Once again , no feedback

LINK 9 1:51 PM

Let's see, I think consideration forbearance. Oh, we have consideration forbearance together. Terrific . Not doing what one has the right to do would be an example of consideration or a forbearance.

Seems to be equating consideration and forbearance.

The question is, is it equivalent to, subset of, I guess it's another isa. You see, you can do what you have the right to do without it being part of a contract. So I'd have to say that not doing what one has the right to do is an isa of consideration forbearance. Or is it the opposite way around. No, it's the other way around. Consideration forbearance is a subset of not doing what one has the right to do.

LINK MADE: "CONSIDERATION FORBEARANCE" IS-A "NOT DOING WHAT ONE HAS THE RIGHT TO DO".

Very good. That is what I would have done.
Oh.

EVALUATION
This is a bad link which Sherlock accepts. He made the link in the opposite direction he intended and has a unique interpretation of "consideration forbearance" icon. But combination led to a correct link.

LINK 10 1:53 PM

What do we have left? We have Act. Let's see, how is act defined in the text here. Consideration may be in giving an act, or it may be an act. So an act is also a subset of consideration [or] forbearance, so we're going to say

Move the Act icon over next to consideration forbearance Is He using ACT to represent acts which are a type of consideration.	Act
He is interpreting the icon as consideration or forbearance.	is a subset of . . . consideration forbearance. So.

LINK MADE: "ACT" IS-A "CONSIDERATION FORBEARANCE".

He's started thinking about the next link,	——— could be giving it a promise, Also.
	Or it could be the giving of another promise. Not just a promise, but the giving of a promise. Let's see if we have the giving of a promise. Yes, we do.
Back to the last link. It's now asking why did you do that?	Why did you do that? 1. every act is a consideration forbearance 2. it is not as likely that a consideration forbearance is an act 3. an act might be a consideration forbearance 4. it is not as likely that act is a consideration forbearance 5. Every consideration forbearance is an act
He is being inconsistent. If he were using ACT to represent a type of forbearance he should have said yes here.	(Re: #1) That's not true. (Re: #2) It is not as likely that a consideration forbearance is an act as an act is a consideration forbearance. It would be the other way around.
	I don't think that's true.
Picks No. 3	(Re: #3) Act might be a consideration forbearance. That is true
Picks No. 4	(Re: #4) It is not as likely that an act is a consideration forbearance as a consideration forbearance is an act That's true too. You can do many kinds of acts, but there are only 2 classes of consideration forbearance, one of which is an act. Every consideration forbearance is an act. That's not true. *Ok, explain to me again the second one you clicked.*
He clarifies here that he is using ACT as a generic act. So problem is misunderstanding about what a type is again.	It is not as likely that an act is a consideration forbearance as a consideration forbearance is an act. There are millions of different kinds of acts one can engage in. Whereas consideration forbearance has only two classes: it can be an act or a promise.

Final choices are 3 and 4. Done button clicked.

_____________ something like consideration act and consideration promise, you know, it depends what dimension you're talking about.; the dimension of it is a consideration, they are the same. But there are different forms of consideration. I don';t know . . .

PLAN FEEDBACK

Part of the problem is when to use is-a.

FACT FEEDBACK

You seem to believe that
act might be a consideration forbearance
I do not agree.

Sherlock does not agree because using its understanding of CONSIDERATION FORBEARANCE it cannot be a generic act, or even act which is a subtype of consideration.

Consideration may be an act. Something you do. So I'm assuming that the converse is true. An act is a consideration

OK, so the computer statement is true . . . it is what you believe?

Yeah.

OK., The computer's saying it doesn't agree. Do you accept that or do you think it's wrong?

I disagree with the computer
Because the act is the consideration.

subject is obviously using consideration forbearance to refer to . . . he keeps thinking of it as consideration. He somehow thinks they are the same thing, overlap, or whatever

OK. And the computer's saying something different, you believe?

Yeah. Well, it's saying it doesn't agree with that. But I don't understand why it would disagree with that.

Computer asks would you like to remove that link? Subject answers No.

EVALUATION

Sherlock recognized the link as bad but was not able to diagnose the problem correctly.

Major problem is the learner's interpretation of icon again, but Sherlock does not see this. It is also a bad plan (same plan has been used a number of times) but Sherlock does not recognize this.

LINK 11 1:59 PM

Moves Illegal Act over near Legal Act

OK, I;m going to take that "chippie" now to build up my confidence again

And say a Legal Act . . . and Illegal Act and use the Not link cause they're just opposites. That I am sure of.

If the computer gives me a hard time on this one . . . I'll give up.

LINK MADE: "LEGAL ACT" NOT "ILLEGAL ACT"

That is OK

OK. Very impressed.

EVALUATION

Link made for the right reasons and accepted by Sherlock. Feedback was perfect.

LINK: 12

OK. You can also say something trivial that a forbearance is a subset of consideration forbearance, but I;m not going to bother doing that. Would it be a subset? No, actually it would be the same as. That's trivial.

Moves "Party 2 does something in return" near "consideration forbearance"

OK. Party 2 does something in return. A consideration forbearance.
That's what it is. So, I;m going to try to say that those two are equivalent

It doesn't;t say in return for what. I;m going . . . does something in return for what Party 1 did.

Is that a valid assumption? So those two are equivalent.

LINK MADE: "CONSIDERATION FORBEARANCE" EQUIV "PARTY2 DOES SOMETHING IN RETURN"

I say equivalent because I think that's what it is. . .. what consideration forbearance is.

WHY DID YOU DO THAT?

1. likelihood of a consideration forbearance being a Party2 return is equal to the reverse
2. every consideration forbearance is a party2 return
3. every party2 return is a consideration forbearance
4. consideration forbearance and party2 return refer to the same concept
5. the likelihood of a consideration forbearance being a party2 return is not equal to the reverse
6. consideration forbearance is another name for party2 return.

likelihood of a consideration forbearance being a party2 return is equal to the reverse. I would say yes. It;s the same thing.
Every consideration forbearance is a party2 in return. The only problem I have in forbearance can be . .. I guess party1 doing something in return for party2. Right? So, ... Alright, that's why it doesn't;t like it.

Menu aborted. Computer asks would you like to remove that link? Subject answers Yes

OK, I'm going to abort this and just make it another subset.

LINK: 13

__________ party2 is the ISA again. Oh yeah, it does that, it leads to a consideration forbearance.

[Before ?] you do something in return, does that lead to . . . no, that doesn't;t lead to a consideration. That is a type of consideration. So it's going to be an ISA

LINK MADE: "PARTY2 DOES SOMETHING IN RETURN" IS-A "CONSIDERATION FORBEARANCE"

Only problem is that he is still using CONSIDERATION FORBEARANCE as consideration. If Sherlock knew that, then the link would be ok.

WHY DID YOU DO THAT

1. every party2 return is a consideration forbearance.
2. it is not as likely that a consideration forbearance. is a party2 return
3. Party2 return might be a consideration forbearance.
4. It is not as likely that a party2 return is a consideration forbearance.
5. every consideration forbearance. has a party2 return

He is trying to figure out why Sherlock did not like the link. He thinks that it has something to do with knowing the legal roles for parties 1 and 2.

So it doesn't like this either. I'm assuming now that party1 does something . . . I don;t know the legal language—how you refer technically to what party1 and party2 is. the party1 . . . I don't have enough information.

In a formal contract, is party1 the originator and the forbearance is on party2 . . . this has to do with the party1 or it could be with the ____________. This party1 or party2 has special meanings.

. . . all right, Every party2 re . . . Next time I make a plan, I;m going to be on . . .

(Re: #1) All right. Every party2 return . . . something return . . . is a consideration forbearance.. ummm Yeah.

Picked #2

(Re: #2) It is not as likely that a consideration forbearance. is a party2 return. Correct, because I guess it could be a party1 return

Picked #3

(Re: #3) Party2 return might be a consideration forbearance. That's true

(Re: #4) It is not as likely that a party2 return is a consideration forbearance.as a consideration forbearance. is a party2 return. No, it's just the opposite. SO I;'ll say no on that.

That is . . A party2 return is definitely a forbearance. but a forbearance is not a party2 return and ______________.

(Re: #5) Every consideration forbearance. is a party2 return. That's not true.

Final choices are Nos. 2 and 3. Answers are sort of consistent with his interpretation of the icon. But should have been that every party2-return is a consideration.

FACT FEEDBACK

You seem to believe that:
it is not as likely that a consideration forbearance. is a party2 return.
I do not agree.

Sherlock actually found that the chances were small (1) but equal in both directions.

I disagree with the computer because I'm assuming

that's something that you believe, right?

Yep. The computer's right. That's something that I believe. And when the computer says it does not agree, I disagree with the computer' statement because what party1 does in return could also be consideration forbearance.

Unless I;m told who initiated the contract.

did not change the link

So I;'m going to click on OK

EVALUATION
Given his interpretation of the icons it was a reasonable link. But Sherlock was still unable to identify interpretation.

LINK 14 2:07 PM

Party1 fulfills a promise that leads to what?
I'm going to work backwards here. I'm going to see what can I find . . . I've done a partof, I've done a isa, I've done plenty of isa's. You can't fault me on that.

where one leads to the other. Party1 fulfills a promise leads to what?

it doesn't lead to a contract. The thing which one person asks another to do in return for a promise is consideration. So those should be equivalent. Oh, I see where I made the mistake before with . . . forbearance means not doing what someone has the right to do,.

Seems to be figuring out that he has been using CONSIDERATION FORBEARANCE incorrectly.

Let's see, do we have forbearance? Yeah. We don't have not doing what someone has the right to do, do we. Oh, yeah, there it is.

So forbearance and not doing something someone has the right to do are equivalent.

LINK MADE: "FORBEARANCE" EQUIV "NOT DOING SOMETHING SOMEONE HAS THE RIGHT TO DO"

Very good. That is what I would have done.

EVALUATION
Link made for the right reasons and accepted by Sherlock.

LINK 15

He is pointing to the link he made between act and consideration forbearance.

OK, I think I misunderstood what forbearance is. consideration forbearance. are what you do in return, so that . . . this is interesting. . . . a subset of that

Now he's pointing to the link he made between party2 does something in return

this one . . . .______________ except for the party1 /party2 issue.

for a promise. I'm only saying that it's a subset because because it could be something that party1 also does.

> *please define the concept represented by the icon that says consideration forbearance.*

You do something in return.

> *so that icon stands for the concept of doing something in return.*

But it doesn't necessarily have to be party2 that does something in return, it could be party1 doing something in return for party2.

pointing to the icon CONSIDERATION

All right, there's an icon above the ACT icon. What is that icon?
ACT?.

no, above that.

Consideration.

What does that represent?

Consideration would be a subset of consideration forbearance, that would be an isa.

Well, what is it, itself? Define . . . tell me the concept that that icon represents.

You make a promise in return for what the other person is doing for you. You do something for me, the consideration is I make a promise to you.

So consideration is making a promise , which is one type of consideration forbearance., which is doing something in return.

Right.

So I guess promise would be an isa of consideration.

Do you want to make that link? [you can quit, etc}

Promise moved over near consideration.

OK. Promise is a subset of a consideration.

LINK: "PROMISE" IS-A "CONSIDERATION"

Very good, that's what I would have done.

EVALUATION
Link made for the right reasons and accepted by Sherlock.

LINK 16 2:16 PM

It doesn't appear anywhere that contract is a form of relationship.

LINK: "CONTRACT" IS-A "RELATIONSHIP"

very good. That is what I would have done.

EVALUATION

Link made for the right reasons and accepted by Sherlock.

LINK 17 2:17 PM

A consideration act is something that is done in return for a promise.

You do something in return for a promise is the consideration.

Something done in return for a promise, I guess, leads to consideration.

No, it's the .... something done in return for a promise is a consideration.

He does not ask the reciprocal question: Can you do something in return for a promise that is not a consideration act?

Can you do something that's a consideration act that's not a [thing done in return for a] promise? and I have to say no, so I think these are equivalent.

Is he using CONSIDERATION ACT now as the generic consideration?

LINK: "CONSIDERATION ACT" EQUIV "SOMETHING DONE IN RETURN FOR A PROMISE"

WHY DID YOU MAKE THAT LINK?

1. the likelihood of a consideration act being a return for a promise is equal to the reverse.
2. every consideration act is a return for a promise
3. every return for a promise is a consideration act
4. consideration act and return for a promise refer to the same concept.
5. the likelihood of a consideration act being a return for a promise is not equal to the reverse.
6. A consideration act is another name for return for a promise.

(Re: #1) Acting in return for a promise is a . . . well it's a return. . . the question is , in a sense, whenever you make a promise, you have an oral contract. So I guess the likelihood of a consideration act being a return for a promise is equal . . . to the reverse.

Picks No 1

Yes. I believe that's true.

Picks number 2

(Re: #2) Every consideration act is a return for a promise. Yes.

Picks No. 3

(Re: #3) Every return for a promise is a consideration act. I would say yes.

So although he did not ask the reciprocal question above, he believed it was true. So his plan was appropriate.

Picks No 4

(Re: #4) A consideration in return for . . . refers to the same concept. Yes.

(Re: #5) The likelihood of that's not true.

Picks no 6

(Re: #6) ...is another name for a return for a promise. Yes.

clicks done.
final choices are 1, 2, 3, 4, and 6

PLAN FEEDBACK
No plan feedback because all of his facts are good reasons to make an EQUIV link.

FACT FEEDBACK

You seem to believe that
every return for a promise is a consideration act.
The likelihood of a consideration act being a return for a promise is equal to the reverse.
I do not agree.

OK. I agree with both those statements, so it knows what i think.

It says I do not agree. and I disagree with the computer based on the assumption that any promise that you make and then fulfill. . . that you make becomes an oral contract.

would you like to remove that link?
NO.

EVALUATION
Sherlock correctly rejected the link and was able to correctly diagnose the problem.

LINK 18 2:22 PM

________ want to say that a contract leads to a moral obligation.

LINK: "CONTRACT" LEADS TO "MORAL OBLIGATION"

WHY DID YOU MAKE THAT LINK?

1. doing contract will cause moral obligation
2. if there is a contract there will be a moral obligation
3. A moral obligation usually has a contract
4. a contract can be a condition of moral obligation
5. A contract is usually true for moral obligation
6. there is no chance of moral obligation with a contract
7. There is no chance of a contract using moral obligation

picks no 1

(Re: #1)..Contract will cause moral obligation.
Yes.

picks no 2	No 2. Yes
	(Re: #3) not necessarily
[running out of tape]	

Appendix D

Sherlock: the Program

Sherlock is located in the directory /ucla/ai_research/feifer/Sherlock5 on the Apollo COGNET network at UCLA (cognet.ucla.edu on the Arpanet).

