

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

HIGH-LEVEL INFERENCE IN A LOCALIST NETWORK

Trent Elliot Lange

**December 1990
CSD-900053**

**High-Level Inferencing in
a Localist Network**

Trent Eliot Lange

December 1990

Technical Report UCLA-AI-90-11

UNIVERSITY OF CALIFORNIA

Los Angeles

High-Level Inferencing In A Localist Network

A thesis submitted in partial satisfaction of the
requirements for the degree Master of Science
in Computer Science

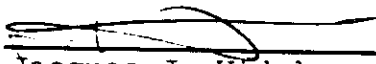
by

Trent Eliot Lange

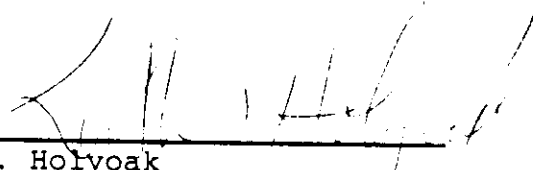
1990

© Copyright by
Trent Eliot Lange
1990


The thesis of Trent Eliot Lange is approved.



Jacques J. Vidal



Keith J. Holyoak



Michael G. Dyer, Committee Chair

University of California, Los Angeles

1990

TABLE OF CONTENTS

Table of Contents	v
List of Figures	vii
List of Tables	viii
Acknowledgements	ix
1. Introduction.....	1
1.1. High-Level Inferencing	1
1.2. Previous Approaches.....	3
1.2.1. Distributed Connectionist Networks.....	4
1.2.2. Structured Connectionist Networks	5
1.2.3. Marker-Passing Networks.....	6
1.3. Overview of the Thesis.....	8
2. Overview of Robin's Knowledge	9
2.1. Selectional Restrictions on Roles	9
2.2. Rules as Relations Between Frames.....	9
2.3. Connection Weights	10
2.4. Overall Knowledge Bases	11
3. Robin.....	13
3.1. Role-Binding and Inferencing With Signatures.....	13
3.1.1. Variable Binding With Signatures	13
3.1.2. Structure of the Network.....	13
3.1.3. Activation Functions.....	14
3.1.3.1. Activation of Concept Nodes	14
3.1.3.2. Activation of Binding Nodes	15
3.1.4. Propagation of Signatures for Inferencing.....	15
3.1.5. Discussion of Signatures.....	17
3.2. Frame Selection With Evidential Activation	18
3.2.1. Selection by the Evidential Semantic Network.....	19
3.2.2. Selection of Ambiguous Role-Bindings.....	19
3.2.3. Structure of the Evidential Network	19
3.2.4. Activation Control	20
3.2.5. Disambiguation Example.....	22
3.2.6. Evidential vs Signature Activation	24

3.3.	Selectional Restrictions.....	25
3.3.1.	Overview of the Frame Selection Process	26
3.3.2.	Gating of Signatures and Evidential Activation	28
3.3.3.	Binding Constraint Structure	29
3.3.4.	Virtual Evidential Structure From Signature Bindings	31
4.	Detailed Examples	33
4.1.	Processing of Hiding Pot.....	33
4.2.	The Effect of Selectional Restrictions	38
5.	Discussion and Conclusions.....	41
5.1.	Future Work.....	41
5.1.1.	Signatures Using Distributed Representations	41
5.1.2.	Signatures as Temporal Frequencies	42
5.1.3.	Embedded Role-Bindings.....	43
5.1.4.	Network Capacity.....	43
5.1.5.	Formation of Long-Term Episodic Memory.....	44
5.1.6.	Lexical Information.....	44
5.2.	Comparison to Related Connectionist Models of Variable Binding and Inferencing.....	44
5.2.1.	Distributed Connectionist Models.....	44
5.2.2.	Ajjanagadde and Shastri's Structured Connectionist Model	45
5.3.	Comparison to Related Connectionist Models of Disambiguation and Reinterpretation.....	48
5.3.1.	Structured Spreading-Activation Models.....	48
5.3.2.	Marker-Passing Models.....	50
5.4.	Conclusions	53
	References.....	54
	Appendix A.....	57

LIST OF FIGURES

Figure 1. Simple structured spreading-activation network	6
Figure 2. Disambiguation in a simple structured spreading-activation network.....	6
Figure 3. Simple marker-passing network	7
Figure 4. Definition of frames in the knowledge base	10
Figure 5. Overview of ROBIN knowledge base	12
Figure 6. Signatures.....	14
Figure 7. Parallel paths of evidential and signature activation.....	16
Figure 8. Propagation of signatures for inferencing.....	17
Figure 9. Propagation of signatures for inferencing.....	18
Figure 10. Detailed structure of the evidential layer.....	21
Figure 11. Global Inhibition Node.....	22
Figure 12. Overview of network activations and bindings after P1.....	23
Figure 13. Evidential activations of meanings of “pot” after P1.....	24
Figure 15. Example of the frame selection problem	27
Figure 16. Gating of evidential and signature binding paths.....	28
Figure 17. Calculation of gating enforcing selectional restrictions.....	30
Figure 18. Evidential activations during processing of Hiding Pot	34
Figure 19. Net inputs of ambiguous frames during processing of Hiding Pot	35
Figure 20. Overview of network activations and bindings after Hiding Pot	37
Figure 21. Evidential activations after Cook-and-Clean example	38
Figure 22. Distributed signatures	43
Figure 23. Inferencing with a phase-clock [Ajjanagadde & Shastri, 1989].....	46
Figure 24. Example of crosstalk in normal structured networks	49
Figure 25. Example of ROBIN eliminating crosstalk	50
Figure 26. Example of ROBIN performing marker-passing task.....	51

LIST OF TABLES

Table 1.	Inferences needed for Hiding Pot	2
Table 2.	Inputs ROBIN handles	42
Table 3.	Comparison of marker-passing heuristics and ROBIN's emergent properties..	52

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor, Michael Dyer, for his continuous support and for many enthusiastic discussions and pieces of advice on this research and thesis. I would also like to thank my committee members: Jacques Vidal has made helpful comments on previous drafts of this work, and has been especially valuable in our collaboration on related inferencing research. Keith Holyoak has also given helpful criticisms on this research, and has helped me look into broadening its scope by exploring how it applies to other problems in cognitive psychology. This research has been supported in part by the ITA and W.M. Keck Foundations.

ABSTRACT OF THE THESIS

High-Level Inferencing in a Localist Network

by

Trent Eliot Lange

Master of Science in Computer Science

University of California, Los Angeles, 1990

Professor Michael G. Dyer, Chair

Connectionist models have had problems representing and applying general knowledge rules that specifically require variables. This *variable binding problem* has barred them from performing the high-level inferencing necessary for planning, reasoning, and natural language understanding. This thesis describes ROBIN, a local connectionist model capable of high-level inferencing requiring variable bindings and rule application. Variable bindings are handled by *signatures* - activation patterns which uniquely identify the concept bound to a role. Signatures allow multiple role-bindings to be propagated across the network in parallel for rule application and dynamic inference path instantiation. Signatures are integrated within a connectionist semantic network structure whose constraint-relaxation process selects between those newly-instantiated inferences. This allows ROBIN to handle an area of high-level inferencing difficult even for symbolic models, that of resolving multiple constraints from context to select the best interpretation from among several alternative and possibly ambiguous inference paths.

1. INTRODUCTION

Critical to cognitive abilities such as natural language understanding and planning is the need to perform *high-level inferencing* to make explanations and predictions from what is known about the world. Connectionist models have been unable to perform high-level inferencing because of their difficulties with representing and applying general knowledge rules. They have so far been unable to solve this *variable binding problem*, i.e. the ability to maintain multiple variable bindings and modify them by rule application. It has recently been argued that these deficits strictly limit the usefulness of connectionist networks for modelling high-level cognitive tasks [Fodor & Pylyshyn, 1988].

This thesis describes a structured connectionist model capable of variable binding and rule application. ROBIN (ROle Binding and Inferencing Network) [Lange & Dyer, 1988, 1989a,b] performs high-level inferencing over structured connections of nodes that encode world knowledge in semantic networks similar to those of other models. However, ROBIN has additional node-pathway structure to handle variables and dynamic role-binding. With this structure, the model is able to maintain multiple role-bindings and propagate them along paths defined by the knowledge base's general knowledge rules, thus performing inferencing.

Although the ability to maintain variables and apply general knowledge rules is necessary for high-level inferencing, it alone is not sufficient. This is because one of the most difficult parts of the high-level inferencing problem is that of *selecting* the best interpretation from among multiple alternative and potentially ambiguous inference paths. The connectionist semantic network within which ROBIN's variable binding network structure is integrated allows a solution to this disambiguation problem, since its smooth constraint satisfaction process allows automatic selection of the most-highly activated path as the network's interpretation.

1.1. High-Level Inferencing

High-level inferencing is the ability to use previous knowledge and rules about the world to build new beliefs about what is true. In natural language understanding, for example, a reader must often make multiple inferences to understand the motives of actors and to causally connect actions that are unrelated on the basis of surface semantics alone. Complicating the inference process is the fact that language is often ambiguous on both the lexical and conceptual levels. Consider the phrase:

P1: *"John put the pot inside the dishwasher"*

Most people will conclude that John transferred a Cooking-Pot inside of a dishwasher in an attempt to get it clean. This conclusion is an example of a high-level inference. However, suppose P1 is followed by:

P2: *"because the police were coming."*

Suddenly, the interpretation selected for the word "*pot*" in P1 changes to Marijuana, and John's Transfer-Inside action becomes a plan for hiding the Marijuana from the police. This reinterpretation requires the inferences shown in Table 1 to understand the most probable *causal relationship* between the actions of phrase P1 and P2 (collectively called the **Hiding Pot** episode).

To understand episodes such as **Hiding Pot**, a system must minimally be able to dynamically make such chains of inferences (by applying general knowledge rules) and temporarily maintain them (with a variable-binding mechanism). For example, a system must know about the general concept (or *frame*) of an actor transferring himself to a location ("coming"). To represent the initial knowledge given by phrase P2 of **Hiding Pot**, the system must be able to temporarily maintain a

I1: If the police see John's marijuana, then they will know that he possesses an illegal object (since marijuana is an illegal substance).
I2: If the police know that John is in possession of an illegal object, then they will arrest him, since possessing an illegal object is a crime.
I3: John does not want to get arrested.
I4: John has the goal of stopping the police from seeing his marijuana.
I5: The police coming results in them being in the proximity of John and his marijuana.
I6: The police being in the proximity of John's marijuana enables them to see it.
I7: John's putting the marijuana inside the dishwasher results in the marijuana being inside the dishwasher.
I8: The marijuana is inside an opaque object (the dishwasher).
I9: Since the marijuana is inside an opaque object, the police cannot see it, thus satisfying John's goal.

Table 1: Inferences needed to understand the sentence "John put the pot inside the dishwasher because the police were coming." (**Hiding Pot**)

particular *instantiation* of this Transfer-Self frame in which the Actor role (a variable) is bound to Police and the Location role is bound to the location of John. The system must also have the general knowledge that when an actor transfers himself to a location, he ends up in the proximity of that location, which might be represented as the rule:

```
R1: [Actor X Transfer-Self Location Y]
    == results-in ==> [Actor X Proximity-Of Object. Y]
```

Applying this rule to the instantiation of the police Transfer-Self would allow the system to make inference I5 in Table 1, that the police will be in the proximity of John and his marijuana. Another piece of knowledge that the system must have is that an actor must be in the proximity of an object in order to see it, which might be represented as the rule:

```
R2: [Actor X Proximity-Of Object Y]
    == precondition-for ==> [Actor X See-Object Object Y]
```

If this rule is applied to the new piece of knowledge that the Police will be in the proximity of John, then the system would be able to infer that there is the potential for them to see John and his marijuana (I6). The rest of the inferences in Table 1 to understand **Hiding Pot** are the result of the application of similar rules and knowledge about the world.

Unfortunately, even the ability to maintain variable bindings and apply general knowledge rules of the above sort is often insufficient for language understanding and other high-level cognitive tasks. This is because language is often ambiguous, as **Hiding Pot** illustrates, with several possible interpretations that must be chosen between. One of the fundamental problems in high-level inferencing is thus that of *frame selection*. When should a system make inferences from a given frame instantiation? And when conflicting rules apply to a given frame instantiation, which should be selected? Only a system that can handle these problems will be able to address the following critical tasks:

Word-Sense Disambiguation: Choosing the meaning of a word in a given piece of text. In **P1**, the word “*pot*” refers to a Cooking-Pot, but when **P2** is presented, the evidence is that the interpretation should change to Marijuana.

Inferencing: Applying causal knowledge to understand the results of actions and the motives of actors. There is nothing in **Hiding Pot** that explicitly states that the police might see the pot (I6), or even that the police will be in proximity of it and John (I5). Nor is it explicitly stated what the police will do if they see he possesses Marijuana (I1, I2). Each of these assumptions must be inferred from phrases **P1** and **P2**.

Concept Refinement: Instantiating an applicable specific frame from a more general one. In **P1**, the fact that the pot was inside a dishwasher tells us more than the simple knowledge that it was inside a container. In contrast, the salient point in **Hiding Pot** is that it is inside of an opaque object (I8), which allows us to infer that the police will not be able to see it (I9).

Plan/Goal Analysis: Recognizing the plan an actor is using to fulfill his goals. In **P1**, it appears that John put the pot into the dishwasher as part of the \$Dishwasher-Cleaning script to satisfy his goal of getting it clean. In **Hiding Pot**, however, it appears that it is part of his plan to satisfy his sub-goal of hiding it from the police (I4), which is part of his overall goal to avoid arrest (I3).

Frame selection is complicated by the effect of additional context, which often causes *reinterpretation* to competing frames. The contextual evidence in **Hiding Pot** can conflict even more, and the explanation change again, if, for example, the next phrase is:

P3: “*They were coming over for dinner.*”

As a result of **P3**, the word “*pot*” might be reinterpreted back to Cooking-Pot. These examples clearly point out two sub-problems of frame selection, those of *frame commitment* and *reinterpretation*. When should a system commit to one interpretation over another? And if it does commit to one interpretation, how does new context cause that interpretation to change?

1.2. Previous Approaches

Symbolic artificial intelligence (AI) systems have so far been the only types of models capable of performing high-level inferencing. A good example is BORIS [Dyer, 1983], a natural language understanding program for modelling in-depth understanding of relatively long and complex stories. BORIS had a symbolic knowledge base containing knowledge structures representing various actions, plans, goals, emotional affects, and methods for avoiding planning failures. When reading in a story, BORIS would fire rules from its knowledge base to perform inferencing and form an internal representation of the story, about which it could then answer questions. Other models that have successfully approached complex parts of the language understanding process have all had similar types of knowledge representation and rule-firing capabilities.

Connectionist networks, however, have significant potential advantages over traditional symbolic approaches to the interpretation process. Their conceptual knowledge is stored entirely in an interconnected network of simple nodes whose activations are calculated based on their previous activation and that of the nodes to which they are connected. As a result, a major portion of the understanding process is controlled by a simple spreading-activation mechanism, instead of by large collections of brittle and sometimes ad-hoc rules.

1.2.1. Distributed Connectionist Networks

Distributed connectionist models have had a great deal of success modelling low-level natural language understanding tasks, especially those requiring similarity-based learning. A number of researchers have argued that this new subsymbolic paradigm will completely subsume the symbolic paradigm, as the explicit rules used in symbolic models are replaced by the more robust interactions of distributed representations and the connection weights learned from experience [Rumelhart & McClelland, 1986]. Although some of the severest criticisms of this stand ([Fodor & Pylyshyn, 1988], [Pinker & Prince, 1988]) have been partially rebutted by recent models showing that distributed models can represent some variable bindings and constituent structure, current distributed models are still quite limited in comparison to symbolic models in their abilities to perform high-level processing such as natural language understanding.