What follows is a listing of the program and the memory files needed for the consideration text. What is not included is the network code written by Michael Gasser and the graphics code written by Seth Goldman. This support code is accessible from the directory listed above.

```
(herald ric_menu
        (env t (sys/t_flavors flavors))
        (read-table (*value user-env '*flavors-read-table*))
        (syntax-table (env-syntax-table user-env)))

;****************************************************
;****************************************************
;**                                                **
;**      THESE FUNCTIONS ARE USED IN               **
;**       COMMUNICATING WITH THE LEARNER.          **
;**                                                **
;****************************************************
;****************************************************

; Set some variables associated with the user interface
(lset *old-pos* nil)
(lset *session* nil)
(send *gm-output* 'set-font 8)
(send *gm-output-little* 'set-font 8)
(lset *first* t)
(lset *temprect* (NewPtr 8))


; INTERFACE UTILITIES
; *****************
; Insures that a menu is active and selected before waiting for the
; learner to select an item from a menu.
(define-method (momentary-menu before choose-item) ()
  (send self 'refresh)
  (pad_$select_window &stream-id 1 nil))

; Puts the message X in Sherlock's big prompt window
(define (promptb x)
  (msg x t))

; Same as promptb, only adds a blank line before
; displaying the new message
(define (cpromptb x)
  (msg t x t))

; This function determines the type of each item on LIST and then
; uses the appropriate method to present the item in WINDOW
(define (output-msg list . window) ;
  (let((this-out
        (if window
          (car window)
          *gm-output*))
      (old-window *current-window*)
   )
  (set *current-window* nil)
  (send this-out 'select)
  (with-acquired-display
    (for-each (lambda (string)
     (cond
      ((eq? string 't)
       (send this-out 'new-line)
       (send this-out 'string-out "  ")
      (msg t))
      ((string? string)
       (send this-out 'string-out string)
      (msg string))
     ((eq? string 'c)
      (send this-out 'clear-screen)
      (send this-out 'string-out "  ")
     (msg t t t))
     ((atom? string)
       (if (string? (eval* string))
        (block
          (send this-out 'string-out (eval* string))
         (msg (eval* string))
        )
        (if (node? (eval* string))
        (block
          (send this-out 'string-out
           (symbol->string (nodes-name
             (eval* string))))
           (msg (symbol->string (nodes-name (eval* string))))
        )
        (block
         (send this-out 'string-out
             (symbol->string (eval* string)))
             (msg (symbol->string (eval* string)))))
       ))
      ((list? string)
       (msg string))
     ))
    list))
  (if old-window
    (send old-window 'select)
    nil)
 ))

(define (clear-little)
  (output-msg '(c) *gm-output-little*))

; Similar to OUTPUT-MSG, but works when the list contains
;  only strings
(define (output list) ;
  (let((output *gm-output*)
     (old-window *current-window*)
  )
  (send output 'select)
  (with-acquired-display
    (for-each (lambda (string)
     (cond
       ((eq? string 't)
        (send output 'new-line)
        (send output 'string-out " "))
       ((string? string)
        (send output 'new-line)
```

```
           (send output 'string-out " ")
            (send output 'string-out string))
          ((eq? string 'c)
           (send output 'clear-screen)
           (send output 'string-out " "))
          ))
        list))
   (send old-window 'select)
  ))


; Returns an identifying string for ICON which is appropriate for
; displaying in a prompt box
(define (icon->sstring icon)
   (let* ((string-list (icon-label1 icon))
        (string (string-append
              (caar string-list)
              (if (cadar string-list)
                 (string-append " "
                   (cadar string-list))
                 ""))))
    (substring string
         0
         (min (string-length string) ))))

; Call the graphics package menu function after displaying the
; optional PROMPT in the small prompt window
(define (run-menu* menu x y . prompt)
  (if prompt
     (output-msg (car prompt) *gm-output-little*)
    nil)
  (let ((choice (run-menu menu x y)))
  (hide-menu menu)
  choice))

; Sends the cursor to position POS.  Needed to send the cursor back to
; the point it was at before a menu was called.
(define (return-cursor pos)
  (dm-cmd (string-append
         "("
          (num->str (pt.h pos))
         ","
          (num->str (pt.v pos))
         ")"
        )))

; Puts the cursor in the middle of the screen
(define (go-middle )
  (dm-cmd "(413,377)"))

; Disable the normal mouse functions so that the graphics window
; is not accidentally modified while building a graphic map.
(define (disable-mouse)
 (dm-cmd "kd m1 ke")
 (dm-cmd "kd m2 ke")
 (dm-cmd "kd m3 ke")
)

; Enable the normal mouse functions. To be used when the user leaves
; the map building mode.
(define (enable-mouse)
 (dm-cmd "kd m1 wge ke")
 (dm-cmd "kd m2 wp ke")
 (dm-cmd "kd m3 wme ke")
)


; TOP LEVEL FUNCTIONS FOR DEALING WITH THE GRAPHIC INTERFACE
;    ****************************************************
; Main loop for interpreting mouse clicks in the main
; graphics window.
(define-method (ami-window pick-node-pb) ()
 (with-acquired-display
```

```
  (send self 'select)
  (chkerr (gpr_$set_cursor_active -1 nil))
  (let ((pos nil) (node nil) (menu? nil)(tmp-pos nil) )
   (send self 'set-event-mask 22)       ;stop only for mouse clicks
   (catch mouse
    (do () (())
     (chkerr (gpr_$set_cursor_active -1 nil))
     (receive (obs evt evd pos status)
             (gpr_$event_wait nil nil +position+ nil)
      (chkerr (gpr_$set_cursor_active 0 nil))
     (select evt
      ((gpr_$locator)
          (chkerr (gpr_$set_cursor_position pos nil)))
      ((gpr_$buttons)   ; 1
      (cond
       ((eq? evd 67) (mouse t))
     ;if the right button was pushed
         ;  exit the loop
       ((eq? evd *pick-button*)
     ;if the pick-button (left button)
         ; was pushed see where the cursor is.
        (block
         (set *old-pos* pos)
          (set tmp-pos (cons (pt.h pos)
                       (pt.v pos)))
         (set node   (icon-at-coords tmp-pos t))
        (cond
         ( node  (block
; If the cursor is on an icon set menu? to
; 'icon so that the icon menu
; will be used
                 (set menu? 'icon)
                 ))
         ( (set node (link-at-coords tmp-pos))
; If the cursor is on a link set menu? to
; 'remove so that that link
; will be removed
```

```
                (set menu? 'remove))
             ( t (block
; If the cursor is not on an icon or link,
; assume it is on the background
; set menu? to 'background so that the
; background menu will be used
                (set menu? 'background)
                (set node tmp-pos))
               ))
           (mouse t)                    ;exit the loop
          ))
          ((eq? evd *move-button*)
    ;if the middle button was pushed
                              ; set node to the icon the cursor is
                              ; on and set menu? to nil so that
                              ; moving icon function will be
                              ; called.
          (block
           (set node (icon-at-coords
                     (cons (pt.h pos)
                           (pt.v pos)) t))
           (set menu? nil)
           (and node (mouse t))))
;          exit the loop only if the cursor
; was on an icon
        ))
       (else nil)))))
    (chkerr (gpr_$set_cursor_active 0 nil))
   (list menu? node) )))
    ; return a list containing the variable
          ; menu? and the node clicked on


; This loop calls the loop PICK-NODE-PB to interpret mouse clicks in
; the graphic mapping window and then take the appropriate action.
(define (push-button)
 (disable-mouse)
```

```
   (set *current-window* nil)
   (output-msg '(c "Ready") *gm-output-little*)
   (send *gm-window* 'select)
   (send *gm-window* 'pop)
   (do ((done nil done))
    (done (block (enable-mouse)
               (set *current-window* nil)
               (cpromptb "Goodbye")
                  )t)
      (cpromptb "Click on the type of link you want... ")
     (promptb "           or"   )
      (promptb "Click on an icon you want to move")
    (go-middle)
  (if (not *first*)
    (flush-all-events)
   nil)
 (let* (
      (choice-whole (send *gm-window* 'pick-node-pb))
     (choice (cadr choice-whole))
     (menu? (car choice-whole))
    )
   (set *first* nil)
  (cond
    ((eq? menu? 'icon)
      (do-link-menu choice))
    ((eq? menu? 'background)
     (set done (eq? (run-menu 'background-menu
                    (car choice)
                    (cdr choice))
                'done)))
    ((eq? menu? 'remove)
      (destroy-wire choice))
    ((not menu?) (block
       (move-node-stable choice)))
 ))))

; This function is called once it has been
; determined that the user is starting to link icons.
; It prompts the learner and waits for the
; second icon (the TO icon) to be selected,
; then provides a menu for choosing a link type.
; Thus function calls EVAL-MOVE which starts
; the production system to decide what to do about the link.
(define (do-link-menu icon)
  (output-msg '(c "The link will come from this icon"
            t "Now, click on the icon you want it to go to")
          *gm-output-little*)
  (send *gm-window* 'select)
  (let* ((from (set-from icon))
       (to (set-to (cadr (send *gm-window* 'pick-node-pb))))
       (type (do-menu from to))
     )
  (msg t from " -- " type " -> " to t)
  (if (and from to type)
    (block
      (clear-little)
      (send *gm-window* 'select)
      (set a (link-gm-icons from to type) )
      (eval-move from to type)
    )
    (msg t "link cancelled" t))
  (send *gm-window* 'select)
  (msg t "about to unset icons" t)
  (unset from)
  (unset to)
  ))

; Sets global variables and starts
; the productions system to evaluate a link
(define (eval-move from to type)
  (msg t "Starting production system." t)
  (init-move-info)
  (set *goal* (list (list 'feedback)))
  (if
```

```
 (and
   (set *link-made* (wire->link type))
   (set *from-icon* (icon-node from))
   (set *to-icon* (icon-node to))
    (set *from-icon-english* (icon-english *from-icon*))
    (set *to-icon-english* (icon-english *to-icon*))
   (set *from-icon-children* (icon-children *from-icon*))
   (set *to-icon-children* (icon-children *to-icon*))
  )
 (do-prods)
 nil))

(define (get-new-move)   ; ends productions system
  (set *goal* (list (list nil)))
  )


; This funtion determines the most appropriate location for
; the link menu (using the locations of FROM and TO), then
; runs the link menu.
(define (do-menu from to)
  (let ((coords-f (icon-coords from))
        (coords-t (icon-coords to))
        (mid-x (quotient (send *gm-window* 'max-x) 2))
        (mid-y (quotient (send *gm-window* 'max-y) 2))
        (y-off (send *gm-window* 'y-offset))
        (x-off (send *gm-window* 'x-offset))
       (left-x 0)
       (right-x 0)
       (horiz-diff 0)
       (top-y 0)
       (bottom-y 0)
      (vert-diff 0)
      (x-menu 0)
      (y-menu 0)
       (top-icon nil)
       (bottom-icon nil)
      (right-icon nil)
      (left-icon nil)
    )
  (if (< (car coords-f)(car coords-t))
    (block
      (set left-x (caddr coords-f))
      (set right-x (car coords-t))
      (set left-icon coords-f)
      (set right-icon coords-t)
    )
    (block
      (set left-x (caddr coords-t))
      (set right-x (car coords-f))
      (set left-icon coords-t)
      (set right-icon coords-f)
    )
  )
  (set horiz-diff (fx- right-x left-x))
  (if (< (cadr coords-f)(cadr coords-t))
    (block
      (set top-y (cadddr coords-f))
      (set bottom-y (cadr coords-t))
      (set top-icon coords-f)
      (set bottom-icon coords-t)
    )
    (block
      (set top-y (cadddr coords-t))
      (set bottom-y (cadr coords-f))
      (set top-icon coords-t)
      (set bottom-icon coords-f)
    )
  )
  (set vert-diff (fx- bottom-y top-y))
  (set x-menu  (fx- (fx+ left-x (quotient horiz-diff 2)) 52))
  (set y-menu  (fx- (fx+ top-y (quotient vert-diff 2)) 100))
  (if (and (< horiz-diff 100)(< vert-diff 200))
    (if  (< horiz-diff vert-diff)
```

```
          (set x-menu
           (if (< x-menu mid-x)
              (caddr right-icon)
              (fx- (car left-icon) 105)))
          (set y-menu
           (if (< y-menu mid-y)
              (cadddr bottom-icon)
              (fx+ 200 (cadr top-icon))
              ))))
       (set x-menu (fx+ x-menu x-off))
       (set y-menu (fx+ y-menu y-off))
       (msg t "x-menu " x-menu t "y-menu " y-menu t)
    (if (eq? from to)
;  the same icon was clicked twice so assume that the user
;  wants to do something with that icon rather than make
;  a link.  Call the node utility menu.
      (block
       (set *tmp3* from)
       (run-menu* 'node-utils x-menu y-menu
         '(c "What would you like to do with this icon?")
       )
      nil)
;  Otherwise go ahead and run the link menu
      (block
        (set *f-icon-string* (icon->sstring from))
        (set *t-icon-string* (icon->sstring to))
       (run-menu* 'test-link x-menu y-menu
         '(c "Pick a link "
           "From (gray):    "  *f-icon-string* t
           "                 To (black):    " *t-icon-string*)))
    )
  ))

; offer to remove last link
(define (offer-change)
        (output-msg '(c "Would you like to remove that link?")
                 *gm-output-little*)
    (if (run-menu 'yes-no-menu)
      (block
        (hide-menu 'yes-no-menu)
        (send *gm-window* 'select)
         (destroy-wire *current-link-icon*)
      )
      (hide-menu 'yes-no-menu)
    ))


(define (offer-suggestion prod)
  (msg t "I am confused." t)
  (if (ask-user "Would you like to know what I would have done here? ")
    (and
      (msg t "I would have used a " )
       (msg (production-link (get-production prod)))
      (msg " link here." t))
    nil))

; This function tells the learner that
; he is over generalizing the
; concept *F* as belonging to all *T* when it should be just all
; TMP. The variable TMP must be set before
; this function is called.
(define (show-over-gen)
  (and
   (output-msg '(c "You seem to be over generalizing.  Not all"
       t *t* " have " *f* ".  Do you want to know"
       t "what type of " *t* " has it?" t))
   (run-menu 'yes-no-menu)
   (hide-menu 'yes-no-menu)
   (output-msg '(t c "All " tmp " have " *f* t)))
 )
```

```
;**************************************************
;**************************************************
;**                                              **
;**                    MENUS                     **
;**                                              **
;**************************************************
;**************************************************

; Menus are defined using DEFINE-MENU, a function from
; the graphics package.  Each item on the menu has the
; following form:
;         NAME          for internal use only
;         LABEL   text which appears in the menu for that item
;         DOC           text which appears in the system prompt window
;                       while the cursor is on that item
;         BODY          action to be taken if that item is selected
;         TEST          (optional) this item appears on the menu only
;                       if the argument for TEST evaluates as true.


; Creates menu which allows the user to select a link type
(define-menu test-link
  :type  momentary
  :x 880
  :y 0
  :label "Choose a link"
  :font   "~code/t3/graphics/fonts/chicago.12"
 :doc   "lets the user choose which link is desired"
 :items ((:name  part
        :label "Part"
        :doc   "Will make a part link"
         :body  ((set *selection* 'parts)) )
      (:name  equiv
       :label "Equiv"
       :doc   "Will make an equiv link"
        :body  ((set *selection* 'equivs)  ))
      (:name  is-a
      :label "Is-a"
      :doc   "Will make an is-a link"
       :body  ((set *selection* 'parents) ))
     (:name  leads
      :label "Leads"
      :doc   "Will make a leads to link"
       :body  ((set *selection* 'results) ))
     (:name  prop
      :label "Prop"
      :doc   "Will make a property link"
       :body  ((set *selection* 'prop) ))
     (:name  not
      :label "Not"
      :doc   "Will make a not link"
       :body  ((set *selection* 'splits) ))
     (:name  dummy1
      :label "----------"
      :test  inactive)
     (:name  no-link
      :label "Cancel"
      :doc   "Cancel the link"
      :body  ((set *selection* nil) ))
 ))

; Menu which allows the user to select actions to
; be taken on an individual icon.
( define-menu node-utils
  :type  momentary
  :x 880
  :y 0
  :label "Choose an action"
   :font  "~code/t3/graphics/fonts/chicago.12"
  :doc   "Choose an action"
  :items ((:name  clear
       :label "Clear"
       :doc   "Will clear this icon"
```

```
       :body  ((unset *tmp3*)) )
      (:name  clone
       :label "copy"
       :doc   "Make an instance of this icon"
       :body  ((make-icon-copy *tmp3*) ))
      (:name  children
       :label "list children"
      :doc   "output icon children"
       :body  ((msg t (icon-children (icon-node *tmp3*)) t)
            (run-menu 'yes-no-menu) ))
      (:name  no-link
       :label "Cancel"
      :doc   "Cancel the link"
       :body  ((set *selection* nil) ))
  ))

; Background menu that allows the user to take actions that
; affect the whole system.
(define-menu background-menu
  :type  popup
  :x 880
  :y 0
  :label "Choose an action"
   :font  "~code/t3/graphics/fonts/chicago.12"
  :doc   "Offers various utilities"
  :items ((:name  quit
        :label "Quit"
        :doc   "Exit to T"
         :body  ((set *selection* 'done)) )
       (:name  end-session
        :label "unLock"
        :doc   "allow drawing new icons, etc"
        :test (*lock*)
        :body  ((set *lock* nil) ))
       (:name  begin-session
       :label "Lock"
```

```
    :doc   "allow drawing new icons, etc"
     :test ((not *lock*))
    :body  ((set *lock* t) ))
   (:name  redo-menus
    :label "Refresh menus"
    :doc   "Fix the menus"
    :body  ((refresh-menus)))
   (:name  draw-law
    :label "Draw law icons"
    :doc   "Draw all of the icons"
     :test ((not *lock*))
    :body  ((hide-menu 'background-menu)
           (draw-all 'law-icon) ))
   (:name  draw-animals
    :label "Draw animal icons"
    :doc   "Draw all of the icons"
     :test ((not *lock*))
    :body  ((hide-menu 'background-menu)
             (draw-all 'animal-icon) ))
   (:name  never-mind
    :label "never mind"
    :doc   "Abort this menu"
    :body  ((msg t "Ignoring menu") ))
   (:name  see-steps
    :label "see production steps"
    :doc   "toggle visible production steps"
   :body  ((if *fb-prod*
            (set *fb-prod* nil)
            (set *fb-prod* t)) ))
 ))

; The following 2 menus are used to allow the user to answer
; a question presented in the prompt window
(define-menu yes-no-menu
  :type  popup
  :x 520
  :y 0
```

```
   :label "Choose one"
    :font   "~code/t3/graphics/fonts/palatino.24"
   :doc   "Offers various utilities"
   :items ((:name  yes
          :label "YES"
           :doc   "answers yes to the question"
           :body  ((set *selection* 't) ) )
          (:name  no
           :label "NO"
           :doc   "answers no to the question"
           :body  ((set *session* nil)  ))
   ))


(define-menu ok-menu
  :type  momentary
  :x 520
  :y 0
  :label "Click to continue"
   :font   "~code/t3/graphics/fonts/palatino.24"
  :doc   "to continue"
  :items ((:name  ok
         :label "OK"
         :doc   "continues"
          :body  ((set *selection* 't)) )
  ))


; This function refreshes all of the menus,
; or creates them if that
; is necessary. It can be called from
; the background menu if for
; some reason the menus have become garbled.
(define (refresh-menus)
 (let (  (back (menu-window (find-menu 'background-menu)))
       (yes  (menu-window (find-menu 'yes-no-menu)))
       (ok   (menu-window (find-menu 'ok-menu)))
    (link (menu-window (find-menu 'test-link)))
    (utils(menu-window (find-menu 'node-utils)))
  )
  (if back
    (send back 'refresh)
    (create-menu 'background-menu))
  (hide-menu 'background-menu)
  (if yes
    (send yes 'refresh)
    (create-menu 'yes-no-menu))
  (hide-menu 'yes-no-menu)
  (if ok
    (send ok 'refresh)
    (create-menu 'ok-menu))
  (hide-menu 'yes-no-menu)
  (if utils
    (send utils 'refresh)
    (create-menu 'node-utils))
  (hide-menu 'node-utils)
  (if link
    (send link 'refresh)
    (create-menu 'test-link))
  (hide-menu 'test-link)))


;*****************************************************
;*****************************************************
;**                                                 **
;**   THESE FUNCTIONS MANIPULATE THE GRAPHIC MAP    **
;**                                                 **
;*****************************************************
;*****************************************************


(herald gm3
(read-table (*value user-env 'ami-read-table)) )

; SET GRAPHIC MAP VARIABLES
```

```
;   ***********************
(lset *new-num* 0)
(lset *draw-role-types?* t)
; Create prompt windows
(set netl2  (init-ami-window "netl2" 0 0 520 300))
(set netl3  (init-ami-window "netl3" 0 0 520 100))

(set *gm-window* netl)
(set *gm-output* netl2)
(set *gm-output-little* netl3)
; SET LOCATION FOR FIRST ICON
(set *next-gm-icon* '(10 . 10))

; GRAPHIC MAP UTILITIES
; ********************
;create a unique small number
(define (new-num)
 (set *new-num* (1+ *new-num*))
 *new-num*)

; Clears the main graphic mapping window
(define (clear-gm)  ;
 (send *gm-window* 'select)
 (clear)
 (set (car *next-gm-icon*)10)
 (set (cdr *next-gm-icon*)50)
)


; Removes all items from the lists that are used to maintain the
; status of the main graphic mapping window
(define (clear)
 (walk (lambda (i) (-arg node-icon (icon-node i) i))
      (send *current-window* 'visible-icons))
 (walk (lambda (li) (let ((l (l-icon-nodes li)))
                 (if (node? l) (-arg node-icons l li))))
      (send *current-window* 'visible-l-icons))
 (send *current-window* 'clear-screen)
 (send *current-window* 'set-visible-icons nil)
 (send *current-window* 'set-visible-head-icons nil)
 (send *current-window* 'set-visible-l-icons nil)
 (send *current-window* 'set-visible-wires nil))

; Fills an icon with COLOR to indicate that it has been selected
(define (fill-icon icon color)
  (apply SetRect *temprect* (icon-coords icon))
  (InsetRect *temprect* 1 1)
  (FillRect *temprect* color))

; indicates that an icon has been selected
; as the FROM end of a link
(define (set-from icon)
  (fill-icon icon gray)
  (display-gm-text icon)
 icon)

; indicates that an icon has been
; selected as the TO end of a link
(define (set-to icon)
  (fill-icon icon black)
  (display-gm-text icon)
 icon)

; removes any color from an icon
(define (unset icon)
  (fill-icon icon white)
  (display-gm-text icon)
 (send *gm-window* 'set-fill-value   1)
 icon)

; Removes the color from all icons currently on the screen
(define (reset-all-icons)
 (send *gm-window* 'select)
 (walk (lambda (node)(unset (node-icon (car node))))
```

```
   (node-children* %icon)))

; SET LOCATION FOR NEXT ICON BASED ON BOTTOM OF ICON JUST
DRAWN
(define (adjust-next-gm bottom)
 (if (> (+ bottom 90 ) (send *gm-window* 'window-max-y))
   (block
     (set (car *next-gm-icon*) (fx+ (car *next-gm-icon*) 100))
    (set (cdr *next-gm-icon*) 10)
   )
   (set (cdr *next-gm-icon*) (fx+ bottom 5))
 ))



; MANIPULATING ICONS
;   *****************

; CREATE ICON CORRESPONDING TO node
(define (gm-icon node)
 (let ((i (make-icon)))
  (set (icon-node i) node)
  (set (icon-name i) (concatenate-symbol node '-icon))
  (set (icon-label1 i) (list (node-english node)))
  (set (icon-label2 i) (list ""))
  (set (icon-window i) *gm-window*)
  (set (icon-l-style i) nil)
  (set (node-icon node) i)
  (send *current-window* 'set-visible-icons
     (cons i (send *current-window* 'visible-icons)))
  (if t
     (send *current-window* 'set-visible-head-icons
         (cons i (send *current-window* 'visible-head-icons))))
i))
```

```
; Creates and displays icons for all
; nodes that are descendents of
; the node TYPE.  TYPE is assumed to be
; the generalization of a set of
; nodes representing icons that are
; associate with a particular text,
; ie; LAW-ICONS or BIRD-ICONS.
(define (draw-all type) ;
 (let ((nodes (node-children* (get-node type)))
    )
  (send *gm-window* 'select)
  (if (node-icon (caar nodes))
    (block
      (set *no-icons* t)
      (set *lock* nil)
     (clear-gm))
    (block
      (set *no-icons* nil)
      (set *lock* t)
      (walk (lambda (node)(draw-next-icon (car node)))
         nodes)
  )
 )))


; Creates an icon for NODE, then displays it at the
; next appropriate location.
(define (draw-next-icon node)  ;
 (let* ((icon nil)
      (x1 (car *next-gm-icon*))
      (y1 (cdr *next-gm-icon*))
     (x2 (fx+ x1  120))
     (y2 (fx+ y1  70))
     (rem-i-coords
           (map icon-coords
              (send *current-window*
                'visible-head-icons)))
    )
   (if (any-overlap? (list x1 y1 x2 y2 )   rem-i-coords)
    (block
     (adjust-next-gm y2)
     (draw-next-icon node)
    )
    (adjust-next-gm
       (display-gm-icon (set icon (gm-icon node)) x1 y1)
    )
  )
  icon
   ))

; serves same function as DRAW-NEXT-ICON except this function
; provides the option of specifying the location of the new icon
(define (draw-gm-icon  node . coords)
  (let* ((icon (gm-icon node))
      (x1 (if coords
           (caar coords)
            (car *next-gm-icon*)
        ))
      (y1 (if coords
           (cdar coords)
            (cdr *next-gm-icon*)
        ))
       (y2  (display-gm-icon icon x1 y1))
     )
    (if (not coords)
      (adjust-next-gm y2)
      nil)
 icon
 ))
```

```
; Display ICON, which already exists, in the graphic mapping window
(define (display-gm-icon icon x1 y1)  ;
  (let* ((ymove (fx+ *y-gap* *char-height*))
        (xmove *x-gap*)
        (x2 (fx+ x1 xmove))
        (y2 y1)
        (maxx 0)
       )
   (with-acquired-display
     (main-font)
    (map
     (lambda (string)
       (set maxx (max (label-length string) maxx))
      (send *current-window* 'move
            x2
          (set y2 (fx+ y2 ymove))  )
       (write-string *current-window* string)
     ) (car(icon-labell icon)))         )
  (set y2 (fx+ y2 3))
  (set x2 (fx+ (* 2 xmove) (fx+ x1 maxx)))
  (send *current-window* 'draw-rectangle x1 y1 x2 y2)
   (send *current-window* 'set-raster-op 6)
   (send *current-window* 'set-raster-op 3)
  (set (icon-coords icon) (list x1 y1 x2 y2))
  y2
  ))

; Creates a copy of an icon.  First creates
; a new node in the network