A good example of how distributed connectionist models have been used to approach language understanding is provided by the case-role assignment model of McClelland & Kawamoto [1986]. The main task of their model is to learn to assign the proper semantic case roles for sentences. For example, given the syntactic surface form of the sentence "*the boy broke the window*", their network is trained to place the semantic microfeature representation of **Boy** in the units representing the Agent role on the output layer, whereas given "*the rock broke the window*", it is trained to place the representation of **Rock** in the Instrument role. Their network is also trained to perform lexical disambiguation, e.g. mapping the pattern for the word "*bat*" to a **Baseball-Bat** for sentences such as "*the boy hit the ball with the bat*", and to a **Flying-Bat** for sentences such as "*the bat flew*". Once the input/output pairs have been learned, the network exhibits a certain amount of generalization by mapping the case roles and performing lexical disambiguation for novel inputs similar to the training sentences.

One of the main limitations of McClelland & Kawamoto's model for language understanding is that its output can only handle direct, one-step mappings from the input to the output, thus limiting it to sentences that can be understood and disambiguated based upon the surface semantics of the input. Two distributed connectionist models that get around this limitation are the models of Miikkulainen & Dyer [1989] and St. John [1990]. Both models use *recurrent networks* with a hidden layer of units whose activation pattern essentially stores the state (or "gestalt") of the stories being understood. This allows them to learn to process more complex language based on scripts (such as going to a restaurant) and other script-like stories [Schank & Abelson, 1977]. Both models have the lexical disambiguation abilities of McClelland & Kawamoto's model, but, more importantly, are able to infer unmentioned story events and role-fillers from the script that has been "recognized" by the hidden layer.

Unfortunately, there may be significant problems in scaling such *pattern-transformation* distributed connectionist models to handle more complex language. Both Miikkulainen & Dyer and St. John's models work by resolving constraints from input context to recognize one of their trained scripts and instantiate it with the bindings of the particular input story. However, much of language understanding involves the inference of causal relationships between events for completely novel stories in which no script or previously-trained input/output pair can be recognized. This requires *dynamic inferencing* — a process of constructing chains of inferences over simple known rules, with each inference resulting in a potentially novel intermediate state [Touretzky, 1990]. It remains to be seen whether a single blended activation pattern on the bank of hidden units in recurrent networks can simultaneously hold and make dynamic inferences from multiple, never-before encountered interpretation chains.

Other distributed models explicitly encode variables and rules, such as the models of Touretzky & Hinton [1988] and Dolan & Smolensky [1989]. Because of this, such *rule-implementing* distributed models are able to perform some of the dynamic inferencing necessary for language understanding. Unfortunately, however, the types of rules they can currently encode are generally limited. More importantly, they are serial at the knowledge level because they can fire only one

rule at a time. This is a serious drawback for natural language understanding, particularly for ambiguous text, in which the combinatorially explosive number of multiple alternative interpretations often requires that the inference paths be explored in parallel [Lange, in press].

1.2.2. Structured Connectionist Networks

Structured connectionist models represent knowledge in semantic networks in which concepts are represented by individual nodes and relations between concepts are encoded by weighted connections between nodes. The numeric activation level on each conceptual node generally represents the amount of *evidence* available for its concept in a given context. Because knowledge is spread across the network (as opposed to the concentration of knowledge in the weights between the single input and output layer of most distributed models), structured models have the potential to pursue multiple candidate interpretations of a story in parallel as each interpretation is represented by activation in different local areas of the network. This makes them ideally suited to the disambiguation portion of the language understanding process, because it is achieved automatically as related concepts under consideration provide graded activation evidence and feedback to one another in a form of analog constraint relaxation.

As an example of how structured connectionist models process language and perform disambiguation, consider the sentence:

"The astronomer married the star." (Star-Marriage)

The word "*star*" could be easily disambiguated to **Movie-Star** by a symbolic rule-based system having selectional restrictions (even astronomers cannot marry celestial bodies, except perhaps metaphorically). However, many readers report this and similar sentences as "cognitive double-takes" because "*astronomer*" initially primes the **Celestial-Body** interpretation. Figure 1 shows an extended version of the semantic portion of the structured network Waltz & Pollack [1985] built to process **Star-Marriage** and illustrate this effect. After the input nodes for **Star-Marriage** are clamped to a high level of activation, the **Celestial-Body** interpretation of "*star*" initially acquires more activation than the **Movie-Star** interpretation because of priming from **Astronomer** through **Astronomy** (Figure 2). However, **Movie-Star** eventually wins out because activation feedback over the semantic connections from the **Marry** node to **Movie-Star** outweighs that spreading from the **Astronomer** node to **Celestial-Body**.

Unfortunately, the applicability of structured connectionist models to natural language understanding has been severely hampered because of their difficulties representing dynamic role-bindings and performing inferencing¹. Their lack of variable binding abilities leaves them prone to crosstalk even for simple sentences. For example, the network of Figure 1 has no way to distinguish between the sentences "*The astronomer saw the star*" and "*The star saw the astronomer*", despite the crucial difference that the role-bindings make in their interpretation. More importantly, without a mechanism to represent such dynamic bindings, they cannot propagate them to make the chains of inferences necessary for understanding more complex language. This has so far stopped them from going beyond simple language processing that can be resolved solely based on the surface semantics of the input.

¹Ajjanagadde & Shastri [1989], Barnden [1990], and Holldobler [1990] describe structured models that can perform some variable-binding and inferencing, but which do not have the disambiguation abilities of normal structured spreading-activation models.

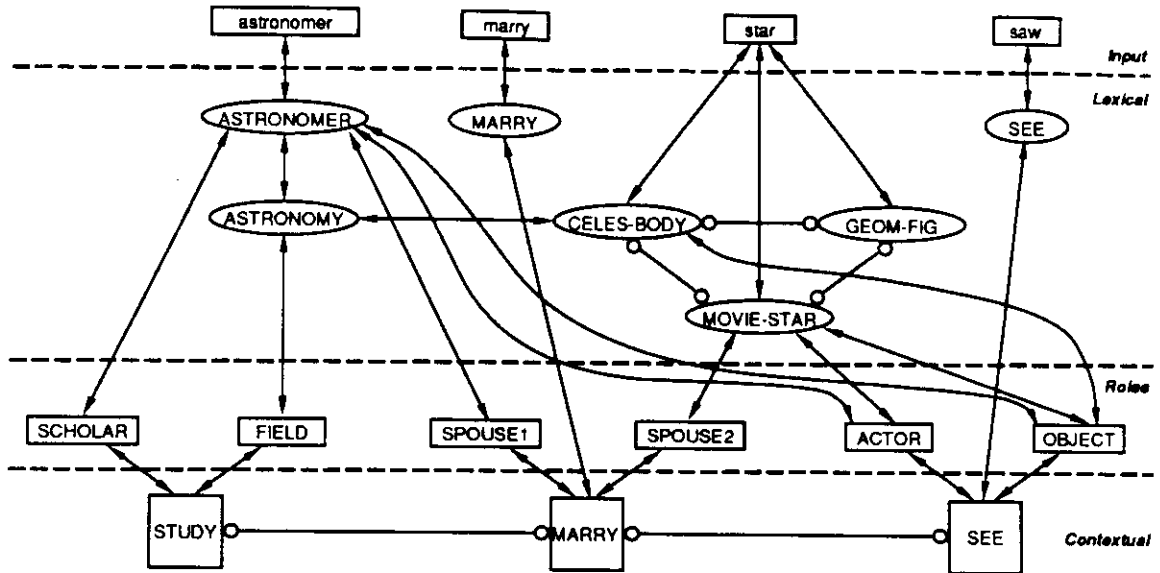


Figure 1. Structured spreading-activation network based on [Waltz & Pollack, 1985]. Lines with arrows are excitatory connections; lines with open circles are inhibitory.

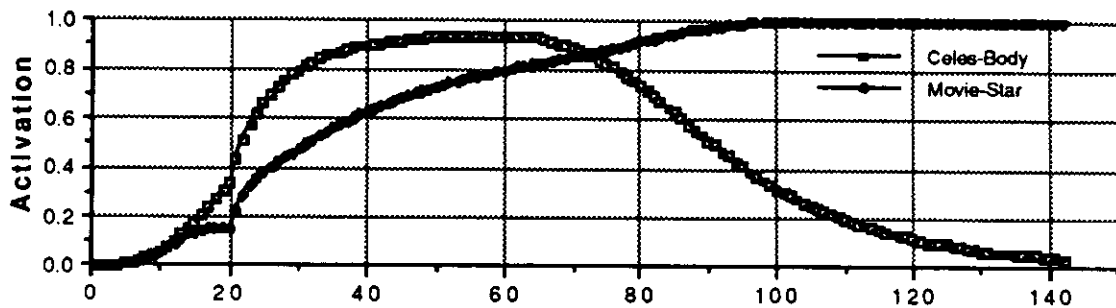


Figure 2. Activations of meaning of word "star" after "astronomer married star" is clamped for network in Figure 1.

1.2.3. Marker-Passing Networks

Marker-passing models operate by spreading symbolic markers in parallel across labelled semantic networks similar to those of structured connectionist networks. Interpretation of the input is achieved when propagation of markers finds a path of nodes connecting words and concepts from the input text. Because of the symbolic information held in their markers and networks, they are able to represent dynamic role-bindings, and so have been able to perform high-level inferencing for natural language understanding (cf. [Charniak, 1986], [Riesbeck & Martin, 1986], [Granger *et al.*, 1986], [Eiselt, 1987], and [Norvig, 1989]).

As an example of how marker-passing networks process language and perform disambiguation, consider the following text (from [Eiselt, 1987]):

"Fred asked Wilma to marry him. Wilma began to cry." (Marriage)

Interpreting this text requires that a causal relationship be inferred between Fred's proposal and Wilma's crying. One possible reason for her crying was that she was happy about his proposal

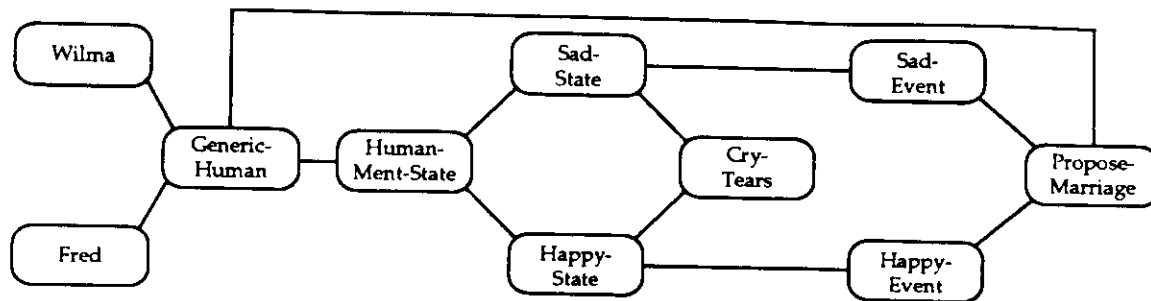


Figure 3. Marker-passing network from [Eiselt, 1987].

and crying “tears of joy”. To understand this sentence and resolve the ambiguity, ATLAST [Eiselt, 1987] uses the network shown in Figure 3 by passing markers starting from the nodes for Cry-Tears and Propose-Marriage. This propagation of markers finds the path Cry-Tears \leftrightarrow Happy-State \leftrightarrow Happy-Event \leftrightarrow Propose-Marriage, returning the “tears of joy” interpretation. Besides finding the inference path representing the interpretation of the story, the symbolic pointers held in the markers also keep track of the role-bindings, so that the model can clearly resolve that it was Fred who did the Propose-Marriage and Wilma who did the Cry-Tears, and not the other way around.

Much text, of course, is ambiguous, and **Marriage** is no exception. Another possible reason that Mary began to cry was that she was saddened or upset by Fred’s proposal. The same propagation of markers that found the above “tears of joy” path will therefore find a second path, Cry-Tears \leftrightarrow Sad-State \leftrightarrow Sad-Event \leftrightarrow Propose-Marriage. To resolve such ambiguities, marker-passing systems generally use a serial heuristic path evaluator separate from the marker-passing process to select the most relevant path from the many paths generated. Such path evaluators usually include rules that select shorter paths over longer ones, reject paths that do not include as much of the input as competing ones, and so forth. For example, to disambiguate between the “tears of joy” and “saddened” paths, ATLAST applies an evaluation metric between two competing paths of equal length that selects the oldest path. The Happy-State path was discovered first, and thus remains as the interpretation of the input.

As their use of heuristic path-evaluators indicate, marker-passing systems generally permit themselves the luxury of using traditional symbolic buffers and programs to complement the spreading-activation process of the network. This allows them to build up complex symbolic representations of stories outside the network (as done by Norvig [1989]) or hold rejected inference paths to allow reinterpretation if a path is rediscovered (as done by ATLAST when **Marriage** is followed by “*Wilma was saddened by the proposal.*”).

The best feature of marker-passing systems is that their parallel instantiation of inference paths makes them extremely efficient at generating different possible interpretations of the input. Unfortunately, the bottleneck for marker-passing systems is the separate path evaluation mechanisms used to select between generated interpretations (the heart of the disambiguation problem). The main problem is the extremely large number of *spurious* (i.e. non-important or logically-impossible) paths that the marker-passing process generates which the path evaluators must separately weed out. For even very small networks, these spurious paths often represent over 90 percent of the paths generated [Charniak, 1986]. More importantly, as the size of the networks increase to represent more world knowledge, there is a corresponding explosion in the number of paths generated. Because these paths must be evaluated serially by a path evaluator, it negates marker-passing systems’ main efficiency advantage.

1.3. Overview of the Thesis

This thesis describes ROBIN [Lange & Dyer, 1989], a purely-structured connectionist model that has many of the variable binding inferencing abilities of marker-passing networks. Because ROBIN also retains the disambiguation abilities of normal structured networks, it is able to perform high-level inferencing that requires disambiguation and other aspects of frame selection.

The thesis is organized as follows: Chapter 2 gives a discussion of the type of frame-based knowledge and rules that define the hand-built knowledge bases used to construct ROBIN's networks. Chapter 3 describes the structure of the networks in detail and how they perform high-level inferencing, including how variable and role-bindings are represented, how rules are "fired", how disambiguation and frame selection are performed, and how crosstalk is kept under control. Chapter 4 provides two detailed examples of the network performing high-level inferencing. And finally, Chapter 5 is a general discussion of the model, future directions for research, how it compares to related connectionist models, and conclusions. The appendix includes a complete listing of the main knowledge base used to test ROBIN's abilities.

2. OVERVIEW OF ROBIN'S KNOWLEDGE

ROBIN's networks consist entirely of connectionist nodes [Feldman & Ballard, 1982] that perform simple computations on their inputs: summation, summation with thresholding and decay, or maximization. Connections between nodes are weighted, and either excitatory or inhibitory. ROBIN uses structured connections of nodes to encode a semantic knowledge base of related frames [Minsky, 1975]. Each frame has one or more roles, with each role having expectations and logical constraints on its fillers. Every frame can be related to one or more other frames, with pathways between corresponding roles (representing general knowledge rules) for inferencing. There is no information in the knowledge base about the specific episodes (such as **Hiding Pot**) that the networks will be used to understand.

As in nearly all structured models, ROBIN's knowledge base is hand-built. The knowledge base, made up of the conceptual frames and rules needed for a given domain, is used to construct the actual networks' structure before any processing begins. After the network has been constructed, nodes in the network are clamped to represent the surface role-bindings from an episode (such as from phrases **P1** and **P2** of **Hiding Pot**). Activation representing role-bindings and evidence for individual concepts then spreads from the nodes representing one frame to the nodes representing related frames, thus automatically instantiating other frames and performing the processes of inferencing and frame selection.

An example of how concepts are statically defined in ROBIN's general semantic knowledge bases is shown in Figure 4. The figure shows a simplified definition of the state frame **Inside-Of**, which represents the knowledge that an object is inside of a container. **Inside-Of** has three roles: an Object that is inside of something, a Location that the object is inside of, and a Planner that caused the state to be reached. The lexical phrase <Subject "is inside of" Direct-Object> directly accesses **Inside-Of**, as in "*the roast is inside of the stove.*"

2.1. Selectional Restrictions on Roles

Every role has *selectional restrictions* (or *logical binding constraints*) that tell which types of concepts may be bound to it. For instance, only a **Stove** or something that *is-a Stove* can be bound to the Location role of **Inside-Of-Stove**. This constraint is needed because **Inside-Of-Stove** is by definition a refinement of **Inside-Of** which allows the possible inference that the Object is being cooked. Similarly, **Inside-Of-Dishwasher** and **Inside-Of-Opaque** each have the binding constraints that their Location be something that *is-a Dishwasher* or *Opaque-Object*, respectively. The binding constraints defined in Figure 4 for state **Inside-Of** are that the Object must be some kind of **Physical-Object**, that the Location must be some sort of **Container-Object**, and that the Planner (if any) must be a **Human**. A role's binding constraint also serves as its *prototypical filler*, i.e. the concept that serves as the role's default binding.

2.2. Rules as Relations Between Frames

The relations that each frame has to other frames define the network's general knowledge rules and alternative inference paths. For example, in Figure 4, **Inside-Of** is related to four other frames. The first frame that it is related to is the action **Transfer-Inside**, which it is a **result-of**, since transferring an object inside of something results in that thing being inside of it. The Figure 4 also displays the links between corresponding roles; showing, for example, that the Object of **Inside-Of** can be inferred to be the same as the Object of **Transfer-Inside**. Defining **Inside-Of**'s relation to **Transfer-Inside** in this way is equivalent to defining it in the form of a rule such as:

```
R3: [Actor X Transfer-Inside Object Y Location Z]
    == results-in ==> [Object Y Inside-Of Location Z]
```

```

(FRAME Inside-Of
  State (Roles (Object (Physical-Object 0.05))
              (Location (Container-Object 0.50))
              (Planner (Human 0.05)))
        (Phrase
          (<S_"is inside of"_DO> 1.0 (Object Subject)
                                           (Location Direct-Object))
        (Result-Of
          (Transfer-Inside 1.0 (Object Object)
                               (Location Location)
                               (Planner Actor))
        (Refinements
          (Inside-Of-Stove 1.0 (Object Object)
                               (Location Location)
                               (Planner Planner))
          (Inside-Of-Dishwasher 1.0 (Object Object)
                                     (Location Location)
                                     (Planner Planner))
          (Inside-Of-Opaque 1.0 (Object Object)
                                (Location Location)
                                (Planner Planner))))

```

Figure 4. Simplified definition of the frame representing the state Inside-Of. The weights (numbers) from each of the concepts correspond to how much evidence there exists for Inside-Of given that the concept is active.

Finally, Figure 4 specifies that there are three potential refinement frames (Inside-Of-Stove, Inside-Of-Dishwasher, and Inside-Of-Opaque) which compete for selection as *the* refinement interpretation of a given instantiation of Inside-Of. These refinements are themselves related to the frames representing the probable reasons for the object being inside of the location. For example, in **Hiding Pot**, it initially appears that it is important that the pot is inside of a dishwasher (Inside-Of-Dishwasher), so that it could be cleaned. However, the final inference is that the salient property is that it is inside of something that is opaque (Inside-Of-Opaque), so that it will be hidden from sight. They are thus *mutually exclusive* parts of any one interpretation.

2.3. Connection Weights

Whenever a frame or concept is activated in a given context or episode, it provides a certain amount of *evidence* that the frames it is related to are activated. For example, if somebody has performed a Transfer-Inside into a container, then there is quite strong evidence that something is now Inside-Of that container. The relative levels of these amounts of evidence are built into the *connection weights* of the networks constructed from the knowledge base.

In general, weights are chosen on the basis of how much evidence the activity of the related frame (F_r) provides for the activity of frame being defined (F_d). Specifically, the connection weight from F_r to F_d is equal to the probability that F_d is active given the knowledge that F_r is active, or:

$$W_{F_r \rightarrow F_d} = P(F_d \mid F_r)$$

This method of selecting connection weights between concepts is similar to that used in the structured evidential reasoning networks of [Shastri, 1988]. Unfortunately, it is usually impossible to

calculate a precise probability of one action or fact given another in the uncertain domains of natural language understanding and planning. The above weight "formula" is therefore used as a rule of thumb when creating the connection weights.

The numbers in Figure 4 specify the basic connection weights from related frames to *Inside-Of* and its roles. For example, if something is inside of a stove (*Inside-Of-Stove*), then the network can definitely infer that it is *Inside-Of* something, so the connection weight from *Inside-Of-Stove* to *Inside-Of* is set at a maximum (1.0 in Figure 4). If a *Container-Object* is active, on the other hand, there would be substantial, though not definite, evidence that something is *Inside-Of* something else (since there are often things inside of mentioned containers, but not always). The weight from it to *Inside-Of* reflects this (0.50). Finally, the fact that a *Physical-Object* is active in an episode provides only limited evidence for it being *Inside-Of* something, so a very small weight is given (0.05). The actual weight values chosen are clearly arbitrary. What is important is that they be in a *range* reflecting the amount of evidence the concepts provide for their related concepts in a certain knowledge base.

2.4. Overall Knowledge Bases

Individual frame definitions combine to describe ROBIN's knowledge base of concepts and rules for inferencing. Figure 5 shows an overview of a relevant portion of a knowledge base consisting of the causal dependencies relating actions, plans, goals, and scripts [Schank & Abelson, 1977]. As can be seen, rules R1-R3 are encoded by the relations between frames shown in the figure, as are a number of the other general knowledge rules necessary to understand **Hiding Pot** and related episodes.

As Figure 5 shows, every relation from one frame to another has an inverse relation. Just as a state of *Inside-Of* can be inferred to be a *result-of* a given *Transfer-Inside* action, one can infer that a *Transfer-Inside* action *results-in* a given state of *Inside-Of*. The connection weights may be different, however: if there is a *Stove* active in an episode, then there is definitely an *Appliance* (so a weight to *Appliance* of 1.0), but if there is an *Appliance* active, then it is not necessarily a *Stove* (so a smallish weight to *Stove* of ~0.2). The complete definition of the knowledge base used to understand **Hiding Pot** and related episodes is shown in Appendix A.

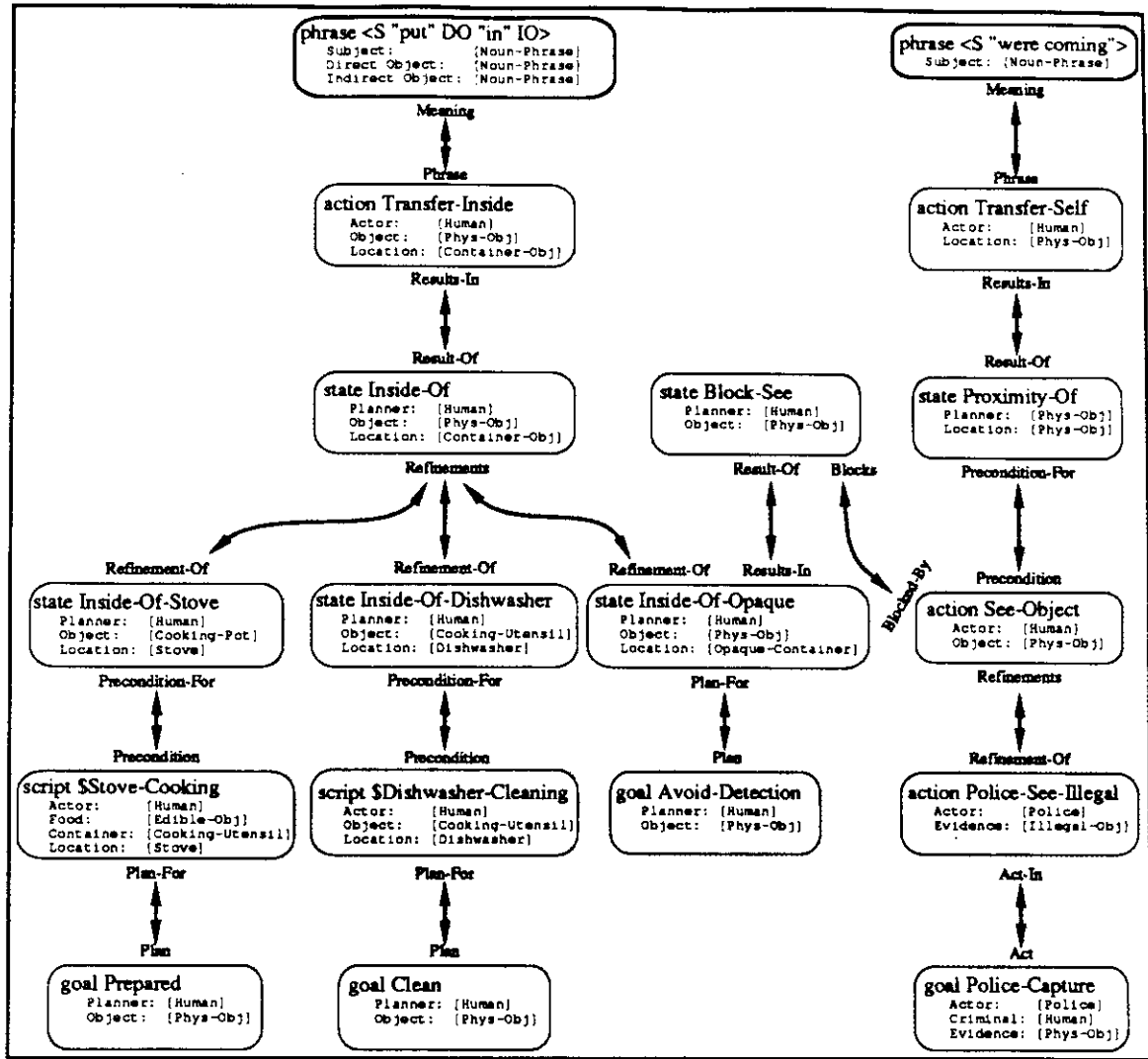


Figure 5. Overview of a relevant portion of a knowledge-base defined in ROBIN. Bracketed objects to the right of a frame's role (e.g. [Container-Obj] in the Location role of Inside-Of) represent its selectional restrictions. The symbolic frames and their relations (links) defined in the knowledge base are not actual nodes and links in the network. They are instead used to initially *construct* a portion of the network which represents them, and in which the node and link names do not affect the actual spread of activation in any way.

3. ROBIN

ROBIN's knowledge bases of frames and their relations are used to initially construct the purely connectionist networks over which inferencing is performed. Once the network is constructed, input for sentences such as **Hiding Pot** are presented to the network, with all inferencing, disambiguation, and reinterpretation happening within the network solely by activation changes. The structure of ROBIN's networks must therefore support role-binding, propagation of role-bindings according to the general knowledge rules of the knowledge base for inferencing, and weighing of contextual evidence to select the most-likely interpretation in a given context.

3.1. Role-Binding and Inferencing With *Signatures*

As in most other structured connectionist models, there is a single node in the network for each frame or role concept. Relations between concepts are represented by weighted connections between the nodes. Activation on a conceptual node is *evidential*, corresponding to the amount of evidence available for the concept and the likelihood that it is selected in the current context.

Simply representing the amount of evidence available for a concept, however, is not sufficient for complex inferencing tasks. A solution to the variable binding problem requires that some means exist for *identifying* a concept that is being dynamically bound to a role. Furthermore, the network's structure must allow these role-bindings to propagate across node pathways that encode the knowledge base's rules, thus dynamically instantiating inference paths representing the input.

3.1.1. Variable Binding With *Signatures*

The variable and role-binding problem is handled in ROBIN by network structure holding *signatures* — activation patterns which uniquely identify the concept bound to a role [Lange & Dyer, 1988]. Every concept in the network has a *signature node* that outputs its signature, a constant activation value different from all other signatures. A dynamic binding exists when a role or variable node's *binding node* has an activation matching the activation of the bound concept's signature.

In Figure 6a, the *virtual binding* of the Actor role node (of action Transfer-Inside) to John is represented by the fact that its binding node (the solid black circle) has the same activation (3.1) as John's signature node. The same binding node could, at another time, hold a different virtual binding, simply by having the activation of another concept's signature (as in Figure 6b, where it is bound to Police). The complete Transfer-Inside frame is represented in the network by the group of nodes that include the conceptual node Transfer-Inside, a conceptual node for each of its roles (only the Actor role shown), and the binding nodes for each of its roles.

3.1.2. Structure of the Network

The most important feature of signature activation is that it propagates across paths of binding nodes to generate candidate inferences. Figure 7 illustrates the structure of the network that automatically accomplishes this.

The conceptual nodes and connections on the bottom plane of Figure 7 (i.e. Transfer-Inside and its Object role) are part of the normal semantic network constructed from the knowledge base of Figure 5. Nodes and connections for the Actor, Planner, and Location roles are not shown. The connections between nodes on this bottom plane are specified by the frame definitions of the knowledge base. For example, the weighted connection from node Transfer-Inside to node Inside-Of represents the *result-of* relation defined in Figure 4. As in other structured models, activation propagating across this structure of the network is *evidential*.

The top plane of Figure 7, on the other hand, consists of the network's binding and signature nodes, over which *signature* activation (representing dynamic role-bindings) spreads. Each role has several binding nodes (two of which are shown). There are no connections from signature nodes to binding nodes, but there are *unit-weighted* connections between *corresponding* binding

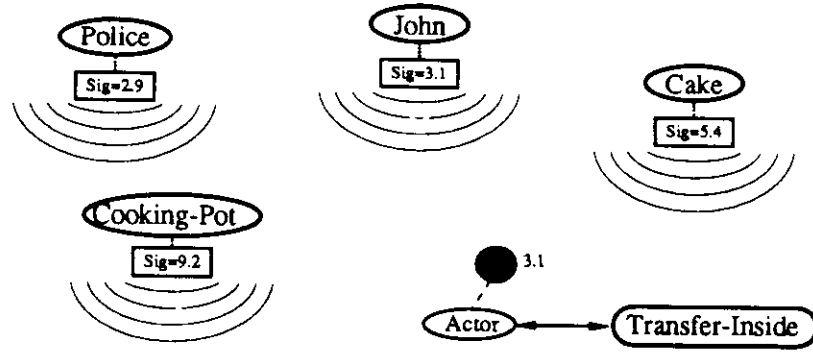


Figure 6a. Several concepts (ovals) and their uniquely-identifying signature nodes (rectangles) are shown, along with the Actor role of the Transfer-Inside frame. The Actor role has a *virtual binding* to John because its binding node (black circle) has the same activation (3.1) as John's signature.

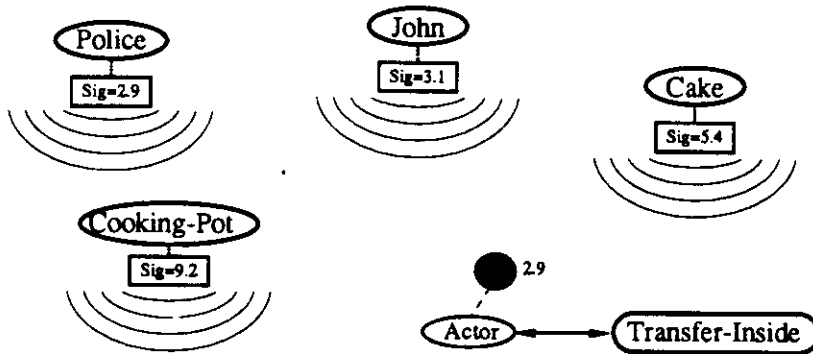


Figure 6b. The same binding node holding a different virtual binding, this time to Police.

nodes over which inferences can be made. For example, the filler of Inside-Of's Object role can be inferred to be the same as the filler of Transfer-Inside's Object (as defined in Figure 4). There is therefore a connection from the left binding node of Transfer-Inside's Object to the left binding node of Inside-Of's Object. A similar link goes between the right binding nodes, as well as one-to-one connections from the (unseen) binding nodes of Transfer-Inside's other roles (Actor and Location) to the binding nodes of Inside-Of's corresponding roles (Planner and Location, respectively).

3.1.3. Activation Functions

There are different activation functions for the conceptual nodes of the bottom evidential layer and the binding nodes of the top signature layer.