; with the same s-links that the node
; corresponding to ICON orignally
; had. Then creates and draws the an
; icon on the screen corresponding
; to that node.
(define (make-icon-copy icon)
  (let ((node (make-node-copy (icon-node icon)))
      (current-window *current-window*)
```

```
        )
  (send *gm-window* 'select)
  (draw-gm-icon node)
  (send current-window 'select)
  ))


; Creates a new node in the network that has the same
; s-links as NODE originally had.
(define (make-node-copy node)
  (let* ( (node (get-node node))
        (new-name  (string->symbol
                (string-append
                  (symbol->string (node-name node))
                 (num->str (new-num)))))
      (new-node (indiv! new-name  'icon))
     )
    (gm-label: new-node  (node-english node))
   ((get-concepts node )
    (map!  'concept-node  new-node) )
   new-node
  ))


; Sets the node-english slot of NODE to TEXT
(define (gm-label: node text)
   (set (node-english  node) text))


; Removes ICON from all of the lists
; that are used to maintain the
; status of the main graphic mapping window
(define (kill-icon icon)
    (send *current-window* 'set-visible-icons
       (delq icon (send *current-window* 'visible-icons)))
    (send *current-window* 'set-visible-head-icons
          (delq icon (send *current-window* 'visible-head-icons)))
    (set (node-icon (icon-node icon)) nil))


; Display just the text for ICON in the graphic mapping screen
(define (display-gm-text icon)    ;
  (let* ((coords (icon-coords icon))
       (x1 (car coords))
       (y1 (cadr coords))
        (ymove (fx+ *y-gap* *char-height*))
       (xmove *x-gap*)
       (x2 (fx+ x1 xmove))
       (y2 y1)
       (maxx '())
      )
  (with-acquired-display
    (main-font)
    (map
     (lambda (string)
       (set maxx (cons (label-length string) maxx))
      (send *current-window* 'move
             x2
          (set y2 (fx+ y2 ymove)) )
       (write-string *current-window* string)
    ) (car(icon-label1 icon)))          )
 ))


; Loop that erases a rectangle of dimensions WIDTH by HEIGHT
; at position OLD-POS and redraws the same
; rectangle at the current
; location of the mouse.
; Used when the user is specifiying a new location
; for an icon, to show where that icon will be
; placed when the middle
; button is pressed again.
(define-method (graphics-mixin position-rect-stable) (old-pos width height)
 (format (repl-output)
```

```
     "Position rectangle inside window: ~A~%and click middle button~%"
    (send self 'label))
(force-output (repl-output))
(with-acquired-display
 (send self 'select)
 (msg t "****** old-pos is " old-pos)
 (send self 'set-raster-op 6)
 (let ((left 0) (top 0) (x 0) (y 0) (done? nil)
     (pos old-pos)
 )
  (chkerr (gpr_$draw_box (set left (car pos))
                    (set top (cdr pos))
                 (+ left width) (+ top height) nil))
  ; Now track the cursor with a moving box
  (send self 'set-event-mask 22)
      ; buttons + locator + left-window
 (do () (done? t)
    (receive (obs evt evd pos sts)
          (gpr_$event_wait nil nil +position+ nil)
     (ignore obs)
      (chkerr sts "gpr_$event_wait")
      (set x (Pt.h pos)) (set y (Pt.v pos))
     (select evt
      ((gpr_$locator)
       ;Erase the previous box
       (chkerr (gpr_$draw_box left top
            (+ left width) (+ top height) nil))
      ;Draw the new box
       (chkerr (gpr_$draw_box (set left x) (set top y)
           (+ left width) (+ top height) nil)))
      ((gpr_$buttons)
      (and (eq? evd 66) (set done? t)))
     ((gpr_$left_window)
      ;Erase the current box
      (chkerr (gpr_$draw_box left top
          (+ left width) (+ top height) nil))
      (set pos (send self 'wait-until-entered))
```

```
        (chkerr (gpr_$draw_box
                (set left (car pos)) (set top (cdr pos))
          (+ left width) (+ top height) nil))
        (send self 'set-event-mask 22)))))
    (chkerr (gpr_$draw_box x y (+ x width) (+ y height) nil))
    (send self 'set-raster-op 3)
    (send self 'set-event-mask 0)
    (list x y))))


; GM-LINKS
;  ********
;  Each gm-link consists of:
;        graphic  components:
;                 line between two icons
;                 possible head at each end of line
;        internal  components:
;                 wire in each direction of gm-link
;


;  Creates the components of a gm-link between
;  2 icons.
(define (link-gm-icons from to type) ;
 (let ((coords-from (icon-coords from))
     (coords-to   (icon-coords to))
      (type-list (list type (icon-node from)(icon-node to)))
    )
   (join-icons from (pop coords-from)(pop coords-from)
            (pop coords-from)(pop coords-from)
         to   (pop coords-to) (pop coords-to)
            (pop coords-to) (pop coords-to)
         type-list)
  (push (icon-head-outside-wires from) type-list)
  (push (icon-head-outside-wires to)
      (rev-wire-triple type-list))
))
```

```
;  Returns the reciprocal type of any wire.
;  Used to create the proper
;  identification for wire representing the
;   other end of a gm-link
(define (get-opposite-wtype wt)  ;
 (case wt
    ((equivs) 'equivs)
     ((splits) 'splits)
    ((map-parents) 'maps)
    ((children) 'parents)
    ((parents) 'children)
    ((exins) 'rev-exins)
    ((rev-exins) 'exins)
    ((scopes) 'rev-scopes)
    ((rev-scopes) 'scopes)
   ((results) 'causes)
   ((causes) 'results)
   ((parts) 'wholes)
   ((prop) 'concept)
   ((concept) 'prop)
   ((wholes) 'parts)
   (else nil)))


;  Removes internal components of a gm-link
(define (kill-wire w-list)
  (del-link (cadr w-list) (car w-list) (caddr w-list))
  (del-link (caddr w-list) (get-opposite-wtype (car w-list))
        (cadr w-list)))

;  Removes internal components of gm-link
; LK of type LK-TYPE that is a
; part of node N.
(define (del-link n lk-type lk)
  (alist-del! lk-type lk (node-misc-links n)))
```

```
; Removes the graphic component of gm-links on the list LINKS.
(define (erase-links links)
  (send *current-window* 'set-raster-op 6)
 (msg t "links " links t)
  (walk
   (lambda (link-struc)
    (let*
      ((link (car link-struc))
       (icons (wire-icons link)))
     (draw-wire link)
      (link-gm-icons (pop(icons))
              (pop (icons))
              nil)
   )) links)
  (send *current-window* 'set-raster-op 3))


; Allows the user to select a gm-link with the
; mouse. That link is then removed.
(define (seek-and-destroy-link)
    (destroy-wire (send *gm-window* 'pick-link)))


; Finds the gm-link whose label is at the
; location COORDS. This is used
; to determine if the user has just clicked
;  on a link label, and if so,
; which link is chosen.
(define (link-at-coords coords)
   (first (lambda (i)
            (point-in-rect? (car coords) (cdr coords)
                     (l-icon-coords i)))
        (send *current-window*  'visible-l-icons)))
```

```
;***************************************************
;***************************************************
;**                                               **
;**            FUNCTIONS THAT RUN THE             **
;**            PRODUCTION SYSTEM                  **
;**                                               **
;***************************************************
;***************************************************

; UTILITIES FOR HANDLING THE GOAL STACK
;    *********************************
(set *goal* nil)

; Determines if the current goal is G
(define (eq-goal? g)
 (eq? (goal) g))

; Returns the current goal
(define (goal)
 (caar *goal*))

; Returns the list of productions that have
; already been used for the current goal
(define (goal-state)
 (cdar *goal*))

; Add production P to the list of productions that
; have already been used for the current goal
(define (goal-state! p)
 (let ((old-car (pop *goal*)))
  (push *goal* (append old-car (list p)))))

; Put goal G on top of the goal stack
(define (goal! g)
 (push *goal*  (list g)))
```

```
; Check to see if production PROD has already
; been used for the current goal.
(define (have-done? prod)
  (memq (get-production prod) (goal-state)))


;*******


; Check to see if production PROD is a possible
; production for the current cycle.
(define (check-conditions-state prod)
 (and
; It has not been used for the current cycle
  (not (memq prod (goal-state)))
; all of its antecedents are true
  (eval* (cons 'and (production-conditions (get-production prod))))))

; This function runs one cycle of the
; production system
(define (fire-prods)
 (let ((best-prod (car (sort-prods
                    (all check-conditions-state *tutoring-strats*))))
    )
     (goal-state! best-prod)
   (msg t "PRODUCTION:  " best-prod t)
  (and *fb-prod*
    (if (ask-user "Continue?")
      nil
      (break)
   ))
  (if *explain* (eval** (production-explain best-prod)))
 (for-each (lambda(action)(eval* action))
     (production-actions best-prod))
   ))
```

```
; This is the main loop that runs the production system
; cycles by repeatedly calling FIRE-PRODS.
(define (do-prods)
 (do ((cycle 1 (1+ cycle)))
    ((eq-goal? nil) t)
    (msg t "CYCLE            " cycle "  GOALS:  " *goal*)
    (fire-prods)))


; Clears all variables before evaluating a new move
(define (init-move-info)
 (set *plan-used* nil)
 (set tmp nil)
 (set *possible-plans* nil)
 (set *plan-used-status* nil)
 (set *link-used* nil)
 (set *from-icon* nil)
 (set *to-icon* nil)
 (set *from-icon-children* nil)
 (set *to-icon-children* nil)
 (set *from-icon-english* nil)
 (set *to-icon-english* nil)
 (set *possible-prods* nil)
 (set *sherlocks-move* nil)
 (set *why-not-sherlocks-move* nil)
 (set *why-wrong-learners-plan* nil)
 (set *other-good-moves* nil)
 (set *buggy-moves* nil)
 (set *reasons* nil)
 (init-fact-info)
 (set *from-interps* nil)
 (set *to-interps* nil)
 (set *interp* nil)
 (set *new-interp* nil)
 (set *asked-facts* nil)
 (set *possible-reasons* nil)
)
```

```
(define (init-fact-info)
  (set *correct-facts* nil)
  (set *wrong-facts* nil)
  (set *correct-neg-facts* nil)
  (set *missing-facts* nil)
            )


; FUNCTIONS FOR DEALING WITH ALTERNATE
; INTERPRETATION OF ICONS
;    ********************************

; An interpretation is a list with the following items:
;  accuracy of the interpretation
;  the interpretation of the FROM icon
;  the interpretation of the TO icon
;  the move Sherlock would have made
;
; The accuracy of the interpretation is equal to:
;
;             number of facts Sherlock and learner agree on
;
;          ______________________________________________________
;          number of facts Sherlock and learner do not agree on
;


; Constructs the interpretation list using RATIO for the
; accuracy and the global variables for the other components
(define (make-interp ratio)
  (list ratio
      *f*
      *t*
       *sherlocks-move*
 ))
```

```
; Returns the accuracy ration of INTER
(define (accuracy interp)
  (if interp
  (car interp)
   -10))

; Computes the accuracy ration for the current interpretation
(define (make-accuracy-ratio)
   (quotient (+ (length *correct-facts*)
            (length *correct-neg-facts*))
         (+ (length *wrong-facts* )
            (length *missing-facts*))))


; This function compares the accuracy
; of the current inerpretation
; (*new-interp*) to the accuracy of the most accurate
; interpretation so far (*interp*).
; If the current interpretation
; is higher, it is made *interp*.
(define   (store-interp)
   (msg t "storing interp. old was:"
          (accuracy *interp*) "new is:" (accuracy *new-interp*))
   (if (> (accuracy *new-interp*) (accuracy *interp*))
      (block (set *interp* *new-interp*)
                  (msg t "putting the new interp into *interp*"))
     (block *interp*
          (msg t "keeping the old interp")))
                  (set *new-interp* nil)
 )

; Sets the global variables to the values of INTERP.  Used to
; make INTERP the current interpretation.
(define (use-interp interp)
 (set *f* (cadr interp))
 (set *t* (caddr interp))
 (set *sherlocks-move* (cadddr interp) )
 (set *why-wrong-learners-plan*
       (if *plan-used* (why-wrong *plan-used*)
                nil))
 (set *why-not-sherlocks-move*
        (if *sherlocks-move* (why-not *sherlocks-move*)
                nil))
 (set *interp* interp)
 )


;**************************************************
;**************************************************
;**                                              **
;**       DEFINITIONS TO SET UP SEMANTIC NETWORK **
;**  REPRESENTATION OF THE CONSIDERATION TEXT    **
;**                                              **
;**************************************************
;**************************************************
(herald consideration)

; The functions that create the network are defined in
; Gasser's code (node.t). They arguments are as follows:

;          (type! name parent . weights)
; Creates a type node with PARENT parent. Weights is an optional
; keyed list: 'up specifies the weight from the created node
; to its parent (default .9), 'down specifies the weight from
; the parent to the new node (default .1).

;          (indiv! name parent . options)
; Creates an individual node with PARENT parent.
; The weight to the
; parent is always 1.0. The weight from
; the parent to the individual
; is optionally specified; otherwise it is .1.

;          (type-role! name parent owner . weights)
; Creates a type role node with PARENT parent and owner OWNER.
; WEIGHTS is an options keyed list of weights. Possible keys