3.1.3.1. Activation of Concept Nodes

The activation function of the network's conceptual nodes is equal to the weighted sum of their inputs plus their previous activation times a decay rate, or:

$$a_c(t+1) = \sum_i w_{ic} o_i(t) + a_c(t) (1-\Theta)$$

where $a_c(t)$ is the activation of conceptual node c at cycle t , w_{ic} is the incoming weight from node i to node c , $o_i(t)$ is the output of node i at cycle t , and Θ is the activation decay rate of conceptual nodes when they are receiving no input. The output function of the conceptual nodes is a simple linear threshold:

$$o_c(t) = \begin{cases} a_c(t) & a_c(t) \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

where θ is the output threshold of all of the conceptual nodes. Generally, the activation values of the conceptual nodes in the network range from 0 to about 1. The output threshold θ is set quite low (usually around 0.05), and so is not much of a factor in the spread of activation. The net effect of the activation and output functions of the conceptual nodes is to allow them to “weigh” evidence from their related concepts, with nodes in paths between multiple sources of activation (i.e. in part of an inference chain between two phrases) tending to reinforce each other.

3.1.3.2. Activation of Binding Nodes

The activation and output functions of the binding nodes are equal to the *maximum* of their unit-weighted inputs, or:

$$a_b(t+1) = \text{MAX} (w_{1b}o_1(t), w_{2b}o_2(t), \dots, w_{nb}o_n(t)), \quad o_b(t) = a_b(t)$$

where $a_b(t)$ is the activation of binding node b at cycle t , and $o_1 \dots o_n$ are the outputs of all binding nodes that have incoming links to binding node b . Since all of the w_{ib} to binding nodes have unit weight, this causes the activation of a binding node to take on the activation of any of its active incoming binding nodes — and therefore allows signatures to be propagated without alteration.

3.1.4. Propagation of Signatures for Inferencing

Initially there is no activation on any of the conceptual or binding nodes in the network. To initiate the inferencing process, a phrase and its role-bindings are presented to the network by hand-clamping the proper roles' binding nodes to the signatures of the concepts bound in the phrase¹. Thus, for phrase P1 (“John put the pot inside the dishwasher”), the binding nodes of Transfer-Inside’s Actor, Object, and Location roles are clamped to the signatures of the concepts meant by the words “John”, “pot”, and “dishwasher”, respectively. For example, the binding nodes of Transfer-Inside’s Object are clamped to the activations 6.8 and 9.2, which are the signatures for objects Marijuana and Cooking-Pot, respectively, representing the candidate bindings from the word “pot” (Figure 7)².

At the same time, the lexical concept nodes for each of the words in the phrase are clamped to a high level of evidential activation. In the case of phrase P1, this clamping directly ends up providing evidential activation for concepts John (from lexical node “John”), Transfer-Inside (from phrase <S “put” DO “in” IO>), Cooking-Pot and Marijuana (from lexical node “pot”), and Dishwasher (from lexical node “dishwasher”).

¹ROBIN does not currently address the problem of performing the original syntactic role-binding assignments, i.e. that “pot” is bound to the Object role of the phrase. Rather, ROBIN’s networks are given these initial bindings and use them for high-level inferencing.

²An alternative input, such as “George put the cake inside the oven”, would be done simply by clamping the signatures of its bindings (i.e. George, Cake, and Oven) instead. A completely different set of inferences would then ensue.

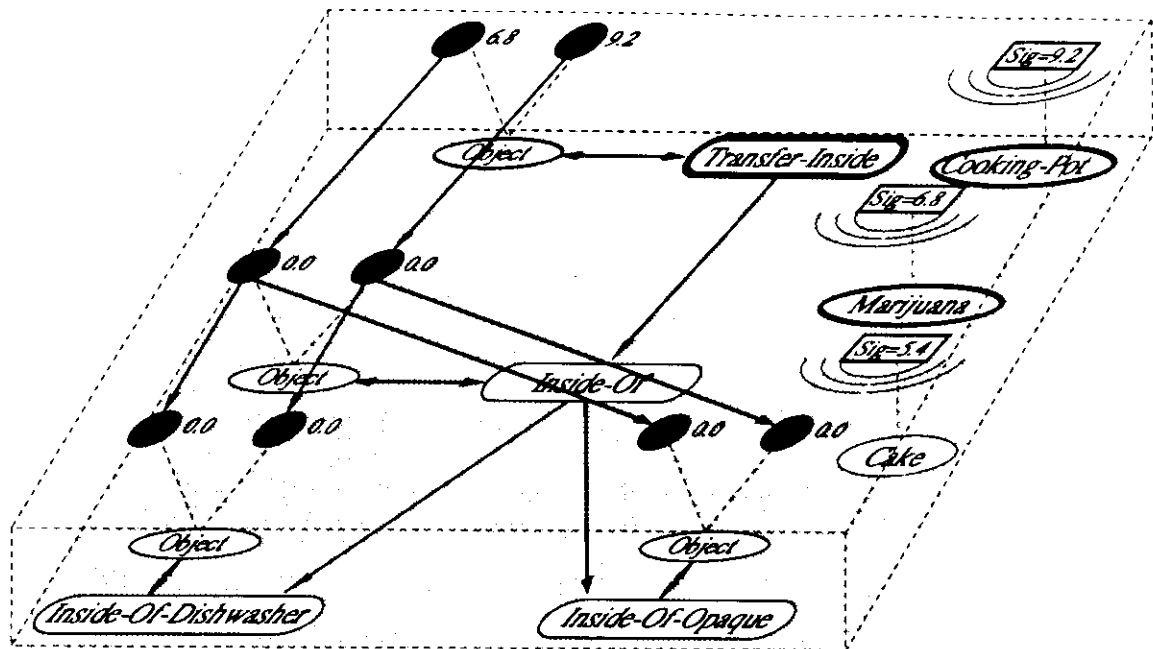


Figure 7. Simplified ROBIN network segment showing the parallel paths over which evidential activation (bottom plane) and signature activation (top plane) are spread for inferencing. For simplicity, the nodes and connections representing Inside-Of-Stove and the rest of the semantic network not shown. The figure shows the initial activation and clamping for the phrase "John put the pot inside the dishwasher" (P1). Signature nodes (outlined rectangles) and binding nodes (solid black circles) are in the top planes. Thickness of conceptual node boundaries (ovals) in the bottom plane represents their levels of evidential activation. (Node names do not affect the spread of activation in any way. They are simply used to initially set the network's structure and to aid in analysis.)

With the original role-bindings thus input to the network, activation starts to spread from the initial clamped activation values. In Figure 8, Inside-Of receives evidential activation from Transfer-Inside, representing the strong evidence that something is now inside of something else. Concurrently, the signature activations on the binding nodes of Transfer-Inside's Object propagate to the corresponding binding nodes of Inside-Of's Object (Figure 8), since each of the binding nodes calculates its activation as the maximum of its inputs.

For example, Inside-Of's left Object binding node has only one input connection, that from the corresponding left Object binding node of Transfer-Inside. Since the connection has a unit weight and the left Object binding node of Transfer-Inside has an activation of 6.8, the activation of Inside-Of's left Object binding node also becomes 6.8 (Marjuana's signature). The potential binding of Cooking-Pot (signature 9.2) to Inside-Of's right Object binding node propagates at the same time, as do the bindings of Inside-Of's Planner role to the signature of JOHN and its Location role to the signature of Dishwasher.

By propagating signature activations from the binding nodes of Transfer-Inside to the binding nodes of Inside-Of, the network has thus made its first inference. Because of the signatures now on Inside-Of's binding nodes, the network not only represents that something is inside of something else, but also represents the inference of *exactly which thing is inside the other* (I7 in Table 1).

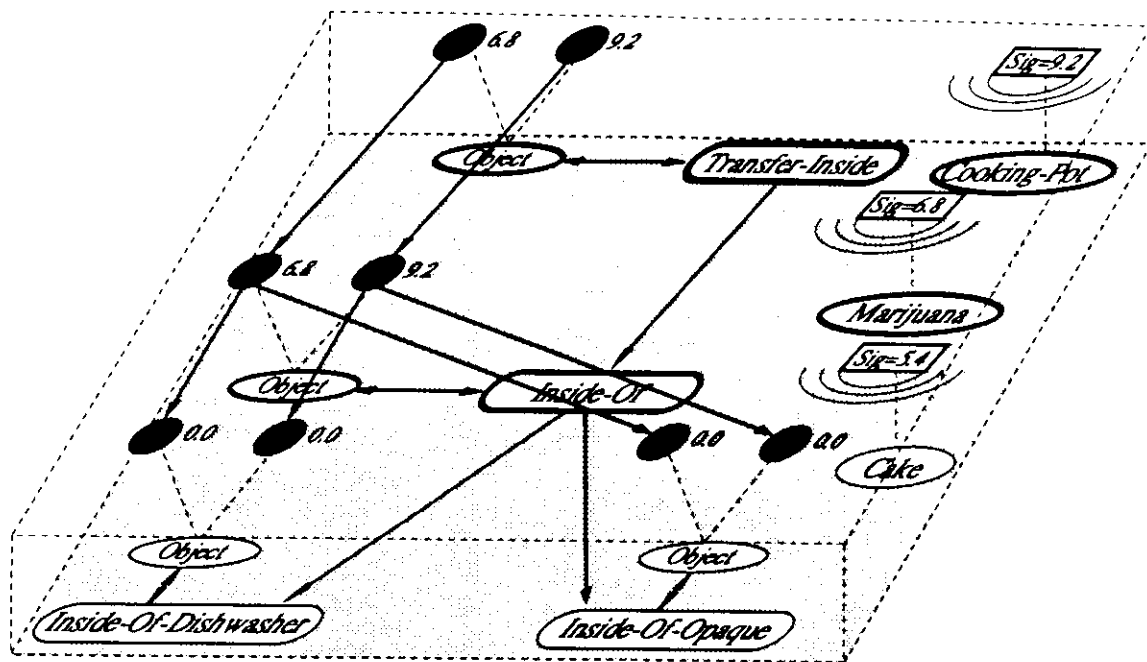


Figure 8. Evidential and signature activations in **Hiding Pot** after reaching **Inside-Of**.

From the activations of this newly inferred piece of knowledge, the network continues to make subsequent inferences. Evidential activation next spreads from **Inside-Of** to its refinements **Inside-Of-Dishwasher** and **Inside-Of-Opaque**. At the same time, the signatures on the binding nodes of **Inside-Of**'s roles propagate to **Inside-Of-Dishwasher** and **Inside-Of-Opaque**'s corresponding binding nodes, instantiating them (Figure 9). The network thus makes the inference that the reason for the **Marijuana** or a **Cooking-Pot** being inside of the **Dishwasher** was either because it is a dishwasher, or because it is opaque. From there, the signature and evidential activations continue to propagate through other parts of the network structure constructed from the definitions of Figure 5, instantiating the chains of related concepts down to the **Clean** goal, with some activation reaching goal **Avoid-Detection**, state **Block-See**, and so on.

Signatures thus propagate without change over the inference binding paths of the network constructed by the definitions of the knowledge base. As a result, **ROBIN** is able to dynamically instantiate inference paths and distinguish each of their inferred role-bindings.

3.1.5. Discussion of Signatures

There are a couple of points that is important to make about signatures. The first is that their actual or relative activation values do not affect the network's processing. The signatures of **Marijuana** and **Cooking-Pot** were arbitrarily chosen to be 6.8 and 9.2 when the network was constructed. However, they could just as easily have been chosen to be any other (even the reverse) values. It is only necessary that each signature be different from all others — and thus uniquely identify the concept bound to a role.

The second point is that a signature *can* happen to have the same activation value as the evidential activation on a conceptual node. The reason for this is that the paths over which evidential and signature activation spread are parallel to and completely separate from each other (i.e. along the bottom and top planes in Figure 9). It is therefore irrelevant whether or not a conceptual node coincidentally has an activation that is the same as some concept's signature. The activation on a conceptual node is always interpreted as the amount of evidence for that concept in the current

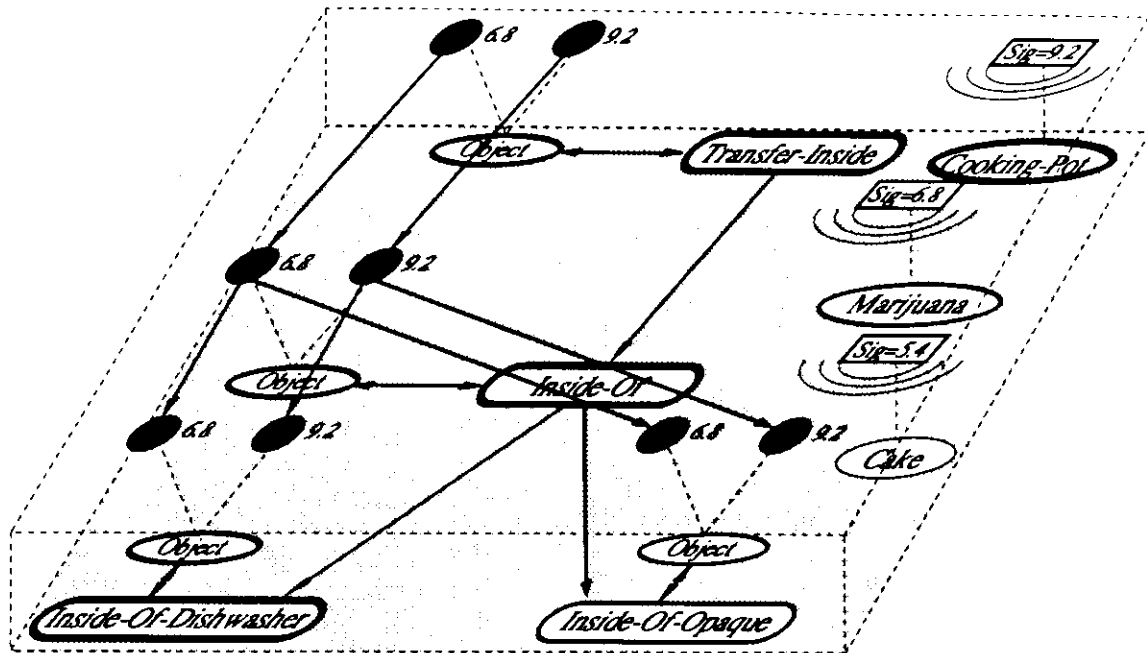


Figure 9. Activation after quiescence has been reached in processing for phrase **P1**. **Cooking-Pot** and **Inside-Of-Dishwasher** have higher (evidential) activations than **Marjuana** and **Inside-Of-Opaque**, as is illustrated by their thicker ovals.

context, while the activation on a binding node is always interpreted as a signature representing a role-binding.

3.2. Frame Selection With Evidential Activation

The ability to maintain variable bindings and propagate them throughout the network is critical for high-level inferencing. However, being able to dynamically generate inference paths alone is not sufficient for cognitive tasks such as natural language understanding and planning. The problem is that there are generally *multiple* alternative inference paths possible, only one of which best explains the input. Choosing the single most-plausible interpretation in a given context is one of the most difficult problems in high-level inferencing, that of *frame selection* [Lytinen, 1984] [Lange & Dyer, 1989a].

An example of the need for frame selection can be seen in Figure 9, where both **Inside-Of-Dishwasher** and **Inside-Of-Opaque** have been instantiated with signatures inferred from phrase **P1** ("John put the pot inside the dishwasher"). The inference path representing **P1** has therefore already split into two alternatives: one *candidate path* that includes **Inside-Of-Dishwasher**, which is part of the interpretation that John is trying to clean the pot, and another candidate path including **Inside-Of-Opaque**, which is part of the interpretation that he is trying to hide it.

The network must somehow be able to weigh the evidence for each of these two alternative refinements of frame **Inside-Of** so that the most plausible of the two inference paths can be selected. With only phrase **P1**, the evidence appears to be that **Inside-Of-Dishwasher** is the best interpretation, but when **P2** ("because the police were coming") is presented, it appears that the **Inside-Of-Opaque** path is the most likely.

Besides the problem of selecting the concept refinement of frame **Inside-Of**, **Hiding Pot** also requires word-sense disambiguation to select the appropriate meaning of the word “*pot*”. The same contextual evidence that causes **Inside-Of-Dishwasher** to be selected in **P1** should cause **Cooking-Pot** interpretation of “*pot*” to become chosen, while the switch to **Inside-Of-Opaque** with evidence from **P2** should cause a reinterpretation to **Marijuana**.

3.2.1. Selection by the Evidential Semantic Network

Deciding between the competing inference paths instantiated by signature activation is the function of the evidential portions of ROBIN’s networks (such as the conceptual nodes on the bottom layer of Figure 9). The activations of the conceptual frame nodes are always approximately proportional to the amount of evidence available for them from their bindings and their related frames. The inference path selected as the interpretation in any given context is therefore simply *the most highly-activated path of frame nodes and their bindings*¹.

Thus, if the conceptual node for **Inside-Of-Dishwasher** has a higher level of (evidential) activation at stability than the node for **Inside-Of-Opaque**, then it will be selected as the refinement of **Inside-Of** and become part of the winning inference path. On the other hand, if **Inside-Of-Opaque** has the higher level of activation, then it will be selected.

3.2.2. Selection of Ambiguous Role-Bindings

Besides being able to select the most-highly activated inference path, the network must be also able to decide between ambiguous role-bindings. All meanings of an ambiguous word are bound to a role with signature activation, as was shown in Figure 7. Each role has several binding nodes, so that multiple candidate bindings for a given role may be propagated through the network at once. In general, this requires that there be as many binding nodes per role as there are possible meanings of the most ambiguous word in the network. In the network used to process **Hiding Pot** and other similar inputs, for example, there are actually three (though Figures 7 through 9 show only two) — since “*pot*” can mean **Marijuana**, **Cooking-Pot** or **Planting-Pot**. When a word bound to a role is unambiguous (like “*dishwasher*”), the extra binding nodes simply remain inactive.

Though having enough binding nodes per role to allow simultaneous propagation of ambiguous bindings increases the size of the network by a small constant factor, it is crucial for resolving ambiguities in context. The network’s interpretation of which binding is selected at any given time is the one whose *conceptual node has greater evidential activation*. Because all candidate bindings propagate at once, with none being discarded until processing is completed, ROBIN is able to *handle meaning reinterpretations without resorting to backtracking*.

3.2.3. Structure of the Evidential Network

The structure of the conceptual layer and the activation function of the conceptual nodes is constructed so that the activation of each conceptual node is always approximately proportional to the amount of evidence available for it. For example, when only **P1** (“*John put the pot inside the dishwasher*”) is presented, there is more evidence and evidential activation for **Cooking-Pot** and **Inside-Of-Dishwasher** than for **Marijuana** and **Inside-Of-Opaque**. When the rest of **Hiding Pot** is presented, however, the balance of evidence — and thus evidential activation — shifts to **Marijuana** and **Inside-Of-Opaque**.

In general, a *candidate frame* is likely to be active if one or more of its instantiating frames are active. For example, if there was a **Transfer-Inside** of an object to a location, then there is strong

¹The network’s “decision” or “selection” is actually simply the interpretation that the human modeler gives to the levels of activation present in it, as in all connectionist models.

evidence that the object is Inside-Of that location. The higher the activation of the instantiating frame, and the stronger the connection weight from it to the candidate frame, the more likely that the candidate frame is active. Similarly, a candidate frame can receive evidential activation from all of the frames that are directly related to it, and from the evidential activation of each of its conceptual roles.

Simply having frames receive evidential activation through direct connections from their related frames, however, would cause serious problems. Candidate frames that have potential relations to a large number of frames would always win out over candidates that have a smaller number of related frames. The activation of Inside-Of, for example, would always dominate Transfer-Inside, simply because Inside-Of has a very large number of potential refinement frames. In reality, however, those refinements are mutually exclusive, and only one will be chosen as *the* refinement of a given instantiation of Inside-Of. Thus, the only refinement relation that actually provides evidence for Inside-Of at a given cycle is the one that is most active.

Because of this, connections from related frames pass through an *input branch* node for their relation before they are received by the candidate frame. This is shown in Figure 10, which displays the connections between nodes on a portion of the evidential network centering around frame Inside-Of. For example, the weighted connections from Inside-Of-Dishwasher, Inside-Of-Opaque, and Inside-Of-Stove go into Inside-Of's refinements branch rather than directly into Inside-Of. Relation input branches calculate their activation as the maximum of their inputs — so that only the currently selected (i.e. maximally activated) interpretation provides evidence for the frame¹.

Similarly, connections providing evidence from the activation of a frame's roles pass through its Roles input branch (as do the Location, Object, and Planner roles of Inside-Of in Figure 10). If none of the roles of a given frame are active, then that frame should receive no evidence from its roles. If all of them are active, then the frame should get maximum role evidence. Ratios in between should provide proportionate evidence. The amount of this evidence should not vary with the number of roles that a frame has: a frame with only a single role should receive just as much activation evidence if its one role is active as a frame with many roles that are all active. Hence the frame's Roles input branch calculates its activation as the *average* of its inputs.

Role nodes, like frame nodes, have several input branches. A role, however, gets evidence only from its competing prototypical fillers (the Prototypes branch) and from the frames that it is used in (the Used-In branch). Each branch calculates its activation as the maximum of its inputs, like the relation input branch for frames. With these evidential connections, the Location role of Inside-Of-Dishwasher, for example, will become activated if either its frame (Inside-Of-Dishwasher) or its prototypical filler (Dishwasher) is activated (Figure 10).

3.2.4. Activation Control

A major issue for all structured connectionist networks is controlling the spread of activation. Other spreading-activation models have usually addressed this problem by using direct inhibitory connections between competing concepts (e.g. [Waltz & Pollack, 1985]). For inferencing tasks, however, the inhibitory connections that these networks use are usually semantically unjustifiable and combinatorially explosive. The biggest problem, however, is that they are *winner-take-all networks*, acting to kill the activations of input interpretations that do not win the competition. This becomes a problem when a new context arises that makes an alternative interpretation more plausi-

¹Input branches are analogous to the input *sites* on [Cottrell & Small, 1982]'s "case" units, which were used to make sure that each case unit only received activation from its maximally activated prototypical filler and predicate.

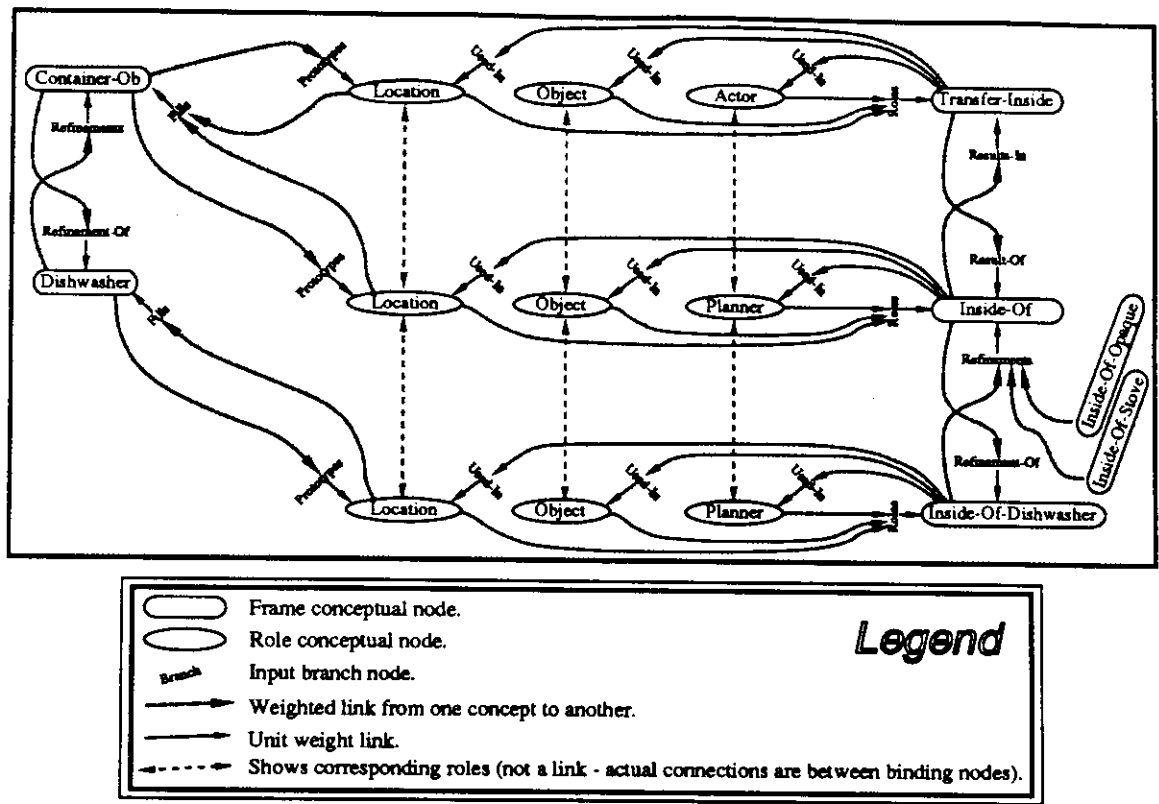


Figure 10. Detailed view of the evidential portion of the network for frame Inside-Of and part of frames Transfer-Inside and Inside-Of-Dishwasher. As usual, the node names are used only for initial construction of the network, and do not affect the spread of activation.

ble. With the activations of the alternative interpretations killed by the inhibition from the false winner, it is exceedingly difficult for the activation from the new context to revive the correct one. The automatic backtracking capabilities of the networks are thus sabotaged.

ROBIN, on the other hand, has no inhibitory links between competing concepts. It instead uses a group of nodes which act as a global inhibition mechanism. These *global inhibition* nodes (Figure 11) serve to inhibit by equal proportions (short-circuit) all concepts in the network when their average activation becomes too high. The concepts in the network are thus free to keep an activation level relative to the amount of evidence in their favor. Global inhibition nodes are similar to the "regulator units" of [Touretzky & Hinton, 1988], except that their regulator units are *subtractive inhibitory*, subtracting a constant amount of activation from all nodes and implementing a winner-take-all network, while ROBIN's global inhibition nodes are *short-circuiting inhibitory*, controlling the spread of activation, but leaving *relative* values of evidential activation unchanged.

The activation function of conceptual nodes shown in chapter 3.1.3.1 is incomplete, having left out the short-circuiting inhibition term. The actual activation function for the conceptual nodes in the network is:

$$a_c(t+1) = \frac{\sum_i w_{ic} o_i(t) + a_c(t) (1-\Theta)}{a_g(t)}$$

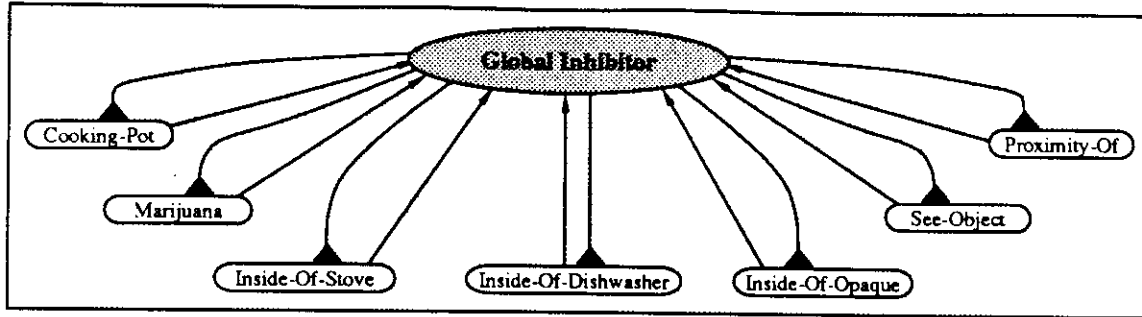


Figure 11. A ROBIN global inhibition node. Inputs to global inhibitor nodes from concept nodes are unit weighted, while outputs to concept nodes are “short-circuiting” inhibitory. The higher the total incoming activation to the global inhibitor, the greater the activation that all the concepts are divided by.

where the other terms are as before, and $a_g(t)$ is the activation at cycle t of the conceptual global inhibitor node.

Because ROBIN’s short-circuiting global inhibition mechanism allows all concepts in the network to hold a level of evidential activation relative to the amount of evidence in their favor (as opposed to driving the “losers” down to 0 using a winner-take-all network), ROBIN is able to easily perform reinterpretation. When new context that favors an alternative interpretation over a previous one enters the network, it boosts the new interpretation’s relative level of evidential activation — often being enough to cause the new interpretation to become most highly-activated. This occurs in **Hiding Pot**, in which the evidence from P1 (“John put the pot inside the dishwasher”) initially favors Cooking-Pot, but in which later evidence from the context of P2 (“the police were coming”) causes a reinterpretation to Marijuana.

3.2.5. Disambiguation Example

With the basic structure of signature propagation and the evidential network described, we can now explain what occurs as activation spreads during processing of input for “*John put the pot inside the dishwasher*” (P1). The full knowledge base needed to understand this and related sentences is embedded in the network using the signature structure of Figure 7 integrated with the evidential network structure of Figure 10.

Processing of P1 proceeds as described in 3.1.4 after the network’s inputs have been clamped to represent it. Figure 12 shows an overview of the levels of activation and inferences made in a portion of the network after the network’s activation has settled for P1. As in Figure 9, the role-bindings of Inside-Of-Dishwasher and Inside-Of-Opaque have been inferred by propagation of signatures. Those activations and evidential activation on the conceptual nodes continued to propagate along the chain of related concepts down to instantiate the Clean goal, and along the other path to goal Avoid-Detection, state Block-See, and finally action See-Object and its refinement Police-See-Illegal, where the level of evidential activation drops below threshold and so stops propagating.

There are two different possible interpretations of P1 in the network: the chain running through Inside-Of-Dishwasher to Clean (representing that John was trying to clean a Cooking-Pot), and the chain running through Inside-Of-Opaque to Avoid-Detection, Block-See, and See-

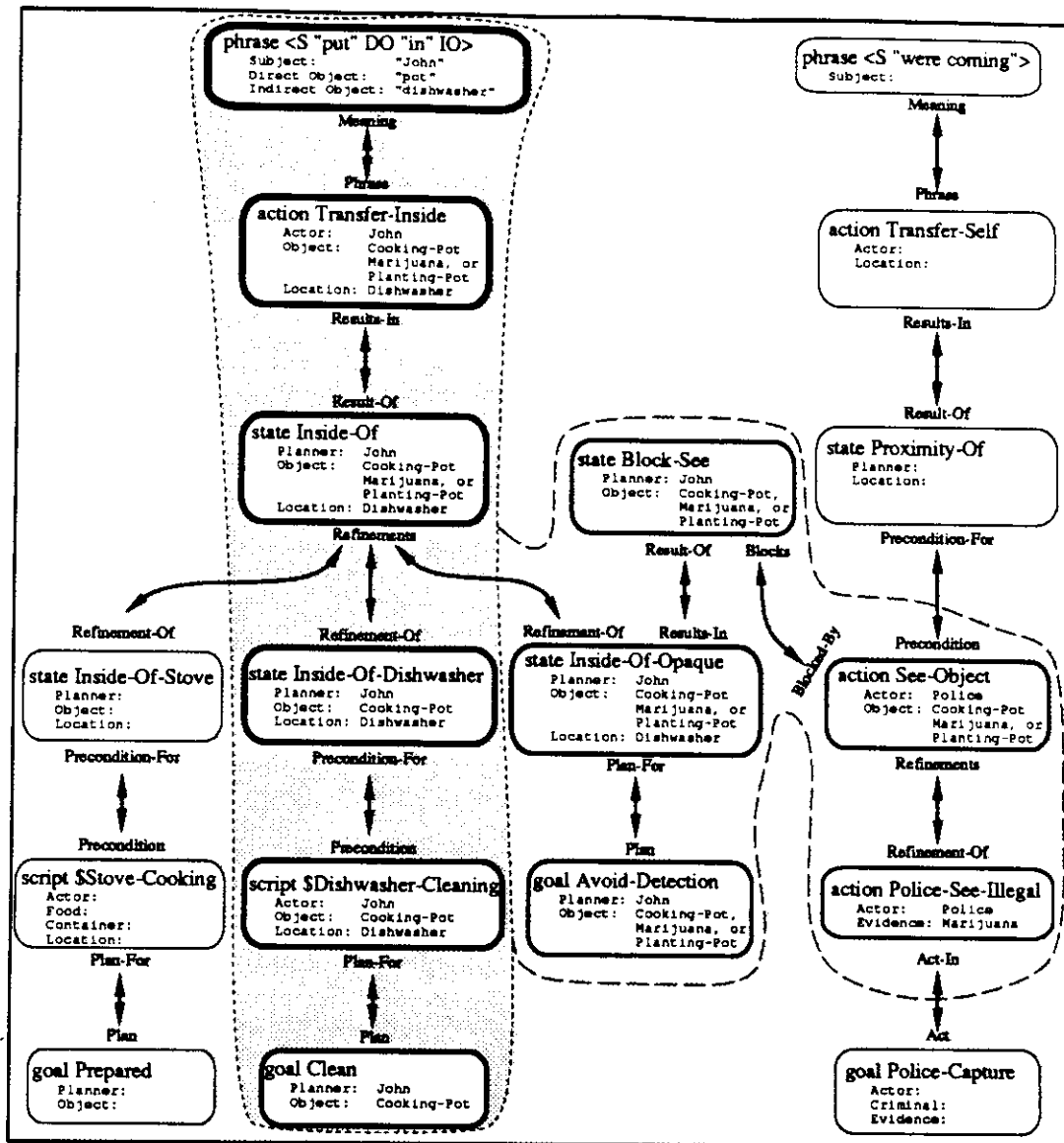


Figure 12. Overview of a small portion of a ROBIN semantic network (actually embedded in network structure such as in Figures 7-9) showing inferences dynamically made after clamping of the inputs for phrase P1. Thickness of frame boundaries shows the amount of *evidential* activation on the frames' conceptual nodes. Role fillers shown are the ones dynamically instantiated by propagation of *signature* activation over the role's binding nodes. Darkly shaded area indicates the most highly-activated path of frames representing the most probable plan/goal analysis of the input. Dashed area shows the losing hiding interpretation. Frames outside of both areas show a very small portion of the rest of the network. These frames received no evidential or signature activation from the phrase.