; are
;    'up for the weight from the role to its parent
;          (default .9)
;    'down for the weight from the parent to the role
;          (default .1)
;    'r for the weight on the "has-a" link from OWNER
;          to the node (default .2)
;    'o for the weight on the "has-a" link from the node
;          to OWNER (default .8)


;          (indiv-role! name parent owner . weights)
; Creates an individual role node with parent PARENT and owner
; OWNER. For WEIGHTS, see last function.


;          (map! parent owner . options)
; Creates a map node with map parent PARENT
; and owner OWNER. OPTIONS
; is a keyed list including weights, as for TYPE-ROLE!
; and INDIV-ROLE!
; (except that the "up-weight" is always 1.0)
; and also an optional name
; (with key 'name). If no name is specified,
;  a name is created from
; the names of OWNER and PARENT.

;          (inhib-link! name par)
; Creates an "inhibition link". The excite component of the link
; is a list of the nodes inhibiting
; each other. PARENT is a parent
; of the inhibited nodes. It is only given to provide a namr for
; the link when NAME is nil.
;          (+inhib-link! nd i-link)
; Adds NODE to the list of nodes inhibited at INHIB-LINK.


;          (is-a! ch par . weights)
; Creates an "is-a" link from CHILD to
; PARENT, actually two excite
; links.  WEIGHTS is an optional keyed
; list specifying the "up-weight"
; (default .9) and/or the "down-weight" (default .1).


;          (inside-is-a! ch par . weights)
; Creates an "inside is-a" link from CHILD to PARENT.  WEIGHTS is
; as for IS-A.


;          (distinct! n1 n2 . parent)
; Causes NODE1 and NODE2 to inhibit each other.
; Creates an "inhibition link" and puts
; NODE1 and NODE2 on the list
; of inhibited nodes.


;*************************************************
;   BASICS
;*************************************************

(type! 'animal nil)
(type! 'human 'animal 'UP .2)

(type! 'time nil)
(type! 'past 'time 'UP .1)
(type! 'present 'time 'UP .1)
(type! 'future 'time 'UP .1)
```

```
(inhib-link! 'timeness 'time)
(+inhib-link! 'past 'timeness)
(+inhib-link! 'present 'timeness)
(+inhib-link! 'future 'timeness)
(type! 'fact nil)


(type! 'part nil)
(type! 'arg nil)
(type! 'prop nil)

(type! 'thing nil)
(type-role! 'expectation 'arg 'thing 'up .1)
(is-a! 'expectation 'fact)

(type! 'act nil)
(type-role! 'condition 'arg 'act 'UP .1)
(is-a! 'condition 'fact 'UP .1)
(indiv-role! 'actor 'arg 'act 'UP .1)
(is-a! (indiv-role! 'start 'arg 'act) 'time 'UP .1)
(is-a! (indiv-role! 'finish 'arg 'act) 'time 'UP .1)
(is-a! 'actor 'human 'UP .1)
(type-role! 'result 'arg 'act 'up .1)
(is-a! 'result 'fact 'up .1)

(type! 'illegact 'act 'UP .1)

(type! 'legact 'act 'UP .1)

(inhib-link! 'legality 'act)
(+inhib-link! 'legact 'legality)
(+inhib-link! 'illegact 'legality)

(type! 'endangact 'illegact)

(type! 'prohibact 'illegact)
```

```
(type! 'hurtingact 'illegact)

(type! 'optact 'act 'UP .1)

(type! 'recqact 'act 'UP .1)

(inhib-link! 'optionality 'act)
(+inhib-link! 'optact 'optionality)
(+inhib-link! 'recqact 'optionality)

;**********************************************
;  RELATIONSHIPS
;**********************************************


(type! 'party 'human 'UP .1)

(type! 'relationship 'thing)
(indiv-role! 'party1 'arg 'relationship 'UP .1)
(is-a! 'party1 'party)
(indiv-role! 'party2 'arg 'relationship 'UP .1)
(is-a! 'party2 'party)


(distinct!  'party1 'party2)

(type! 'mtrans 'act 'UP .1)
(indiv-role! 'mtcontent 'arg 'mtrans 'UP .1)
(is-a! 'mtcontent 'fact 'UP .1)
(indiv-role! 'mthearer 'arg 'mtrans 'UP .1)
(is-a! 'mthearer 'human 'UP .1)

(type! 'promise 'mtrans )
(map! 'mtcontent 'promise)
(is-a! (indiv-role! 'pr-act 'part 'promise::mtcontent 'UP .1)
     'act 'UP .1)
(is-a! (type-role! 'pr-condition 'part 'promise::mtcontent 'UP)
```

```
     'fact 'UP .1)
(map! 'condition 'pr-act)
(inside-is-a! 'pr-condition 'pr-act::condition)

(is-a! (map! 'start 'pr-act) 'future)

(type! 'forbearance 'optact )
(is-a! (indiv-role! 'fb-act 'arg 'forbearance 'UP .1)
     'legact )

;**********************************************
;  CONSIDERATIONS
;**********************************************

(type! 'consideration 'legact 'UP .1)

(type! 'cons-act 'consideration)
(is-a! 'cons-act 'optact)


(type! 'cons-prom 'consideration)
(is-a! 'cons-prom 'promise)
(map! 'mtcontent 'cons-prom)
(is-a! (map! 'pr-act 'cons-prom::mtcontent) 'optact)

(is-a! (type! 'cons-forb 'consideration) 'forbearance)
(is-a! (map! 'fb-act 'cons-forb) 'legact)

(inhib-link! 'considerations 'consideration)
(+inhib-link! 'cons-forb 'considerations)
(+inhib-link! 'cons-prom 'considerations)
(+inhib-link! 'cons-act 'considerations)

;**********************************************
;  AGREEMENTS & CONTRACTS
;**********************************************
(type! 'exist-moral-obligation 'fact)
```

```
(type! 'exist-legal-obligation 'fact)

(type! 'agreement 'relationship)
(is-a! (indiv-role! 'mtrans1 'part 'agreement 'UP .1) 'mtrans)
(is-a! (indiv-role! 'mtrans2 'part 'agreement 'UP .1) 'mtrans)

(is-a! (map! 'expectation 'moral-obligation) 'exist-moral-obligation)

(type! 'contract 'agreement)
(map! 'mtrans1 'contract)


(is-a! (indiv-role! 'cons1 'part 'contract 'up .1) 'consideration)
(is-a! 'cons1 'promise 'down .2)
(map! 'pr-condition 'cons1)
(map! 'pr-condition (map! 'pr-act 'cons1))
(inside-is-a! 'cons1::pr-condition 'cons1::pr-act::pr-condition)
(map! 'result 'cons1)
(indiv-role! 'cons1-done 'arg 'cons1::result 'up .1)
(map! 'result 'cons1::pr-act)
(indiv-role! 'cons1-pr-act-done 'arg 'cons1::pr-act::result)

(IS-A! (INDIV-ROLE! 'CONS2 'PART 'CONTRACT 'UP .1) 'CONSIDERATION)
(map! 'condition 'cons2)
(map! 'result 'cons2 )
(indiv-role! 'cons2-done 'arg 'cons2::result 'up .1)

(INSIDE-is-a! 'cons2-done 'cons1::pr-condition)

(INSIDE-is-a! 'cons1-done 'cons2::condition)

(distinct! 'cons1 'cons2)
(map! 'actor 'cons1)
(map! 'actor 'cons2)
(map! 'party1 'contract)
(map! 'party2 'contract)
(inside-equiv! 'contract::party1 'cons1::actor)
```

```
(inside-equiv! 'contract::party2 'cons2::actor)

(is-a! (map! 'expectation 'contract) 'exist-legal-obligation)

(type! 'written-agree 'agreement)
(type! 'oral-agree 'agreement)
(distinct! 'written-agree 'oral-agree)

(type! 'makeContract 'legact  'up .3)
(is-a! (map! 'result 'makeContract) 'contract)

(type! 'contract-document 'thing)

;*************************************************
;   INSTANCES
;*************************************************

(indiv! 'j-m 'contract)
(map! 'contract::party1 'j-m)
(map! 'contract::party2 'j-m)
(type! 'john 'human)
(type! 'mary 'human)
(distinct! 'john 'mary)
(equiv! 'j-m::party1 'john)
(equiv! 'j-m::party2 'mary)

(indiv! 'b-s 'agreement)
(map! 'party1 'b-s)
(map! 'party2 'b-s)
(type! 'bob 'human)
(type! 'sue 'human)
(+inhib-link! 'bob 'john_inhib_mary)
(+inhib-link! 'sue 'john_inhib_mary)
(equiv! 'b-s::party1 'john)
(equiv! 'b-s::party2 'mary)
(IS-A! (INDIV-ROLE! 'BOB-PAY-SUE 'PART_'B-S) 'PROMISE)
(is-a! 'bob-pay-sue 'legact)
```

```
(IS-A! (INDIV-ROLE! 'SUE-RETURN-BOOKS 'PART 'B-S) 'ACT 'UP .1)
(map! 'pr-condition 'bob-pay-sue)
(is-a! 'sue-return-books 'bob-pay-sue::pr-condition)

(indiv! 'ag3 'agreement)
(indiv! 'ag4 'agreement)
(is-a! 'ag3 'moral-oblig)
(is-a! 'ag4 'contract)

;**************************************************
;    ICONS
;**************************************************
; nodes that represent screen icons
(type! 'icon nil)
; the role CONCEPT-NODE is referred to in the text
; as INTERPRETATION-OF-ICON (IOI)
(type-role! 'concept-node 'arg 'icon)

(type! 'law-icon 'icon)

; the function GM-LABEL: attaches the
; string to the node that will
; be used when the screen icon is created.
(gm-label: (indiv! 'i-contract 'icon) '("Contract"))

(is-a! 'i-contract 'law-icon)

; the function SET-CONCEPTS attaches a lambda expression
; to the node that
; creates the links to the nodes possible interpretations.
; A lambda express is used, rather than directly linking the
; the interpretations so that if a clone
; is later made of the icon
; it will have the original interpretations
;  rather that the possibly
; modified interpretations.
(set-concepts 'i-contract
  (lambda (x)(is-a! 'contract x 'down .8 'up .1)
        (is-a! 'makeContract x 'down .6 'up .1)
        (is-a! 'contract-document x 'down .5 'up .1)
    ))

; The function GET-CONCEPTS generates the interpretation
; links for the node
((get-concepts 'i-contract)
    (map! 'concept-node 'i-contract))

;***************************************************


(gm-label: (indiv! 'i-act 'icon) '("Act"))
(is-a! 'i-act 'law-icon)

(set-concepts 'i-act
  (lambda (x)(is-a! 'act x 'down .6  'up .1)
        (is-a! 'cons-act x 'down .3 'up .1)))

((get-concepts 'i-act) (map! 'concept-node 'i-act))

;*****
(gm-label: (indiv! 'i-oral 'icon) '("oral""agreement"))
(is-a! 'i-oral 'law-icon)

(set-concepts 'i-oral
  (lambda (x)(is-a! 'oral-agree x 'down .6  'up .1)
        ))

((get-concepts 'i-oral) (map! 'concept-node 'i-oral))

;*****
(gm-label: (indiv! 'i-written 'icon) '("written""agreement"))
(is-a! 'i-written 'law-icon)

(set-concepts 'i-written
  (lambda (x)(is-a! 'written-agree x 'down .6  'up .1)
          ))

((get-concepts 'i-written) (map! 'concept-node 'i-written))

;*****
(gm-label: (indiv! 'i-party-proms 'icon) '("Party1 promise"
                              "to do something"
                              "for Party2."))
(is-a! 'i-party-proms 'law-icon)

(set-concepts 'i-party-proms
   (lambda (x)(is-a! 'cons1 x 'down .6 'up .1)
          (is-a! 'cons1-done x 'down .7 'up .1)
        ))

((get-concepts 'i-party-proms) (map! 'concept-node 'i-party-proms))

;*****
(gm-label: (indiv! 'i-fulfil-proms 'icon) '("Party1 fulfills"
                              "promise"
                             ))
(is-a! 'i-fulfil-proms 'law-icon)

(set-concepts 'i-fulfil-proms
   (lambda (x)(is-a! 'cons1::pr-act x 'down .7  'up .1)
          (is-a! 'cons1-pr-act-done x 'down .6 'up .1)
        ))

((get-concepts 'i-fulfil-proms) (map! 'concept-node 'i-fulfil-proms))

;*****
(gm-label: (indiv! 'i-party2-return 'icon) '("Party2 does"
                              "something"
                              "in return."))
(is-a! 'i-party2-return 'law-icon)

(set-concepts 'i-party2-return
   (lambda (x)(is-a! 'cons2 x 'down .6  'up .1)
          (is-a! 'cons2-done x 'down .8 'up .1)
        ))

((get-concepts 'i-party2-return) (map! 'concept-node 'i-party2-return))

;*****

(gm-label: (indiv! 'i-condition 'icon) '("Condition"))
(is-a! 'i-condition 'law-icon)

(set-concepts 'i-condition
  (lambda(x) (is-a! 'condition x 'down .5 'up .1)
         (is-a! 'pr-condition x 'down .3 'up .1)))

((get-concepts 'i-condition)(map! 'concept-node 'i-condition) )


;*******
(gm-label: (indiv! 'i-party1 'icon) '("Party 1"))
(is-a! 'i-party1 'law-icon)

(set-concepts 'i-party1
  (lambda(x) (is-a! 'party1 x 'down .8 'up .1)))

((get-concepts 'i-party1) (map! 'concept-node 'i-party1))

;*****

(gm-label: (indiv! 'i-party2 'icon) '("Party 2"))
(is-a! 'i-party2 'law-icon)

(set-concepts 'i-party2
  (lambda(x) (is-a! 'party2 X 'down .8 'up .1)))

((get-concepts 'i-party2) (map! 'concept-node 'i-party2))
```

```
;*****
(gm-label: (indiv! 'i-relationship 'icon) '("Relationship"))
(is-a! 'i-relationship 'law-icon)


(set-concepts 'i-relationship
  (lambda(x) (is-a! 'relationship X 'down .8 'up .1)))

((get-concepts 'i-relationship)(map! 'concept-node 'i-relationship) )

;*******
(gm-label: (indiv! 'i-promise 'icon) '("Promise"))
(is-a! 'i-promise 'law-icon)


(set-concepts 'i-promise
  (lambda(x) (is-a! 'promise X 'down .8 'up .1)
        (is-a! 'cons-prom X 'down .5 'up .1)))

((get-concepts 'i-promise) (map! 'concept-node 'i-promise))

;***
(gm-label: (indiv! 'i-forbearance 'icon) '("Forbearance"))
(is-a! 'i-forbearance 'law-icon)

(set-concepts 'i-forbearance
  (lambda(x) (is-a! 'forbearance X 'down .8 'up .1)
        (is-a! 'cons-forb X 'down .5 'up .1)))

((get-concepts 'i-forbearance) (map! 'concept-node 'i-forbearance))

;***
(gm-label: (indiv! 'i-not-doing 'icon) '("Not doing what"
                              "one has the right"
                              "to do"))
(is-a! 'i-not-doing 'law-icon)
```

```
(set-concepts 'i-not-doing
  (lambda(x) (is-a! 'forbearance X 'down .8 'up .1)
        (is-a! 'cons-forb X 'down .5 'up .1)))

((get-concepts 'i-not-doing) (map! 'concept-node 'i-not-doing))

;****
(gm-label: (indiv! 'i-illegact 'icon) '("Illegal" "Act"))
(is-a! 'i-illegact 'law-icon)

(set-concepts 'i-illegact
  (lambda(x) (is-a! 'illegact X 'down .8 'up .1)))

((get-concepts 'i-illegact) (map! 'concept-node 'i-illegact))

;***
(gm-label: (indiv! 'i-legact 'icon) '("Legal" "Act"))
(is-a! 'i-legact 'law-icon)

(set-concepts 'i-legact
  (lambda(x) (is-a! 'legact X 'down .8 'up .1)))

((get-concepts 'i-legact) (map! 'concept-node 'i-legact))

;***
(gm-label: (indiv! 'i-optact 'icon) '("Optional" "Act"))
(is-a! 'i-optact 'law-icon)

(set-concepts 'i-optact
  (lambda(x) (is-a! 'optact X 'down .8 'up .1)))

((get-concepts 'i-optact) (map! 'concept-node 'i-optact))

;***
(gm-label: (indiv! 'i-consideration 'icon) '("Consideration"))
(is-a! 'i-consideration 'law-icon)
```

```
  (lambda(x) (is-a! 'b-s X 'down .8 'up .1)))

((get-concepts 'i-b-s) (map! 'concept-node 'i-b-s)  )



(herald birds)

;**************************************************
;   REPRESENTATION FOR BIRD AND FISH
;   ICONS USED AS TRAINING DOMAIN
;**************************************************