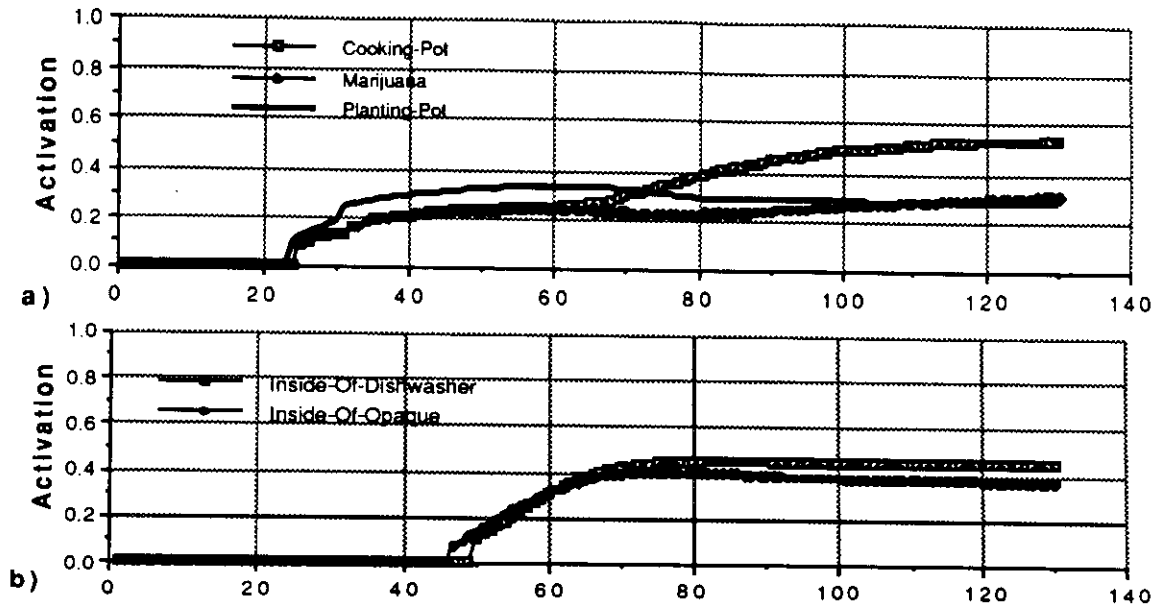


Figure 13. Evidential activations of meanings of word “pot” and competing refinements of Inside-Of after clamping for “John put the pot inside the dishwasher” (P1).

Object (representing that John was trying to hide either a Cooking-Pot, Marijuana, or a Planting-Pot)¹.

However, after the spread of activation, Inside-Of-Dishwasher has more evidential activation than Inside-Of-Opaque because of feedback between it and Cooking-Pot and Dishwasher, which have strong connections because of their highly stereotypical usage in \$Dishwasher-Cleaning (see plot of activations in Figure 13). The Inside-Of-Dishwasher path is thus selected as the refinement of Inside-Of, so it and the rest of the frames along the Clean path are chosen as the interpretation of the phrase (represented by the darkly shaded area in Figure 12). Also, because of reinforcement and feedback from the Inside-Of-Dishwasher path, Cooking-Pot ended up with more evidential activation than either Marijuana or Planting-Pot (Figure 13). Cooking-Pot is thus selected over the Marijuana and Planting-Pot *bindings* throughout the network.

3.2.6. Evidential vs Signature Activation

It is important to emphasize the differences between ROBIN’s evidential and signature forms of activation. Both are simply activation from a computational point of view, but they propagate across separate pathways and fulfill different functions.

¹In this network, Avoid-Detection and Block-See are complementary parts of the same chain, since Inside-Of-Opaque is a Plan-For Avoid-Detection and it Results-In Block-See. However, Inside-Of-Dishwasher and Inside-Of-Opaque form different chains, since they are mutually exclusive refinements of Inside-Of.

Evidential Activation:

- 1) *Previous work* — Similar to the activation of other structured models.
- 2) *Function* — Activation on a conceptual node represents the amount of evidence available for a node and the likelihood that its concept is selected in the current context.
- 3) *Node pathways* — Evidential activation spreads along weighted link pathways between related frames.
- 4) *Dynamic structure* — Evidential activation decides among candidate structures; i.e. **Cooking-Pot** is more highly-activated than **Marijuana** at the end of **P1**, so is selected as the currently most plausible role-binding throughout the inference path in Figure 12.

Signature Activation:

- 1) *Previous work* — First introduced in ROBIN.
- 2) *Function* — Signature activation on a binding node is part of a unique pattern of activation representing a dynamic, virtual binding to the signature's concept.
- 3) *Node pathways* — Signature activation spreads along role-binding paths (having unit-valued weights that preserve their activation) between corresponding roles of related frames.
- 4) *Dynamic structure* — Signature activation represents a potential (candidate) dynamically instantiated structure; i.e., that either **Marijuana** or **Cooking-Pot** is **Inside-Of** a **Dish-washer**.

3.3. Selectional Restrictions

Although the propagation of signatures allows ROBIN to dynamically generate inferences and evidential activation allows the one with the most evidence in a given context to be selected, there is still the potential problem of crosstalk from *logically unrelated inferences*. An example of this is the following sentence:

"John ate some rice before he went to church." (Church Service)

The most probable interpretation of **Church Service** is that **John** had rice for breakfast before he went to attend services at his church (**\$Church-Service**). Without considering selectional restrictions, however, the node for the **\$Wedding** script would likely become the most highly-activated, because of the combined activity of **Church** and **Rice**. The **\$Church-Service** script would lose out, because it would only receive evidence from **Church**.

Clearly there is a need to inhibit the spread of activation for inferencing when frames' selectional restrictions are violated. In **Church Service**, **Rice** should not provide any evidence for **\$Wedding**, because it was being eaten and not thrown.

Returning to the processing of **Hiding Pot** and examining the possible refinements of **Inside-Of** (Figure 14), we find that **Inside-Of-Stove** would not have received enough evidential activation to compete with **Inside-Of-Dishwasher** or **Inside-Of-Opaque**. They both receive strong activation from their **Locations'** fillers, **Dishwasher** and **Opaque-Object**, and so easily outstrip it. This is as it should be, since **John** could have been either trying to clean the pot or hide it, but (assuming he is rational) there is no way that **John** was trying to cook it by putting it into the dishwasher.

Frame	Binding Constraints	Used In
Inside-Of-Stove	A Cooking-Pot is inside of a Stove	\$Stove-Cooking
Inside-Of-Dishwasher	A Utensil is inside of a Dishwasher	\$Dishwasher-Cleaning
Inside-Of-Opaque	A Physical-Object is inside of an Opaque-Object	Avoid-Detection

Figure 14. Three of the competing refinements of state Inside-Of.

But what if Stove or \$Stove-Cooking happened to be strongly activated from previous processing? If Stove or \$Stove-Cooking is highly activated, then it is quite conceivable that Inside-Of-Stove could end up with more activation than Inside-Of-Dishwasher and Inside-Of-Opaque. It would thus be chosen as the refinement of Inside-Of, and the network would arrive at the ludicrous decision that John was trying to cook something in the pot when he put it into the dishwasher.

These examples illustrate the necessity to have *selectional restrictions control the spread of activation*. Frames whose binding constraints (as defined by the knowledge base, as in Figure 4) are violated, and hence cannot possibly be part of the inference path, should not receive either evidential or signature activation.

3.3.1. Overview of the Frame Selection Process

To solve the problem of crosstalk from unrelated inferences, the structure of the network insures that a frame competing for selection is ruled out completely when its selectional restrictions are violated. ROBIN instantiates, with signature and evidential activation, only those candidate frames whose binding constraints are met.

Gating and constraint-matching built into the network's structure assure that only legal interpretations receive evidence for selection, thus avoiding crosstalk from logically unrelated inferences (such as eating the Rice in Church Service, or priming of Stove in Hiding Pot). The accumulation of evidential activation solely from *applicable* context allows one of the legal candidates to be selected as the most likely interpretation.

Though *all inferencing is accomplished solely by the spread of activation* through the structure of the network, the complete frame selection process performed by this propagation can be viewed as a four-part process:

- 1) *Choosing candidate frames:* When the role bindings of a frame match the selectional restrictions on the roles of a related frame, then that related frame becomes a *candidate frame* for instantiation. Related frames are rejected when their selectional restrictions are violated.
- 2) *Propagating signature bindings to candidate frames:* Candidate frames receive signature activation (representing role-bindings) from their instantiating frame. New candidate inferences can then propagate from each of the candidate frames to explore their respective inference paths.
- 3) *Propagating evidential activation to candidate frames:* Candidate frames receive weighted evidential activation from their instantiating frame. Candidates whose binding constraints are only partially matched receive proportionately less evidential activation than if their constraints were matched perfectly.
- 4) *Selection between candidate instantiation frames:* At any given time, the candidate frame with the most evidential activation represents the preferred interpretation. Commitments may change if new context gives more evidence to a competing frame.

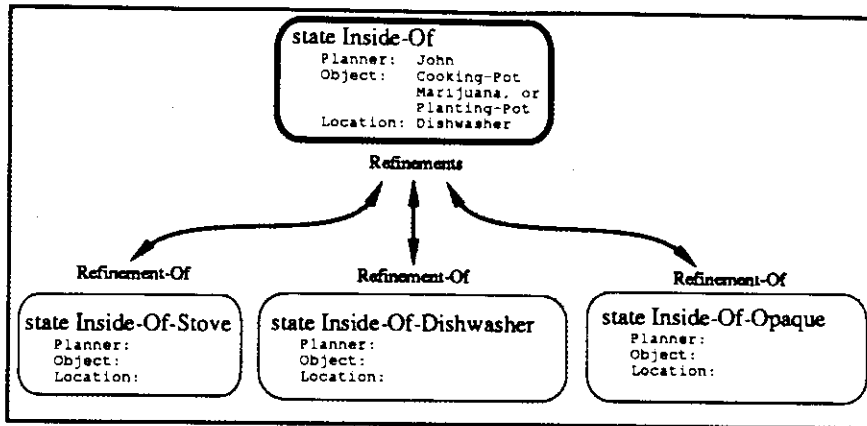


Figure 15a. An example of the frame selection problem — overview of bindings instantiated with signature activation (Figure 8).

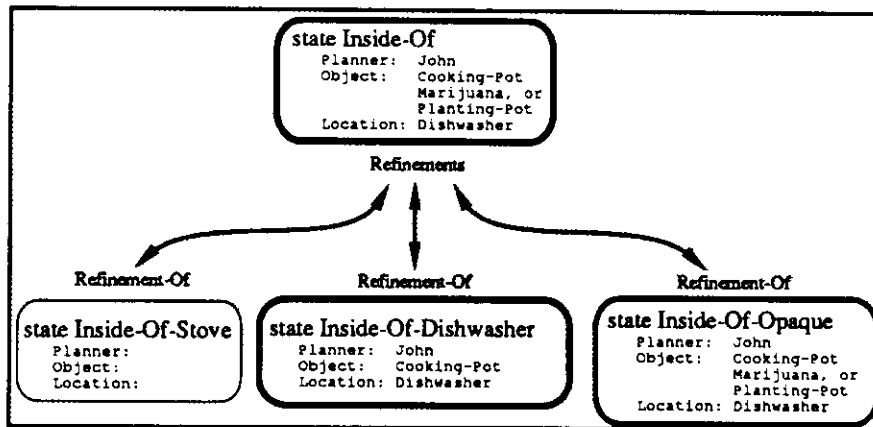


Figure 15b. Overview after Inside-Of-Dishwasher and Inside-Of-Opaque become candidate refinements of Inside-Of (Figure 9).

As an example of how the frame selection process proceeds in ROBIN, consider Figure 15a, which shows frame *Inside-Of* and its three refinements (from Figure 14) during processing for phrase **P1**. At this point, evidential activation and signature role-bindings have reached *Inside-Of* (as in Figure 8), so the candidates for its concept refinement must now be chosen. *Inside-Of-Stove* is rejected since a *Dishwasher* does not match the *Stove* constraint on its *Location* role. It therefore receives no signature or evidential activation. *Inside-Of-Dishwasher*, however, is chosen as a candidate refinement frame, since its constraints are matched. *Inside-Of-Opaque* is also chosen as a candidate, since a *Dishwasher* *is-a* *Opaque-Object*.

The result can be seen in Figure 15b, where both candidates have been instantiated. *Marijuana* and *Planting-Pot* violate the constraints on the *Object* of *Inside-Of-Dishwasher* (neither of them is cleaned in a dishwasher), so only *Cooking-Pot* is allowed through. The chosen candidate will be the frame with the greatest evidential activation. After activation has settled for **P1** of **Hiding Pot**, *Inside-Of-Dishwasher* has the greater evidential activation (thicker oval), and is selected as the refinement-of *Inside-Of*, serving as the plan for cleaning his *Cooking-Pot*.

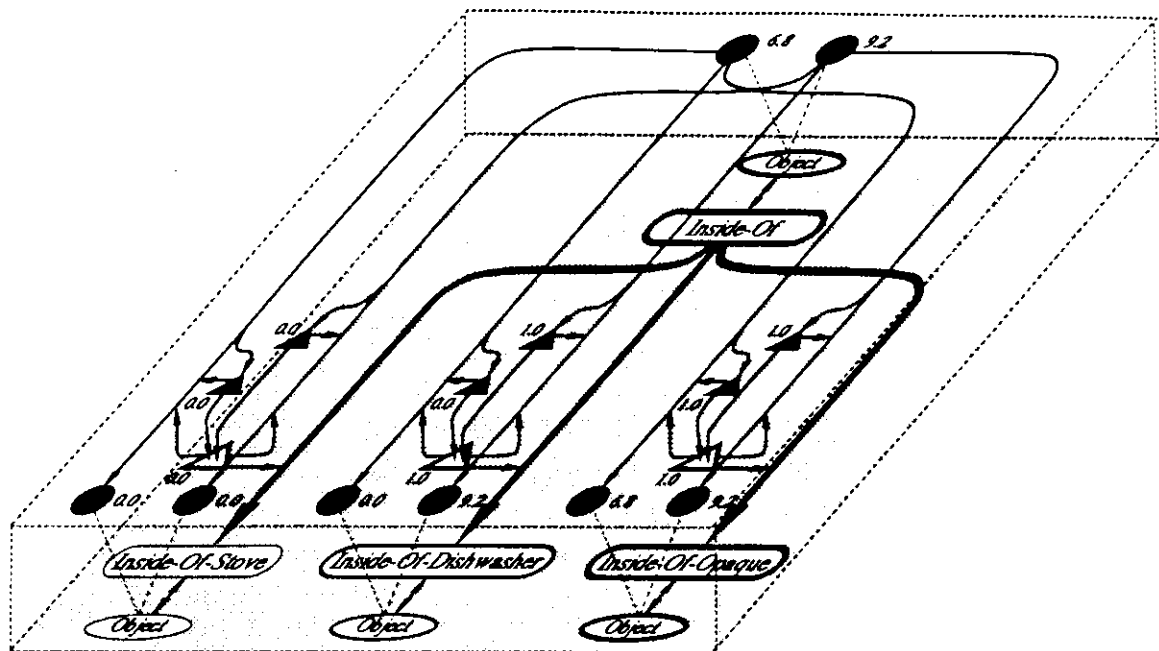


Figure 16. Paths from binding nodes of *Inside-Of*'s *Object* to the corresponding binding nodes of its refinement frames are gated by connections from their frame *candidacy* nodes (outlined triangles on bottom plane). The weighted links allowing evidential activation to pass from *Inside-Of* to its refinement frames are also gated by their *candidacy* nodes. Individual binding paths are also gated by connections from their *binding constraint* nodes (solid triangles on top level). All connections to gated links must be active for activation to pass through.