(type! 'bird 'animal)
(type! 'sparrow 'bird)
(type! 'wing 'thing)  ;
(type! 'flap-wings 'act)
(type! 'fly 'act)
(map! 'condition  'fly)
(is-a! 'flap-wings 'fly::condition)
(is-a! (type-role! 'bird-wing 'part 'bird) 'wing)
(type! 'fish 'animal)
(inhib-link! 'animals 'animal)
(+inhib-link! 'fish 'animals)
(+inhib-link! 'bird 'animals)
(indiv-role! 'has-feathers 'arg 'bird 'up .1)
(indiv-role! 'can-fly 'arg 'bird 'up .1)
(indiv-role! 'has-scales 'arg 'fish 'up .1)

;**************************************************
;  DEFINE ICONS FOR BIRD DOMAIN
;**************************************************
```

```
(type! 'animal-icon 'icon)

;**********************

(gm-label: (indiv! 'i-bird 'icon) '("bird"))

(is-a! 'i-bird 'animal-icon)

(set-concepts 'i-bird
  (lambda (x)(is-a! 'bird x 'down .8 'up .1)
     ))

((get-concepts 'i-bird)
     (map! 'concept-node 'i-bird))
;**********************
(gm-label: (indiv! 'i-sparrow 'icon) '("sparrow"))

(is-a! 'i-sparrow 'animal-icon)

(set-concepts 'i-sparrow
  (lambda (x)(is-a! 'sparrow x 'down .8 'up .1)
     ))

((get-concepts 'i-sparrow)
     (map! 'concept-node 'i-sparrow))
;**********************
(gm-label: (indiv! 'i-flap-wings 'icon) '("flap-wings"))

(is-a! 'i-flap-wings 'animal-icon)

(set-concepts 'i-flap-wings
  (lambda (x)(is-a! 'flap-wings x 'down .8 'up .1)
     ))

((get-concepts 'i-flap-wings)
     (map! 'concept-node 'i-flap-wings))
;**********************
```

```
(set-concepts 'i-consideration
  (lambda(x) (is-a! 'consideration X 'down .8 'up .1)))

((get-concepts 'i-consideration) (map! 'concept-node 'i-consideration))

;***
(gm-label: (indiv! 'i-return-for-prom 'icon) '("Something done in"
                                   "return for a promise"))
(is-a! 'i-return-for-prom 'law-icon)

(set-concepts 'i-return-for-prom
  (lambda(x) (is-a! 'consideration X 'down .8 'up .1)))

((get-concepts 'i-return-for-prom) (map! 'concept-node 'i-return-for-prom))

;***
(gm-label: (indiv! 'i-cons-act 'icon) '("Consideration" "Act"))
(is-a! 'i-cons-act 'law-icon)

(set-concepts 'i-cons-act
  (lambda(x) (is-a! 'cons-act X 'down .8 'up .1)))

((get-concepts 'i-cons-act)(map! 'concept-node 'i-cons-act) )

;***
(gm-label: (indiv! 'i-cons-prom 'icon) '("Consideration" "Promise"))
(is-a! 'i-cons-prom 'law-icon)

(set-concepts 'i-cons-prom
  (lambda(x) (is-a! 'cons-prom X 'down .8 'up .1)))

((get-concepts 'i-cons-prom) (map! 'concept-node 'i-cons-prom))

;***
(gm-label: (indiv! 'i-cons-forb 'icon) '("Consideration" "Forbearance"))
(is-a! 'i-cons-forb 'law-icon)
```

```
(set-concepts 'i-cons-forb
  (lambda(x) (is-a! 'cons-forb X 'down .8 'up .1)))

((get-concepts 'i-cons-forb) (map! 'concept-node 'i-cons-forb))

;***
(gm-label: (indiv! 'i-agreement 'icon) '("Agreement"))
(is-a! 'i-agreement 'law-icon)

(set-concepts 'i-agreement
  (lambda(x) (is-a! 'agreement X 'down .8 'up .1)))

((get-concepts 'i-agreement) (map! 'concept-node 'i-agreement) )

;***
(gm-label: (indiv! 'i-moral-oblig 'icon) '("Moral" "Obligation"))
(is-a! 'i-moral-oblig 'law-icon)

(set-concepts 'i-moral-oblig
  (lambda(x) (is-a! 'moral-oblig X 'down .8 'up .1)
         (is-a! 'exist-moral-obligation X 'down .7 'up .1)  ))
((get-concepts 'i-moral-oblig) (map! 'concept-node 'i-moral-oblig))

;***
(gm-label: (indiv! 'i-legal-oblig 'icon) '("Legal" "Obligation"))
(is-a! 'i-legal-oblig 'law-icon)

(set-concepts 'i-legal-oblig
  (lambda(x)
         (is-a! 'exist-legal-obligation X 'down .7 'up .1)  ))
((get-concepts 'i-legal-oblig) (map! 'concept-node 'i-legal-oblig))

;***
(gm-label: (indiv! 'i-b-s 'icon) '("Bob & Sue's" "Agreement"))
(is-a! 'i-b-s 'law-icon)

(set-concepts 'i-b-s
(gm-label: (indiv! 'i-fly 'icon) '("fly"))

(is-a! 'i-fly 'animal-icon)

(set-concepts 'i-fly
  (lambda (x)(is-a! 'fly x 'down .8 'up .1)
        (is-a! 'can-fly x 'down .5 'up .1)
    ))

((get-concepts 'i-fly)
    (map! 'concept-node 'i-fly))
;*********************

(gm-label: (indiv! 'i-wing 'icon) '("wing"))

(is-a! 'i-wing 'animal-icon)

(set-concepts 'i-wing
  (lambda (x)(is-a! 'wing x 'down .8 'up .1)
        (is-a! 'bird-wing x 'down .5 'up .1)
    ))

((get-concepts 'i-wing)
    (map! 'concept-node 'i-wing))
;*********************
(gm-label: (indiv! 'i-fish 'icon) '("fish"))

(is-a! 'i-fish 'animal-icon)

(set-concepts 'i-fish
  (lambda (x)(is-a! 'fish x 'down .8 'up .1)
    ))

((get-concepts 'i-fish)
    (map! 'concept-node 'i-fish))
;*********************
(gm-label: (indiv! 'i-has-feathers 'icon) '("has-feathers"))
```

```
(is-a! 'i-has-feathers 'animal-icon)

(set-concepts 'i-has-feathers
  (lambda (x)(is-a! 'has-feathers x 'down .8 'up .1)
    ))

((get-concepts 'i-has-feathers)
    (map! 'concept-node 'i-has-feathers))
;*********************
(gm-label: (indiv! 'i-has-scales 'icon) '("has-scales"))

(is-a! 'i-has-scales 'animal-icon)

(set-concepts 'i-has-scales
  (lambda (x)(is-a! 'has-scales x 'down .8 'up .1)
    ))

((get-concepts 'i-has-scales)
    (map! 'concept-node 'i-has-scales))


;****************************************************
;****************************************************
;**                                                **
;**            PRODUCTIONS REPRESENTING            **
;**            SHERLOCK'S TUTORING STRATEGY        **
;**                                                **
;****************************************************
;****************************************************


(herald tutoring_strats)


(set *tutoring-strats* '())
```

```
; Fully annotated examples of the tutoring productions are
; included in Chapter 4.

; Each production has the following components:
;         name
;         list of antecedents
;         explanation  (this was not included in the productions shown
;                         in chapter 4.  It is used
;                         to print an explanatory
;                         message to the transcript window when a
;                         production is chosen)
;         list of consequents
;         strength


(tproduction!
 'nothing
 '(
  t
  )
 '(msg t "Couldn't find anything to fire" t)
 '(
  (msg t "popping goal because couldn't find anything." t)
  (pop *goal*)
  )
 -1)


(tproduction!
 'get-new-move
 '(
  (eq-goal? 'get-new-move)
  )
 '(msg t "Ready for learner's next move." t)
 '(
  (msg t "Get learner's next move" t)
  (pop *goal*)
  (get-new-move)
  )
 )

;*******************************************************
; GOAL = EVAL LINK
;*******************************************************

(tproduction!
 'no-prods
 '(
  (eq-goal? 'eval-link)
  (not *possible-prods*)
  )
 '(msg t "You want to evaluate the link and you have not propagated to"
       t "see what it might mean." t  "Do propagations now" t)
 '(
  (check-prod-dub (caar *from-icon-children*)(caar *to-icon-children*))
  )
 1)

(tproduction!
 'check-best
 '(
  (eq-goal? 'eval-link)
  *sherlocks-move*
  (eq? (production-link *sherlocks-move*) *link-made*)
  )
 '(msg t "The link made matches what s would have done." t)
 '(
  (set *plan-used* *sherlocks-move*)
  (set *plan-used-status* 'ok)
  (pop *goal*)
  (goal! 'change-net)
  )
 .9)
```

```
(tproduction!
 'check-ok
 '(
  (eq-goal? 'eval-link)
   *other-good-moves*
  (some (lambda (x)(if (eq? (production-link x) *link-made*)
                  (set tmp x)
                  nil))
       *other-good-moves*)
 )
 '(msg t "The link made matches one of S less optimal moves." t)
 '(
  (set *plan-used* tmp)
  (set *plan-used-status* 'ok)
  (pop *goal*)
  (goal! 'change-net)
 )
 .8)

(tproduction!
 'find-plan
 '(
  (eq-goal? 'eval-link)
  (not *plan-used*)
   *possible-prods*
 )
 '(msg t "The link doesn't match anything S would have done." t)
 '(
  (goal! 'find-plan)
 )
 .5)

(tproduction!
 'check-facts
 '(
  (eq-goal? 'eval-link)
```

```
   *plan-used*
   (not *new-interp*)
  )
 '(msg t "Check which facts are true now." t)
 '(
   (prop-rules *reasons*)
   (set *new-interp* (make-interp (sort-facts)))
  )
 .5)

(tproduction!
 'give-up
 '(
  (eq-goal? 'eval-link)
  (not *plan-used*)
   (not *possible-prods*)
  )
 '(msg t "No possible productions for those nodes." t)
 '(
;  (output '(c "I am not sure that I would have made" "a link there."))
  (goal! 'find-plan)
  )
 .5)


(tproduction!
 'new-from-icon
 '(
  (eq-goal? 'eval-link)
  (< (accuracy *new-interp*) .8)
  (>  (length *from-icon-children*) 1)
 )
'(msg t "There seems to be a text problem but the icon has alternative"
     t "interpretations.  Run the evaluations again with the next "
     t "FROM alternative.")
'(
  (store-interp)
```

```
   (set *possible-prods* nil)
   (goal! 'eval-link)
   (pop *from-icon-children*)
  )
 .4)


(tproduction!
 'first-new-to-icon
 '(
   (eq-goal? 'eval-link)
   (< (accuracy *new-interp*) .8)
   (>  (length *to-icon-children*) 1)
   (eq? (length *to-icon-children*)
       (length (icon-children *to-icon*)))
  )
 '(msg t
 "There seems to be a text problem but the icon has alternative"
 t "interpretations.  Run the evaluations again with the first "
     t "TO alternative.")
 '(
   (set *from-icon-children* (icon-children *from-icon*))

   (store-interp )
   (set *possible-prods* nil)
   (goal! 'eval-link)
   (pop *to-icon-children*)
  )
 .3)

(tproduction!
 'new-to-icon
 '(
   (eq-goal? 'eval-link)
   (< (accuracy *new-interp*) .8)
   (>  (length *to-icon-children*) 1)
  )
```

```
 '(msg t
  "There seems to be a text problem but the icon has alternative"
   t "interpretations.  Run the evaluations again with the next "
     t "TO alternative.")
 '(
   (store-interp )
   (set *possible-prods* nil)
   (goal! 'eval-link)
   (pop *to-icon-children*)
  )
 .2)


;*******************************************************
;  GOAL = Feed back
;*******************************************************

(tproduction!
 'not-evald
 '(
   (eq-goal? 'feedback)
   (not *possible-prods*)
  )
 '(msg t "You want to give feedback, but haven't evaled yet"
      t)
 '(
   (goal! 'eval-link)
  )
 1)

(tproduction!
 'alt-froms
 '(
  (eq-goal? 'feedback)
  (neq? *plan-used-status* 'ok)
   *interp*
   (> (accuracy *interp*)(accuracy *new-interp*))
  )
 '(msg t "previous interpretation was better, use it" t)
 '(
   (use-interp *interp*)
  )
  .9)

 (tproduction!
  'no-alt-froms
  '(
   (eq-goal? 'feedback)
   (neq? *plan-used-status* 'ok)
   (not (> (accuracy *interp*)(accuracy *new-interp*)))
  )
 '(msg t "no previous interpretation was better.  Make the" t
            "present interpretation the only interpretation." t)
 '(
   (set *new-interp* (make-interp (sort-facts)))
   (use-interp *new-interp*)
  )
  .9)


 (tproduction!
  'good-ok
  '(
   (eq-goal? 'feedback)
   *plan-used*
   (eq? *plan-used-status* 'ok)
   (> (rule-strength *plan-used*) .5)
  )
 '(msg t "The link made matches one of S less optimal moves." t)
 '(
   (output-msg '(c "I might not have done that,"
             t "But it looks like a good move.")
            *gm-output-little*)
```

```
   (pop *goal*)
  )
  .7)

 (tproduction!
  'best-ok
  '(
   (eq-goal? 'feedback)
   *plan-used*
   (eq? *plan-used* *sherlocks-move*)
   (eq? *plan-used-status* 'ok)
   (> (rule-strength *plan-used*) .2)
  )
 '(msg t "The link made matches one of S less optimal moves." t)
 '(
   (output-msg '( c "Very good!"
            t "That is what I would have done.")
     *gm-output-little*)
   (pop *goal*)
  )
  .8)

 (tproduction!
  'why
  '(
   (eq-goal? 'feedback)
   *plan-used*
   (eq? *plan-used-status* 'ok)
   (= (rule-strength *plan-used*) 0)
  )
 '(msg t "The link made matches one of S less optimal moves." t)
 '(
   (output-msg '(c "That might be true,"
            t "But... Who cares?")
            *gm-output-little*)
   (pop *goal*)
  )
 .8)

(tproduction!
 'fair-ok
 '(
  (eq-goal? 'feedback)
  *plan-used*
  (eq? *plan-used-status* 'ok)
  (> (rule-strength *plan-used*) 0)
  (< (rule-strength *plan-used*) .6)
 )
 '(msg t "The link made matches one of S less optimal moves." t)
 '(
  (output-msg '(c "That is OK"
           t)
            *gm-output-little*)
  (pop *goal*)
 )
 .8)

(tproduction!
 'no-plan
 '(
  (eq-goal? 'feedback)
  (not *plan-used*)
 )
 '(msg t "Never found a plan." t)
 '(
  (goal! 'tutor-plan)
 )
 .8)

(tproduction!
 'bad-plan
 '(
  (eq-goal? 'feedback)
  *plan-used*
```

```
  (or
   (< (rule-strength *plan-used*) 0)
   (neq? (production-link (get-production *plan-used*)) *link-made*)
  )
 )
 '(msg t "The plan found is buggy." t)
 '(
  (goal! 'tutor-plan)
 )
 .8)

(tproduction!
 'wrong-link
 '(
  (eq-goal? 'feedback)
  *plan-used*
  nil
  (neq? (production-link (get-production *plan-used*))
    *link-made*)
 )
 '(msg t "The plan found is buggy." t)
 '(
  (goal! 'tutor-plan)
 )
 .8)

(tproduction!
 'bad-text
 '(
  (eq-goal? 'feedback)
  (< (accuracy *interp*) 1)   ; changed 6-21-88 so that any
                                          ; misinformation can at least be
                                          ; considered (used to be .8)
 )
 '(msg t "Some mis-information was found." t)
 '(
  (goal! 'tutor-text)
  )
 .8)

(tproduction!
  'change-mind
  '(
   (eq-goal? 'feedback)
   (or (< (accuracy *interp*) .7)
      (not *plan-used*)
      (<  (production-strength
      (get-production *plan-used*)) .1 )
   )
  )
  '(msg t "not a good move, ask if want to remove" t)
  '(
   (offer-change)
  )
 .1)


;*******************************************************
;  GOAL = CHANGE NET
;*******************************************************

(tproduction!
  'assign-concept-role
  '(
   (eq-goal? 'change-net)
   (eq? *link-made* 'part)