3.3.2. Gating of Signatures and Evidential Activation

The previously described frame selection process of the network is performed by inhibitory gating on those links which allow propagation of signature and evidential activation from one frame to another. Activation is only allowed to pass from a frame to one of its related frames when its role-bindings match the candidate frame's selectional restrictions.

The nodes and connections providing this inhibitory gating are shown in Figure 16. Each frame has a separate *candidacy* node (outlined triangles on bottom plane) for every frame that it is related to. This node will be active (with an activation of 1.0) if the related frame is a candidate for interpretation, i.e. if each of the frame's role-bindings match the candidate's binding constraints. However, if any of the frame's binding constraints are violated, then its *candidacy* node will be inactive (activation of 0.0). A description of the structure of the network performing these constraint-matching calculations appears in the next section.

Figure 16 shows the separate *candidacy* node for each of *Inside-Of*'s refinement frames. When the activation of the network reaches the state of Figure 15a, with *Inside-Of*'s *Object* bound both to *Marijuana* (signature 6.8) and *Cooking-Pot* (signature 9.2), the associated *candidacy* nodes of *Inside-Of-Dishwasher* and *Inside-Of-Opaque* become active (activation 1.0 in Figure 16) to choose them as candidate refinement frames. *Inside-Of-Stove*'s *candidacy* node, however, remains inactive, since *Inside-Of*'s *Location* violates its *Stove* constraint (a *Dishwasher* is not a *Stove*).

A further constraint on the propagation of each *individual* signature is that the binding it represents must match the role's logical constraints. Therefore, each separate binding path has a *binding constraint* (BC) node (solid black triangle) which calculates whether the object bound to the binding node matches the candidate's constraints.

For a signature to pass to the corresponding binding unit on a related frame, *both* the frame's candidacy node and the binding constraint node on the individual signature path *must be active*. As shown in Figure 16, each binding propagation link is gated by both its BC node and its frame's candidacy node. The link is conjunctive, as in the sigma-pi units described in [Rumelhart *et al.*, 1986].

The weighted link that allows evidential activation to pass through to the frame's conceptual node from the related frame's conceptual node is also gated by its candidacy node (Figure 16). Because of this, a related frame whose role-bindings do not match the frame's selectional restrictions will not provide evidence for it.

In Figure 16, both Marijuana and Cooking-Pot match the Physical-Object constraint on Inside-Of-Opaque's Object role, so both of its BC nodes are active (1.0). Since their candidacy and BC gates have an activation of 1.0, the signatures pass through the conjunctive link to instantiate Inside-Of-Opaque's Object binding nodes, as shown. Evidential activation is allowed to pass through.

The same is true of the signature representing Cooking-Pot (9.2) for Inside-Of-Dishwasher (on its right signature path). Marijuana (signature 6.8), however, is not cleaned in a dishwasher, so its BC node (on the left signature path) stays inactive (0.0). The gate on that path therefore remains closed, and Marijuana is not inferred as a possible Object to be cleaned in Inside-Of-Dishwasher. Inside-Of-Dishwasher's overall binding constraints are matched, however (since Cooking-Pot matched the constraints), so its candidacy node is active and evidential activation is allowed to pass through.

Finally, because Inside-Of-Stove's binding constraints were violated (a Dishwasher is not a Stove, the constraint on its Location), its candidacy node has an activation of 0.0. It therefore receives none of Inside-Of's evidential activation or signature bindings, and so it remains uninstantiated — without possibility of being considered as the current refinement of Inside-Of.

3.3.3. Binding Constraint Structure

The binding constraint nodes calculate whether or not a signature matches the selectional restrictions on a role. The original knowledge base definitions that construct the network (i.e. Figure 4) define a type (or types) of concept(s) that may be bound to each individual role. Examples are that Inside-Of-Dishwasher's Location role can only be filled with something that *is-a* Dishwasher, and that its Object role must be a Cooking-Utensil or something that *is-a* Cooking-Utensil, such as a Skillet or a Cooking-Pot (since they are the things that are cleaned in dishwashers).

In order for the network to decide whether a signature matches the binding constraints on a role, it must be compared to each of the signatures that form the role's *legal* bindings. Figure 17 shows how this is done for the BC node on the path from binding node B1 (bound to Cooking-Pot, signature 9.2) of Inside-Of's Object role to the corresponding binding node of Inside-Of-Dishwasher's Object.

In Figure 17, there are three concepts that Inside-Of-Dishwasher's Object can be bound to: (1) Cooking-Utensil, (2) Cooking-Pot, and (3) Skillet (because both Cooking-Pot and Skillet *is-a* Cooking-Utensil). To calculate whether B1 is bound to one of these three, there are three *comparator* nodes, C1, C2, and C3 (squares with an '=' inside). Each comparator node has two inputs to match: one candidate signature from the binding node, and one signature from the logical

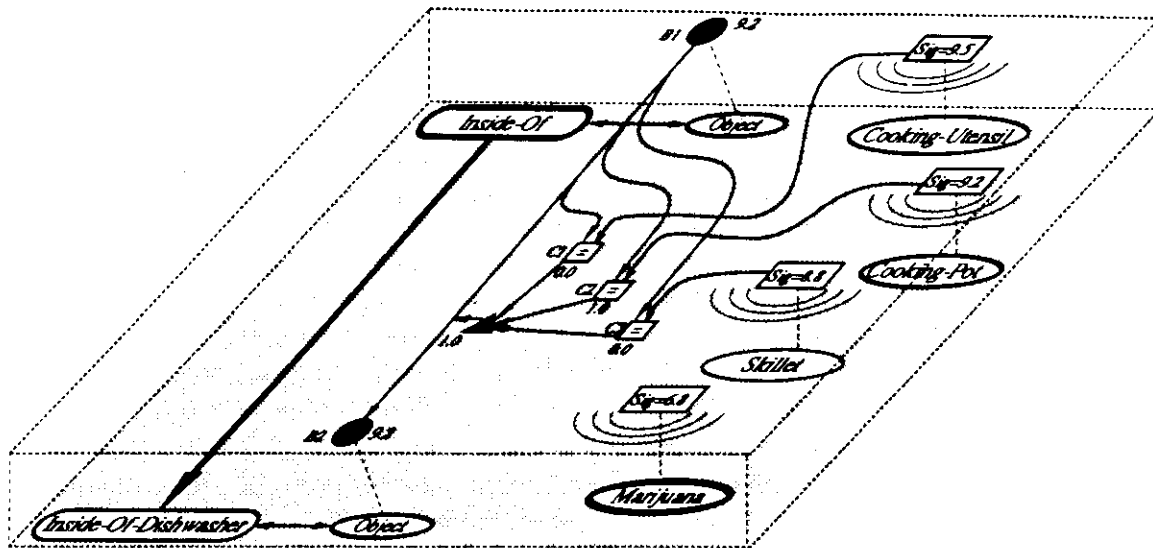


Figure 17. Figure illustrating how the binding constraint node (solid black triangle) for the path from one of the binding nodes of *Inside-Of*'s *Object* role to the corresponding binding node of *Inside-Of-Dishwasher*'s *Object* calculates its activation. The rectangular nodes with an "=" inside (C1, C2, and C3 on the top plane) are *comparator nodes*, which have an activation of 1.0 iff the signature activations from both incoming links are equal, 0.0 otherwise.

constraint. A comparator node will be active (activation 1.0) if the two input activations (signatures) are the same, inactive (activation 0.0) otherwise. Comparator nodes are actually made up of three nodes having small thresholds and binary outputs of 0.0 or 1.1. The first detects if activation A is greater than activation B (by having a weight of 1.0 from A and -1.0 from B). The second detects if activation B is greater than activation A (by having a weight of -1.0 from A and 1.0 from B). The third node is the actual comparator output node, and is active as long as neither A is greater than B nor B is greater than A (by having a self bias of 0.5 and weights of -1.0 from both the first and second nodes). Because of this, there is actually some leeway on the comparator nodes allowing matches even with a small amount of noise, since the two activations will match as long as the activations are within the thresholds on the comparator nodes.

Comparator node C1 has an input from binding node B1 and the signature node of *Cooking-Utensil*, so directly calculates whether or not B1's signature matches that of *Cooking-Utensil*. It does not in Figure 17, so C1 is inactive (activation 0.0). Comparator C3 also fails in this case, since it compares B1 to the signature of *Skillet*. Comparator C2, on the other hand, compares B1's activation to the signature of *Cooking-Pot*. *Cooking-Pot* is indeed the concept that is bound to binding node B1, so both of C2's inputs have the same activation (9.2), and C2 produces an activation of 1.0. This activation propagates to the BC node that gates the path from B1 to B2. With this BC node active (and *Inside-Of-Dishwasher*'s candidacy node also active), the signature of binding node B1 is clear to propagate past the BC gate to B2.

An equivalent structure (of comparator nodes) calculates the activation of the BC node for the other binding node of *Inside-Of-Dishwasher*'s *Object*. However, since *Marijuana* is the signature on the corresponding binding node (Figure 16), none of its corresponding (C1-C3) comparators will find that it matches *Cooking-Utensil*, *Cooking-Pot*, or *Skillet*, and so will all remain inactive. The path's BC node will have an activation of 0.0, thus blocking the (logically impossible)

signature of Marijuana from propagating to Inside-Of-Dishwasher's Object. Consequently, Cooking-Pot is inferred as an object that might be cleaned, but Marijuana is not.

Besides calculating when individual bindings match the constraints on a role and should be allowed to propagate on through, the network must also be able to calculate when the role-bindings of a related frame match *all* of a frame's selectional restrictions. Even though a Cooking-Pot matches the selectional restriction on the Object of Inside-Of-Stove, Inside-Of-Stove's overall selectional restrictions are violated because a Dishwasher is not a Stove, as expected on its Location role. As Figure 16 suggests, the individual BC nodes drive this calculation by having inputs into the frame's overall candidacy node. Though not shown in Figure 16, a simple intervening structure of nodes and inhibitory links for each candidacy node calculates this and causes it to be active (with activation 1.0) if all of the selectional restrictions are matched, or inactive (activation 0.0) if any of them are violated.

3.3.4. Virtual Evidential Structure From Signature Bindings

The connectivity of ROBIN's conceptual layer (e.g. Figure 10) and that of previous structured disambiguation models encodes static relations between frames and their roles' prototypical fillers without making any assumptions about the *actual* role-bindings being processed at any given time. This structure causes each frame node to accumulate an amount of evidential activation that is appropriate *if the actual role-bindings are not known*.

However, in ROBIN, the actual role-bindings usually *are* known (as signatures). Inferences and evidence from actual frame instantiations can be quite different than those from unfilled frame instantiations. For example, if the network knows that somebody is smoking something, then the network can "guess" that the actual object being smoked is either Cigarette or Marijuana by providing evidential activation to its prototypical fillers. However, if John is smoking a Cigarette, then he is not smoking Marijuana. It doesn't matter that Marijuana is something that can be smoked — if Cigarette is the actual filler, then it, and not Marijuana, should receive evidence from that smoking. To take this into account and limit each conceptual node to evidence accumulation from the actual instances in the network, connections between concepts in ROBIN are also gated to control the flow of activation based on their signature bindings. Thus, the actual bindings in a given context cause the network to have a *virtual structure* that combines evidence *as if the network was hand-built for those particular instances*, without modifying the hard-wired "default" structure of the network.

As seen in Figure 16, the candidacy nodes that gate the spread of evidential activation between frames are part of this virtual evidential network structure, since they stop evidential activation from being received by frames whose selectional restrictions have been violated (such as Inside-Of-Stove in Hiding Pot). Because of this, the network acts as if it had been hand-wired *without* any connections from the frame whose role-bindings violate the selectional restrictions of a (normally) related frame.

The second part of virtual structure in ROBIN channels evidential activation from bound concepts to the roles that they fill, and vice versa. As described previously, when the filler of a role is unknown, the role receives activation from its prototypical filler or fillers through its Prototypes branch (Figure 10). At the same time, the unbound role provides evidential activation for its prototypical fillers. However, when a role's binding unit is instantiated with signature activation representing a binding, its Prototypes branch is inhibited so that the role does not receive any activation from or provide any activation to its prototypical fillers. Instead, the role receives activation from the actual concept (or concepts) bound to the role, as represented by the signature binding. To complete the feedback loop, the role provides activation to the concept bound to the role (rather than the prototypical filler). This insures, for example, that Marijuana receives no evidence from the smoking frame when somebody is smoking a Cigarette. The network structure that gates the

spread of evidential activation based on the actual signature instantiations is all part of the network of connectionist units, and is similar to the structure enforcing selection restrictions.

22
28
3

4. DETAILED EXAMPLES

ROBIN has been implemented in the DESCARTES connectionist simulator, which allows the flexible simulation of structured heterogeneous networks [Lange *et al.*, 1989] [Lange, 1990]. In this chapter we give a detailed description of the spreading-activation process for disambiguation and reinterpretation of the **Hiding Pot** example and show an example of how the structure enforcing selection restrictions within the network controls crosstalk.

4.1. Processing of Hiding Pot

As a detailed example of how evidence is combined to perform disambiguation and reinterpretation, consider the processing of input for the sentence "*John put the pot inside the dishwasher because the police were coming*" (**Hiding Pot**). As described in chapter 3.1.4, a phrase is presented to the network by clamping the lexical concept nodes for each of its words to a high activation value (1.0) and by clamping the appropriate binding nodes to the signatures given by the phrase's syntactic bindings. To roughly simulate the sequentiality of reading, the network is allowed to iterate for 10 cycles between the presentation of each word/role-binding. The clamping for the first phrase of **Hiding Pot** is thus the same as that described for Figure 7, except that the lexical node for "*John*" is clamped on cycle 1, the lexical node for phrase <S "put" DO "inside" IO> is clamped on cycle 11, and so on.

Figure 18 shows the levels of evidential activation through time for most of the relevant frames during the spreading-activation process for **Hiding Pot**. The first thing that happens is that **Transfer-Inside** and **Inside-Of** become activated, in order, as they are inferred from the phrase (Figure 18a). Note that it takes approximately 19 cycles for activation to propagate from one frame to another through the input branch nodes and the nodes that enforce selectional restrictions. During this time, the three ambiguous meanings of "*pot*" are differentiated only by the strength of their weights from the "*pot*" node, since they all can be put inside of things. In this particular network, **Planting-Pot** has the strongest weight from "*pot*", so it is winning up until about cycle 60 (Figure 18e).

At about cycle 50, activation and signatures from **Inside-Of** reach its refinements (Figure 18b). As described earlier, the selectional restrictions in the network stop **Inside-Of-Stove** and most of **Inside-Of**'s other refinements from receiving either evidential or signature activation, since a pot inside a Dishwasher does not match their roles' binding constraints. Just as important is that the signature for **Cooking-Pot** is the only one that reaches **Inside-Of-Dishwasher**'s Object (Figure 16). Because of this, as **Inside-Of-Dishwasher** increases in activation, **Cooking-Pot** receives evidence from it (that **Marijuana** and **Planting-Pot** do not), causing **Cooking-Pot**'s activation to start rising around cycle 65. Because **Inside-Of-Dishwasher** in turn receives activation from the virtual binding of **Cooking-Pot**, a small feedback loop is created between the two that pushes both higher, including the newly instantiated **\$Dishwasher-Cleaning** frame (Figure 18c). By cycle 100, **Cooking-Pot** is clearly dominant over the other interpretations of "*pot*", and would be chosen as the binding throughout the network were the simulation to be halted at this point.