   (eq? *plan-used-status* 'ok)
   (not (eq? f-child-role-t  *f*))
  )
 '(msg t "from was used to represent a specialization." t)
 '(
   (msg t "changing net" t
     *from-icon* " will refer to " f-child-role-t)
```

```
   (assign-concept f-child-role-t *from-icon*)
   (pop *goal*)
   (goal! 'get-new-move)
  )
 .9)

(tproduction!
  'assign-concept-from
  '(
   (eq-goal? 'change-net)
   (not(eq? (car *from-icon-children*)(car (icon-children *from-icon*))))
  )
  '(msg t
   "Alternate interpretation for icon was ok so change net." t)
  '(
   (msg t "changing net" t
     *f* " will refer to " (caar *from-icon-children*))
   (assign-concept (caar *from-icon-children*) *from-icon*)
   (pop *goal*)
   (goal! 'get-new-move)
  )
 .9)

(tproduction!
  'assign-concept-to
  '(
   (eq-goal? 'change-net)
   (not(eq? (car *to-icon-children*)(car (icon-children *to-icon*))))
  )
 '(msg t
  "Alternate interpretation for icon was ok so change net." t)
 '(
  (msg t "changing net" t
    *t* " will refer to " (caar *to-icon-children*))
  (assign-concept (caar *to-icon-children*) *to-icon*)
  (pop *goal*)
  (goal! 'get-new-move)
  )
 .9)

;************************************************************
; GOAL = FIND PLAN
;******************************,*****************************

(tproduction!
 'found-plan
 '(
  (eq-goal? 'find-plan)
  *plan-used*
  )
 '(msg t "Was looking for plan, one was found" t)
 '(
   (msg t "Subject used plan: " *plan-used* t)
   (set *new-interp* (make-interp (sort-facts) ))
  (pop *goal*)
  )
 .1)


(tproduction!
 'no-found-plan
 '(
  (eq-goal? 'find-plan)
  )
 '(msg t "couldnt find plan, set new-interp anyway" t)
 '(
   (set *new-interp* (make-interp (sort-facts) ))
  (pop *goal*)
  )
 0)

(tproduction!
 'check-all-plans
 '(
  (eq-goal? 'find-plan)
  (not *plan-used*)
  )
 '(msg t "trying to find plan, check all possibilities." t)
 '(
  (set *possible-plans*
   (pick-plan (append
           (all (lambda (x)
         (eq? (production-link x) *link-made*))
                *buggy-moves*)
           (if *sherlocks-move*
             (list *sherlocks-move*)
             nil)
           (prods-with-goal *link-made*) )))
  (goal! 'pick-plan)
  )
 .9)


(tproduction!
 'check-buggy-plans
 '(
  (eq-goal? 'find-planX)
  (not *plan-used*)
  *buggy-moves*
  (set tmp (all (lambda (x)
     (eq? (production-link x) *link-made*))
           *buggy-moves*))
  )
 '(msg t "trying to find plan, check the buggy productions." t)
 '(
   (set *possible-plans* (pick-plan tmp))
  (goal! 'pick-plan)
  )
 .7)

(tproduction!
 'check-sher-plan
 '(
  (eq-goal? 'find-planX)
  (not *plan-used*)
   *sherlocks-move*
  )
 '(msg t "trying to find plan, check to see if the learner" t
          "agrees with the facts for sherlock's rule." t)
 '(
    (pick-fact (list *sherlocks-move*))
;    (goal! 'pick-plan)
  )
 .4)


(tproduction!
 'check-other-plans
 '(
  (eq-goal? 'find-planX)
  (not *plan-used*)
  )
 '(msg t
    "trying to find plan, check productions with same link." t)
 '(
    (set *possible-plans*
          (pick-plan (prods-with-goal *link-made*)))
    (goal! 'pick-plan)
  )
 .5)

;*********************************************************
;  GOAL = Pick Plan
;*********************************************************

; minimum activation for plan to be considered the one
; the learner was using
```

```
(lset *plan-threshold* 70)


;the first plan is above the threshold, and is greater than the second
(tproduction!
 'pick-plan
 '(
   (eq-goal? 'pick-plan)
   (not *plan-used*)
    *possible-plans*
   (> (cadar *possible-plans*) *plan-threshold*)
   (not (almost-equal? (cadadr *possible-plans*)
                              (cadar *possible-plans*)))
  )
 '(msg t "Pick a plan out of possible plans." t)
 '(
 (set *plan-used* (caar *possible-plans*))
  )
 .8)

;  This is a dummy production eventually to be replaced by some
;  strategy that will ask the learner questions to determine which
;  of the top two productions is most likely
(tproduction!
 'disambiguate
 '(
   (eq-goal? 'pick-plan)
   (not *plan-used*)
    *possible-plans*
   (> (cadar *possible-plans*) *plan-threshold*)
   (almost-equal? (cadadr *possible-plans*)
                              (cadar *possible-plans*))
  )
 '(msg t "Pick a plan out of possible plans." t)
 '(
 (msg t T "PICK THE RIGHT PLAN" T T )
  )
 .8)

(tproduction!
 'disambiguate-stronger
 '(
  (eq-goal? 'pick-plan)
  (not *plan-used*)
   *possible-plans*
  (> (cadar *possible-plans*) *plan-threshold*)
  (almost-equal? (cadadr *possible-plans*)
                           (cadar *possible-plans*))
  (not(almost-equal? (production-strength (node-english (caar *possible-
plans*)
))
      (production-strength (node-english
                  (caadr *possible-plans*))))
 )
 )
 '(msg t "Pick the strongest plan." t)
 '(
   (set *plan-used*
     (if (> (production-strength
                  (node-english (caar *possible-plans*)))
           (production-strength
                  (node-english (caadr *possible-plans*)))
     )
        (caar *possible-plans*)
        (caadr *possible-plans*)
     ))
 )
 .8)

(tproduction!
 'fulfil-plan
 '(
  (eq-goal? 'pick-plan)
```

```
  (not *plan-used*)
   *possible-plans*
  (<= (cadar *possible-plans*) .7)
  (> (cadar *possible-plans*) .4)
 )
 '(msg t "Pick a plan out of possible plans." t)
 '(
 (set *plan-used* (caar *possible-plans*))
 )
 .8)


;**********************************************************
; GOAL = TUTOR TEXT
;**********************************************************

(tproduction!
 'wrong-facts
 '(
  (eq-goal? 'tutor-text)
;  (< (accuracy *interp*).8)  taken out 6-21-88.  if it makes
;  the learners plan wrong, it should be noted.
  *plan-used*
  (> (length *why-wrong-learners-plan*) 0 )
  )
'(msg t "show the learner wrong facts that led to his" t
       "bad plan" t)
'(
 (output-msg '(c "You seem to believe that: " t))
 (explain-fact *why-wrong-learners-plan*)
 (output-msg '(t "I do not agree." ))
 (run-menu 'ok-menu)
 (hide-menu 'ok-menu)
 )
.7
)
```

```
(tproduction!
 'missing-facts
 '(
  (eq-goal? 'tutor-text)
  (< (accuracy *interp*) .9)
   *sherlocks-move*
   (> (production-strength *sherlocks-move*) .1)
  (> (length  *why-not-sherlocks-move* ) 0 )
   )
 '(msg t "show the learner missing facts that would have" t
          "led to sherlock's plan." t)
 '(
  (output-msg '(c "Were you aware that: " t))
   (explain-fact  *why-not-sherlocks-move* )
  (output-msg '(t "Would this change your choice of link?" ))
  (run-menu 'ok-menu)
  (hide-menu 'ok-menu)
  )
 .4
)


(tproduction!
 'check-over-gen
 '(
  (eq-goal? 'tutor-text)
  (eq? *link-made* 'part)
   )
 '(msg t "there is a text problem.  A part link was made."
      t "Check to see if a descendent of the to node would"
      t "have that part." t)
 '(
   (set tmp (desc-has-role *t* *f* 3))
   (if tmp
    (block
     (show-over-gen)
      (pop *goal*)
     )
     nil) )
 .6
 )

;**********************************************************
;  GOAL = TUTOR PLAN
;**********************************************************

(tproduction!
 'plan-tutor
 '(
   (eq-goal? 'tutor-plan)
   *plan-used*
    )
 '(msg t "The goal is to tutor plan (there was a plan found)." t)
 '(
   (output-msg '(c "You used the plan: " t) )
   (general-explain *plan-used*)
   (output-msg '(t "That is not a good plan." t ))
   (exec-prod *plan-used*)
  (run-menu 'ok-menu)
  (hide-menu 'ok-menu)
  )
 .7
)

(tproduction!
 'suggest-link
 '(
  (eq-goal? 'tutor-plan)
  *plan-used*
  (neq? (production-link (get-production *plan-used*))
                               *link-made*)
   )
'(msg t "The goal is to tutor plan (there was a plan found)." t)
```

```
 '(
   (set tmp  (production-link (get-production *plan-used*)) )
    (output-msg '(t "*************************************" t
       "At times like this I would make a "
     tmp  " link."  t) )
   (run-menu 'ok-menu)
   (hide-menu 'ok-menu)
  )
 .6
 )

(tproduction!
 'what-plan
 '(
   (eq-goal? 'tutor-plan)
   (not *plan-used*)
   *reasons*
    )
 '(msg t "The goal is to tutor plan (there was no plan found)." t)
 '(
  (output-msg '(c "You seem to have used the plan: " t))
   (explain-list *reasons*)
  (output-msg '(t "I do not recognize that plan." t))
  (run-menu 'ok-menu)
  (hide-menu 'ok-menu)
 )
 .7
)


(herald strategy
   (read-table (*value user-env 'ami-read-table)) )

;*****************************************************
;*****************************************************
;**                                                 **
;**       STRATEGY FOR BUILDING A GRAPHIC MAP       **
```

```
;**
;***************************************************
;***************************************************
;

; FACTS USED IN DECIDING WHAT TYPE OF GM-LINK SHOULD BE USED
;    ***********************************************
(set *min* 10)

;  Facts are represented in the semantic network.
;  Whether a fact is true
;  should be represented by links in network.
;  Currently, however, the truth status of a
;  fact is stored separately
;  the network.

;  Create atoms for each of the facts used
;  by Sherlock's graphic mapping
;  strategies.  These
;  atoms are used to store the English generation
;  of the facts and
;  if the fact is currently true.
(set r-f-is-t (list nil
          '( "Every " *from-icon-english*
                   " is  a " *to-icon-english* )))

(set r-t-is-f (list nil
          '( "Every " *to-icon-english*
                   " is a " *from-icon-english* )))

(set r-t-unlik-f (list nil
          '("It is not as likely that a "
                  *to-icon-english* " is a " *from-icon-english*)))

(set r-f-unlik-t (list nil
          '("It is not as likely that a "
                  *from-icon-english* " is a " *to-icon-english*)))
```

```
(set r-f-not-t  (list nil
          '("There is no chance that a "
                    *from-icon-english* " is a " *to-icon-english*)))

(set r-t-not-f  (list nil
          '("There is no chance that a "
                    *to-icon-english* " is a " *from-icon-english*)))

(set r-eq-ch  (list nil
          '("The liklihood of  a " *to-icon-english*
                    " being a " *from-icon-english* " is equal to the
                    reverse.")) )

(set r-same-conc (list nil
          '(*to-icon-english* " and " *from-icon-english*
                    " refer to the same concept.")))

(set r-t-has-f (list nil
          '("A " *to-icon-english* " usually has a "
                    *from-icon-english* )))

(set r-f-thing (list nil
          '(*from-icon-english* " is a thing.")))

(set r-f-part-t (list nil
          '(*from-icon-english* " is a portion of something."))  )

(set r-t-cond-act (list nil
          '(*to-icon-english* " is an act which has a condition.")))

(set r-f-cond-t (list nil
          '(*from-icon-english* " can be a condition of "
                    *to-icon-english*)))

(set r-f-maybe-t (list nil
          '(*from-icon-english* " might be a " *to-icon-english* )))
```

```
(set r-t-fact (list nil
          '(*from-icon-english* " is a fact.")))

(set r-t-expect-f (list nil
          '("If there is a " *from-icon-english*
                    " there will be a  " *to-icon-english* ".")))

(set r-t-result-f (list nil
          '("Doing  " *from-icon-english*
                    " will cause " *to-icon-english* ".")))


; This function sets the TRUTH component
; of the list associated with
; each of the fact atoms according to the
; result of the most recent
; spreading activation.
(define (check-facts)
 (set (car r-f-is-t )  (> *f->t* *min*))
 (set (car r-t-is-f )  (> *t->f* *min*))
 (set (car r-t-unlik-f )  (> *f->t* *t->f*))
 (set (car r-f-unlik-t )  (> *t->f* *f->t*) )
 (set (car r-f-not-t )  (< *f->t* 1))
 (set (car r-t-not-f  ) (< *t->f* 1))
 (set (car r-eq-ch  )  (< ( abs (- *f->t*  *t->f*)) 5))
 (set (car r-same-conc )
          (intersection-cars *from-icon-children* *to-icon-children*))
 (set (car r-t-has-f )  (or  f-role-t f-child-role-t)  )
 (set (car r-f-thing )  arg-f )
 (set (car r-t-fact)  fact-t)
 (set (car r-f-part-t )   part-f)
 (set (car r-t-cond-act )   cond-t)
 (set (car r-f-cond-t )
          (and f-child-role-t cond-t
          (p-is-a? f-child-role-t (get-node 'condition))))
 (set (car r-t-expect-f )
          (and t-child-role-f
          (p-is-a? t-child-role-f (get-node 'expectation))))
 (set (car r-t-result-f )
          (and t-child-role-f
          (p-is-a? t-child-role-f (get-node 'result))))
 (set (car r-f-maybe-t ) (> *f->t* 0))
)


; PRODUCTIONS USED TO DECIDE WHAT TYPE OF GM-LINK SHOULD BE
MADE
;    ********************************************************
; The explanation of each of these productions
;  is included in Chapter 4.
(production!
 'is-a
 '(
  (r-f-is-t .7)
  (r-t-unlik-f .3)
 )
 '(
  (make-link 'is-a *f* *t*)
 )
 .8)

(production!
 'weak-is-a
 '(
  (r-f-maybe-t .4)
  (r-t-unlik-f .6)
 )
 '(
  (make-link 'is-a *f* *t*)
  (output-msg
  '(c "That might be true..." t "But, who cares?")
          *gm-output-little*)
```

```
  )
  0)

 (production!
  'not
  '(
   (r-f-not-t .5)
   (r-t-not-f .5)
 )
  '(
    (make-link 'not *f* *t*)
   )
  .1)


 (production!
  'rev-is-a
  '( (r-t-is-f .7)
     (r-f-unlik-t .3)
     )
 '(
   (make-link 'is-a *f* *t*)
   (output-msg
          '("I think you are making your IS-A link in the wrong"
                t  "direction."))
  )
 -.5
 t)

(production!
 'half-is-a
 '(
   (r-f-is-t .9)
 )
 '(
   (make-link 'is-a *f* *t*)
  (output-msg
           '("That is too general.
           Unless you also know that there is less chance"
    t "that a " *t-icon-string* " is a " *f-icon-string* ", an  "
            t "EQUIV link might be more appropriate."))
 )
 -.5
 t)


(production!
 'dub-is-a
 '(
  (r-f-is-t .4)
  (r-t-is-f .4)
 )
 '(
   (make-link 'equiv *f* *t*)
   (output-msg '("You may want a one way IS-A link instead" ))
  )
 -.4)

(production!
 'equal-high
 '(
  (r-f-is-t  .7)
  (r-eq-ch   .2)
 )
 '(
   (make-link 'equiv *f* *t*)
  )
 .5)

(production!
 'same
 '(
  (r-same-conc .9)
 )
```

```
 '(
   (make-link 'equiv *f* *t*)
  )
 .5)


(production!
 'part
 '(
  (r-t-has-f .6)
  (r-f-part-t .4)
 )
 '(
   (make-link 'part *f* *t*)
 )
 1)

(production!
 'maybe-part
 '(
  (r-t-has-f .9)
  )
 '(
   (make-link 'part *f* *t*)
 )
 .2)


(production!
 'prop
 '(
  (r-t-has-f .9)
  )
 '(
   (make-link 'prop *f* *t*)
 )
 .9)
```

```
(production!
 'is-one
 '(
  (r-t-is-f)
 )
 '(
   (make-link 'prop *f* *t*)
   (output-msg '("If all Y's have the properties of X then" t
            "then maybe Y IS-A X."))
 )
 -0.5)

(production!
 'prop-leads
 '(
  (r-t-has-f .9)
 )
 '(
   (make-link 'leads *f* *t*)
   (output-msg '("If all Y's have an X, I would say that" t
            "X is a part of Y."))
)
 -0.8)


(production!
 'leads
 '(
  (r-f-cond-t .9)
 )
 '(
  (make-link 'leads *f* *t*)
))


(production!
```

```
 'expect
 '(
   (r-t-expect-f .8)
  )
 '(
   (make-link 'leads *f* *t*)
 ))

(production!
 'results
 '(
   (r-t-result-f .8)
  )
 '(
   (make-link 'leads *f* *t*)
 ))


; The productions for building a graphic
; map should be a part of the
; larger group of productions for tutoring.
; The only distinction should
; be that the each have an antecedent:
;                      (eq-goal?   'make-a-link)
; Currently, however, the graphic mapping productions are kept
; separate and processed separately, and
; thus this antecedent would
; be superfluous.
; These productions are stored on a
; separate list (*STRATEGIES*).
; Only the productions on this list are
; considered when deciding what
; type of link should be made.
(define (reset)
 (set *strategies*
   (list
     $is-a $weak-is-a $not   $rev-is-a $half-is-a $dub-          $is-a
$equal-high
     $same $part $maybe-part   $prop $is-one
          $prop-leads $leads
     $expect $results )))

(reset)


;**************************************************
;**************************************************
;**                                              **
;**      FUNCTIONS FOR HANDLING SEMANTIC NET     **
;**              REPRESENTATION OF RULES         **
;**                                              **
;**************************************************
;**************************************************

(herald ruleNet
(read-table (*value user-env 'ami-read-table)) )


(set *min* 10)

; UTILITIES FOR DEALING WITH RULES IN THE SEMANTIC NETWORK
;    ************************************************

; Prints the currect activations on all roles of NODE
(define (check-role-strength node)
 (walk (lambda (x) (aa (car x))
          (msg t "------------------------" ))
     (node-roles node)))

(define (crs node)
  (check-role-strength (get-node node)))

; Returns  the strength of RULE
```

```
(define (rule-strength rule)
 (cond
   ((production? rule)(production-strength rule))
   ((node? rule)  (production-strength (node-english rule)))
   (t (msg t "Do not know how to find the strength of " rule))))

; Does a spreading activation by first
; activating facts on the list FACTS
; as ANTECS and the rule pointed to by *PLAN-USED* as the only
; CONSEQS.
(define (prop-rules facts )
 (check-propagate-r 'rule
          (cons 'antecs facts)
        (list 'conseqs (link-node *link-made*))
        'inhib-link 'ruleness 'classify? )
 (r-a))


; returns a list of facts which are antecedents for
; RULE but were not indicated by the learner
(define (why-not rule)
  (let ((reasons nil))
   (walk (lambda (pr)
    (and
      (p-is-a? (car pr) (get-node 'fact))
      (walk (lambda (conn)
       (and
         (memq? 'parent conn)
         (memq? 'role conn)
         (push reasons (car conn) ) ))
          (node-excite (car pr)))
   ))
      (map-subset not-indicated
                              (node-roles (get-node rule)))
   )
 reasons ) )
```

```
; RETURNS LIST OF FACTS WHICH WERE INDICATED BY THE LEARNER
; AND ARE ANTECEDENTS OF RULE, BUT ARE NOT TRUE.
(define (why-wrong rule)
  (let ((reasons nil))
    (map (lambda (theRole)
     (and *debug* (msg t "checking role: " theRole))
     (if (indicated theRole )
      (block
        (and *debug* (msg t "it was indicated"))
        (map (lambda (conn)
         (and
           (memq? 'parent conn)
           (memq? 'role conn)
           (set a (car conn))
           (not (truth  (node-name (car conn))))
           (push reasons (car conn) )
         )
        ) (node-excite (car theRole)))
     )
      nil)
     ) (node-roles (get-node  rule)))
   reasons
))


;  returns T if FACT is both true and was indicated by the learner
(define (true-indicated fact)
  (and (indicated fact)(true fact)))


;  returns T if ROLE received some activation from the facts that
;  the learner indicated.
(define (indicated role)
  (let ((node-act (node-activation (if (list? role)
                           (car role)
                           (get-node role )       ) )))
    (and *debug* (msg t "node act is: " node-act))
    (and
     (eq? (caar  node-act) 'b)
     (number? (cadar node-act))
     (> (cadar node-act) 2)
   )))


(define (not-indicated role)
   (not (indicated role)))

(define (r-a)
  (let ((rules (sort (node-children %rule) stronger?)))
  (walk (lambda (node)(msg t  "str: "
                 (stronger? node)
             "   rule: "  (car node))
        )
     rules)
      (map (lambda (x) (list (car x) (stronger? x) ))rules)
   ))

(define (f-a)
  (let ((facts (node-children %fact)))
    (walk (lambda(x) (aa (car x))) facts)))


(define (stronger?  a )
 (let ((str
  (activ-str (car(node-activation (car a)))))
      )
  (if (number? str)
    str
    0)))


;****************************************************
```

```
(map! 'conseqs 'r-part)
(is-a! (indiv-role! 'part-link 'make-part 'r-part  ) 'conseqs)
(is-a! (indiv-role! 'part-t-has-f 'r-t-has-f 'r-part  'o .6) 'antecs)
(is-a! (indiv-role! 'part-f-part-t 'r-f-part-t 'r-part 'o .4) 'antecs)
(+inhib-link!  'r-part 'ruleness)


(set (node-english (type! 'r-maybe-part 'rule) ) (get-production 'maybe-part))
(+inhib-link! 'r-maybe-part  'ruleness)
(map!  'antecs 'r-maybe-part)
(map! 'conseqs 'r-maybe-part)
(is-a! (indiv-role! 'maybe-part-link 'make-part 'r-maybe-part  ) 'conseqs)
(is-a! (indiv-role! 'maybe-part-t-has-f 'r-t-has-f 'r-maybe-part  'o .9) 'antecs)


(set (node-english (type! 'r-prop 'rule) ) (get-production 'prop))
(+inhib-link! 'r-prop  'ruleness)
(map!  'antecs 'r-prop)
(map! 'conseqs 'r-prop)
(is-a! (indiv-role! 'prop-link 'make-prop 'r-prop  ) 'conseqs)
(is-a! (indiv-role! 'prop-t-has-f 'r-t-has-f 'r-prop 'o .9 ) 'antecs)


(set (node-english (type! 'r-is-one 'rule) ) (get-production 'is-one))
(+inhib-link! 'r-is-one  'ruleness)
(map!  'antecs 'r-is-one)
(map! 'conseqs 'r-is-one)
(is-a! (indiv-role! 'is-one-link 'make-prop 'r-is-one  ) 'conseqs)
(is-a! (indiv-role! 'is-one-t-is-f 'r-t-is-f 'r-is-one 'o .9 ) 'antecs)


(set (node-english (type! 'r-leads 'rule)  ) (get-production 'leads))
(+inhib-link! 'r-leads  'ruleness)
(map!  'antecs 'r-leads)
(map! 'conseqs 'r-leads)
(is-a! (indiv-role! 'leads-link 'make-leads 'r-leads  ) 'conseqs)
(is-a! (indiv-role! 'leads-f-cond-t 'r-f-cond-t 'r-leads  'o .9) 'antecs)
```

```
(set (node-english (type! 'r-prop-leads 'rule) ) (get-production 'prop-leads))
(+inhib-link! 'r-prop-leads  'ruleness)
(map!  'antecs 'r-prop-leads)
(map! 'conseqs 'r-prop-leads)
(is-a! (indiv-role! 'prop-leads-link 'make-leads 'r-prop-leads  ) 'conseqs)
(is-a! (indiv-role! 'prop-leads-t-has-f 'r-t-has-f 'r-prop-leads  'o .9) 'antecs)


(set (node-english (type! 'r-expect 'rule)  ) (get-production 'expect))
(+inhib-link! 'r-expect  'ruleness)
(map!  'antecs 'r-expect)
(map! 'conseqs 'r-expect)
(is-a! (indiv-role! 'expect-link 'make-leads 'r-expect  ) 'conseqs)
(is-a! (indiv-role! 'expect-t-expect-f 'r-t-expect-f 'r-expect 'o .9) 'antecs)


(set (node-english (type! 'r-results 'rule)  ) (get-production 'results))
(+inhib-link! 'r-results  'ruleness)
(map!  'antecs 'r-results)
(map! 'conseqs 'r-results)
(is-a! (indiv-role! 'results-link 'make-leads 'r-results  ) 'conseqs)
(is-a! (indiv-role! 'results-t-result-f 'r-t-result-f 'r-results 'o .9) 'antecs)
```

```
;***************************************************
;**                                              **
;**      DEFINITIONS FOR SEMANTIC NET            **
;**       REPRESENTATION OF RULES                **
;**                                              **
;***************************************************
;***************************************************


(type! 'rule nil )
(type-role! 'antecs 'fact 'rule)
(type-role! 'conseqs 'act 'rule)

(inhib-link! 'ruleness 'rule)

; FACTS
;  ******

(type! 'r-f-maybe-t 'fact)

(type! 'r-f-is-t 'r-f-maybe-t 'up .8)
(is-a! 'r-f-is-t 'fact)

(type! 'r-t-is-f 'fact)
(type! 'r-t-unlik-f 'fact)
(type! 'r-f-unlik-t 'fact)
(type! 'r-f-not-t 'fact)
(type! 'r-t-not-f 'fact)
(type! 'r-eq-ch  'fact)
(type! 'r-same-conc 'fact)
(type! 'r-t-has-f   'fact)
(type! 'r-f-thing   'fact)
(type! 'r-f-part-t 'r-f-has-t)
(is-a! 'r-f-part-t 'fact)
(type! 'r-t-cond-act  'fact)
(type! 'r-f-cond-t    'fact)
(type! 'r-t-fact 'fact)
(type! 'r-t-result-f 'fact)
(type! 'r-t-expect-f 'fact)

(is-a! 'r-t-not-f 'r-t-unlik-f 'up .8)
(is-a! 'r-f-not-t 'r-f-unlik-t 'up .8)
(distinct! 'r-f-is-t 'r-f-not-t)
(distinct! 'r-t-is-f 'r-t-not-f)

; CONSEQUENTS
;  ***********

(type! 'make-link 'act)
(inhib-link! 'linkness 'make-link)

(type! 'make-part 'make-link)
(+inhib-link! 'make-part 'linkness)

(type! 'make-is-a 'make-link)
(+inhib-link! 'make-is-a 'linkness)

(type! 'make-leads 'make-link)
(+inhib-link! 'make-leads 'linkness)

(type! 'make-equiv 'make-link)
(+inhib-link! 'make-equiv 'linkness)

(type! 'make-prop 'make-link)
(+inhib-link! 'make-prop 'linkness)

(type! 'make-not 'make-link)
(+inhib-link! 'make-not 'linkness)


; RULES
;******

; the NODE-ENGLISH slot of a rule node is given a pointer to the
; production version of the rule.

(set (node-english (type! 'r-half-is-a 'rule )) (get-production 'half-is-a))
(+inhib-link!  'r-half-is-a 'ruleness)
(map!  'antecs 'r-half-is-a)
(map! 'conseqs 'r-half-is-a)
(is-a! (indiv-role! 'half-is-a-f-is-t 'r-f-is-t 'r-half-is-a 'o .7) 'antecs)
(is-a! (indiv-role! 'half-is-a-link 'make-is-a 'r-half-is-a) 'conseqs)


(set (node-english (type! 'r-is-a 'rule) ) (get-production 'is-a))
(+inhib-link! 'r-is-a  'ruleness)
(map!  'antecs 'r-is-a)
(map! 'conseqs 'r-is-a)
(is-a! (indiv-role! 'is-a-f-is-t 'r-f-is-t 'r-is-a 'o .6) 'antecs)
(is-a! (indiv-role! 'is-a-t-unlik-f 'r-t-unlik-f 'r-is-a 'o .4) 'antecs)
(is-a! (indiv-role! 'is-a-link 'make-is-a 'r-is-a ) 'conseqs)


(set (node-english (type! 'r-weak-is-a 'rule)) (get-production 'weak-is-a))
(+inhib-link! 'r-weak-is-a  'ruleness)
(map!  'antecs 'r-weak-is-a)
(map! 'conseqs 'r-weak-is-a)
(is-a! (indiv-role! 'weak-is-a-link 'make-is-a 'r-weak-is-a  ) 'conseqs)
(is-a! (indiv-role! 'weak-is-a-f-maybe-t 'r-f-maybe-t 'r-weak-is-a 'o .4) 'antecs)
(is-a! (indiv-role! 'weak-is-a-t-unlik-f 'r-t-unlik-f 'r-weak-is-a 'o .6) 'antecs)


(set (node-english (type! 'r-not 'rule)  ) (get-production 'not))
(+inhib-link!  'r-not 'ruleness)
(map!  'antecs 'r-not)
(map! 'conseqs 'r-not)
(is-a! (indiv-role! 'not-link 'make-not 'r-not   ) 'conseqs)
(is-a! (indiv-role! 'not-f-not-t 'r-f-not-t 'r-not 'o .5) 'antecs)
(is-a! (indiv-role! 'not-t-not-f 'r-t-not-f 'r-not 'o .5) 'antecs)
```

```
(set (node-english (type! 'r-rev-is-a 'rule)  ) (get-production 'rev-is-a))
(+inhib-link!  'r-rev-is-a 'ruleness)
(map!  'antecs 'r-rev-is-a)
(map! 'conseqs 'r-rev-is-a)
(is-a! (indiv-role! 'rev-is-a-link 'make-is-a 'r-rev-is-a   ) 'conseqs)
(is-a! (indiv-role! 'rev-t-is-f 'r-t-is-f 'r-rev-is-a  'o .7) 'antecs)
(is-a! (indiv-role! 'rev-f-unlik-f 'r-f-unlik-t 'r-rev-is-a  'o .3) 'antecs)


(set (node-english (type! 'r-dub-is-a 'rule)  ) (get-production 'dub-is-a))
(+inhib-link!  'r-dub-is-a 'ruleness)
(map!  'antecs 'r-dub-is-a)
(map! 'conseqs 'r-dub-is-a)
(is-a! (indiv-role! 'dub-is-a-link 'make-equiv 'r-dub-is-a  ) 'conseqs)
(is-a! (indiv-role! 'dub-is-a-f-is-t 'r-f-is-t 'r-dub-is-a  'o .4 ) 'antecs)
(is-a! (indiv-role! 'dub-is-a-t-is-f 'r-t-is-f 'r-dub-is-a  'o .4 ) 'antecs)

(set (node-english (type! 'r-equal-high 'rule)  ) (get-production 'equal-high))
(+inhib-link!  'r-equal-high 'ruleness)
(map!  'antecs 'r-equal-high)
(map! 'conseqs 'r-equal-high)
(is-a! (indiv-role! 'equal-high-link 'make-equiv 'r-equal-high  ) 'conseqs)
(is-a! (indiv-role! 'equal-high-f-is-t 'r-f-is-t 'r-equal-high  'o .4 ) 'antecs)
(is-a! (indiv-role! 'equal-high-t-is-f 'r-t-is-f 'r-equal-high  'o .4 ) 'antecs)
(is-a! (indiv-role! 'equal-high-eq-ch  'r-eq-ch  'r-equal-high  'o .2 ) 'antecs)


(set (node-english (type! 'r-same 'rule )  ) (get-production 'same))
(+inhib-link!  'r-same 'ruleness)
(map!  'antecs 'r-same)
(map! 'conseqs 'r-same)
(is-a! (indiv-role! 'same-link 'make-equiv 'r-same  ) 'conseqs)
(is-a! (indiv-role! 'same-same-conc 'r-same-conc 'r-same 'o .9 ) 'antecs)

(set (node-english (type! 'r-part 'rule)   ) (get-production 'part))
(map!  'antecs 'r-part)
```