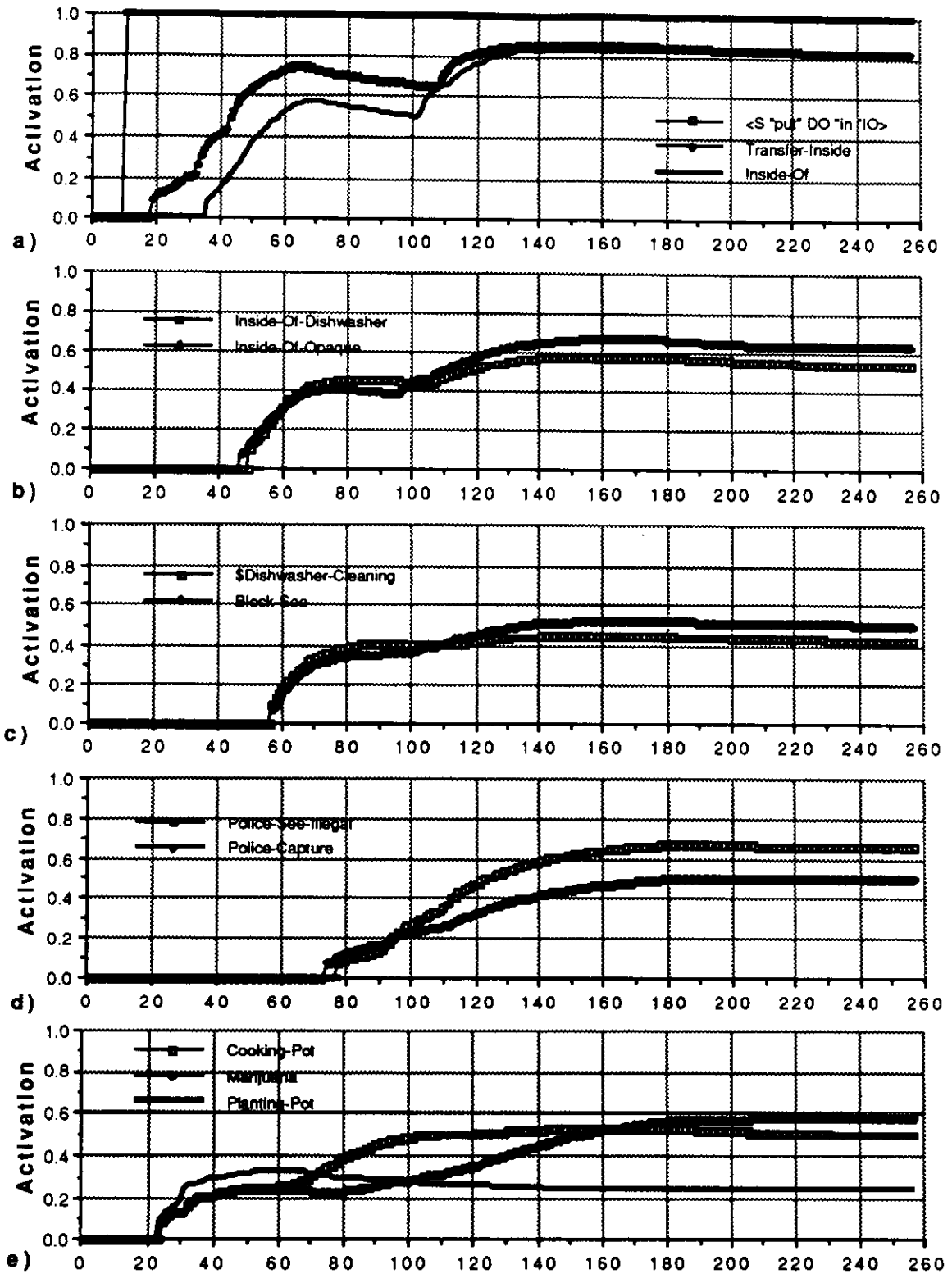


Figure 18. Evidential activations in network after presenting input for Hiding Pot.

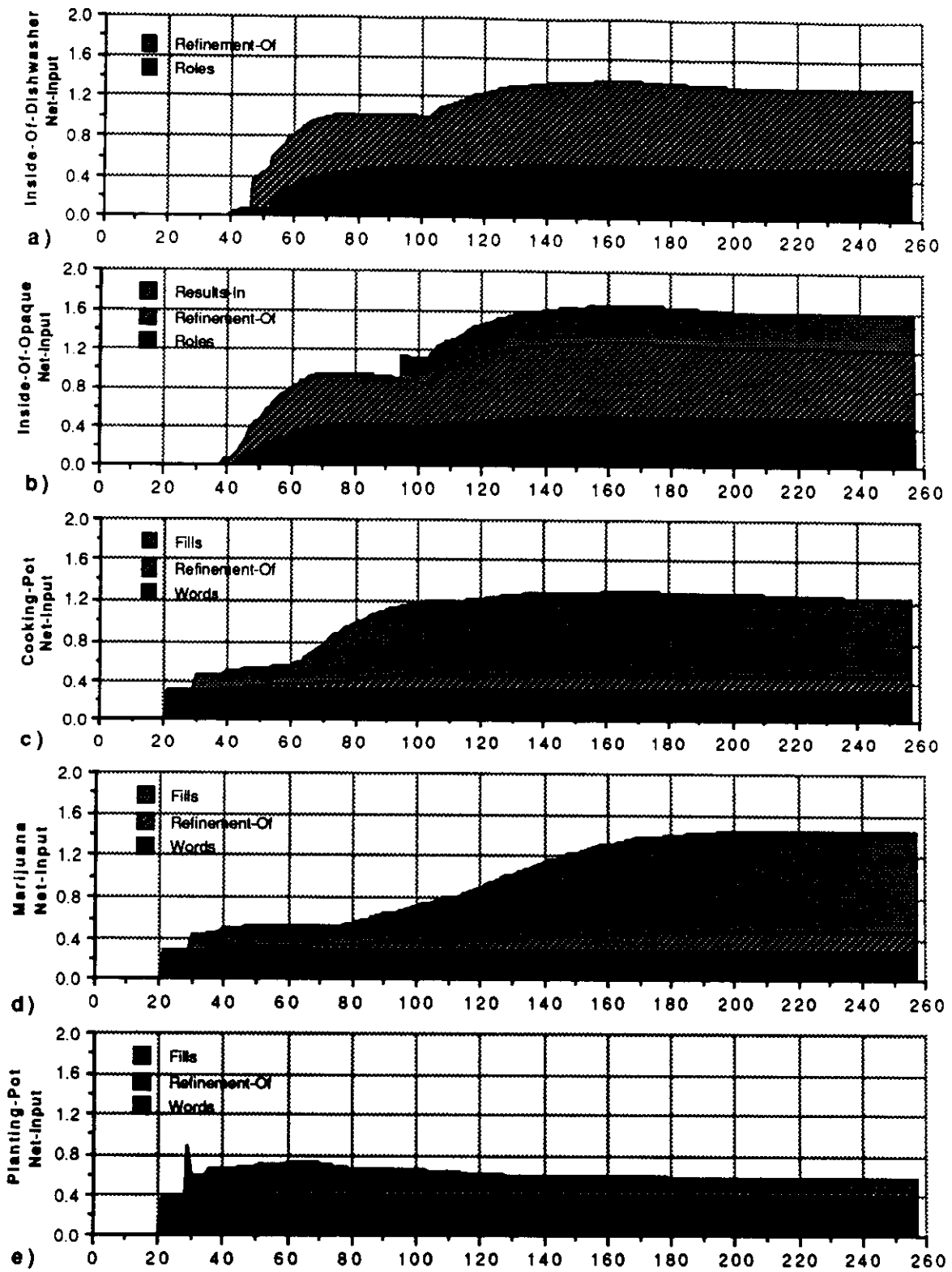


Figure 19. Activation of input branches for the evidential nodes of ambiguous frames in Hiding Pot.

Inside-Of-Dishwasher also becomes more highly-activated than its competitor Inside-Of-Opaque during these cycles, but its advantage is much smaller. The reason for this can be observed in Figure 19, which details the activations of the input branches through which the frames receive their input¹. Cooking-Pot quickly becomes more active than Marijuana and Planting-Pot because of the unique role that it fills in the highly-specific dishwasher cleaning frames, so that its Fills input branch provides it with much more activation than theirs do in cycles 60 to well over 100 (Figures 19c-e). On the other hand, the added boost that the ascension of Cooking-Pot provides to Inside-Of-Dishwasher is averaged with the activation from its other role fillers. The advantage of Inside-Of-Dishwasher's Role input branch over Inside-Of-Opaque's between cycles 65 and 100 is thus relatively small (Figures 19a & b), since Inside-Of-Dishwasher and Inside-Of-Opaque share most of the same role-bindings (both were planned by John, have the Location of a Dishwasher, and have Cooking-Pot as a potential Object).

While this is occurring, input for **Hiding Pot** continues to be presented to the network. At cycle 50, the lexical node "*police*" is clamped, as is its signature binding and the evidential node for phrase <S "were coming"> at cycle 60. Inferences are made by propagation of signature and evidential activation through Transfer-Self, Proximity-Of, and See-Object, until two things happen. First, Police-See-Illegal gets instantiated from See-Object (which has itself received evidential activation through inferences from both phrases of the sentence). Police-See-Illegal accepts only objects that are Illegal-Objects, so the network's selectional restrictions allow only Marijuana through, giving it an opportunity to receive unshared activation from Police-See-Illegal and eventually Police-Capture. Thus the activation of Marijuana's Fills input branch increases at about cycle 90 (Figure 19d), with the activation of Marijuana itself following closely behind. The second important thing to happen is that the inferences from the second phrase reach Inside-Of-Opaque. This allows Inside-Of-Opaque to start receiving activation from Block-See (Figure 18c) through its results-in branch, causing an immediate leap in Inside-Of-Opaque's incoming activation at cycle 93 (Figure 19b)².

As time goes on, feedback through the virtual structure between the Police-Capture frames and their Marijuana filler enables Marijuana to overtake Cooking-Pot (cycle 160), thus lexically reinterpreting the meaning of the word "*pot*". Similarly, the new evidence from the Transfer-Self ↔ Proximity-Of ↔ See-Object ↔ Block-See path causes Inside-Of-Opaque to accumulate more evidential activation than Inside-Of-Dishwasher, thus making a pragmatic reinterpretation of John's reason for putting the pot inside the Dishwasher. The final interpretation of **Hiding Pot** is the most highly-activated path of frames in the network (Figure 20), which at stability in-

¹Recall that each conceptual node's activation function includes part of their previous activation, while being short-circuited (normalized) by the global inhibition nodes. Because of this normalization, the activation of each conceptual node is generally less than the sum of its net inputs, as can be seen by comparing nodes' activations in Figures 18b and 18e with their net inputs in Figure 19.

²In addition to propagating signatures representing role-bindings, ROBIN propagates the signature of the frame the bindings originally came from, i.e. the original sentence's phrase. Structure similar to that enforcing selectional restrictions ensures that each frame only receives evidential activation from at most one frame with the same phrase origination, preventing potentially ruinous effects from circular self-feedback loops. Here, for instance, Inside-Of-Opaque initially receives no evidential activation from Block-See, since the sole evidence for Block-See came through Inside-Of-Opaque itself (from P2). By cycle 93, however, Block-See has evidence in its own right from P2 (through See-Object), so its evidential activation can start reinforcing Inside-Of-Opaque.

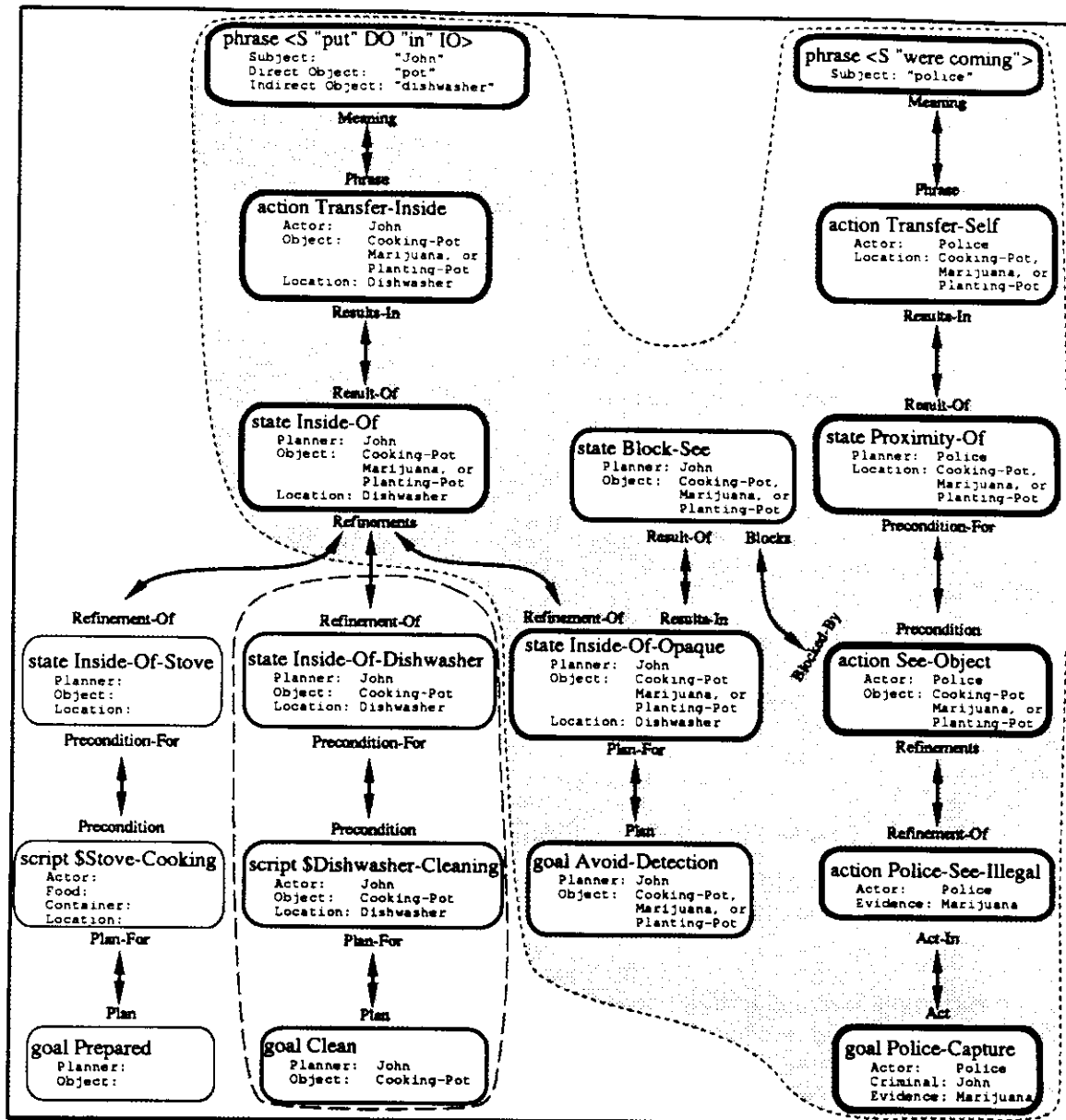


Figure 20. Overview of a small portion of a ROBIN semantic network showing inferences dynamically made after network settles after presentation of input for phrases P1 and P2 of **Hiding Pot**. Darkly shaded area indicates the most highly-activated path of frames representing the most probable plan/goal analysis of the input. Dashed area shows the discarded dishwasher-cleaning interpretation.

cludes the Inside-Of-Opaque path and the Police-Capture frames, representing the interpretation that John was trying to avoid the detection of his Marijuana from the police by hiding it in an opaque dishwasher.

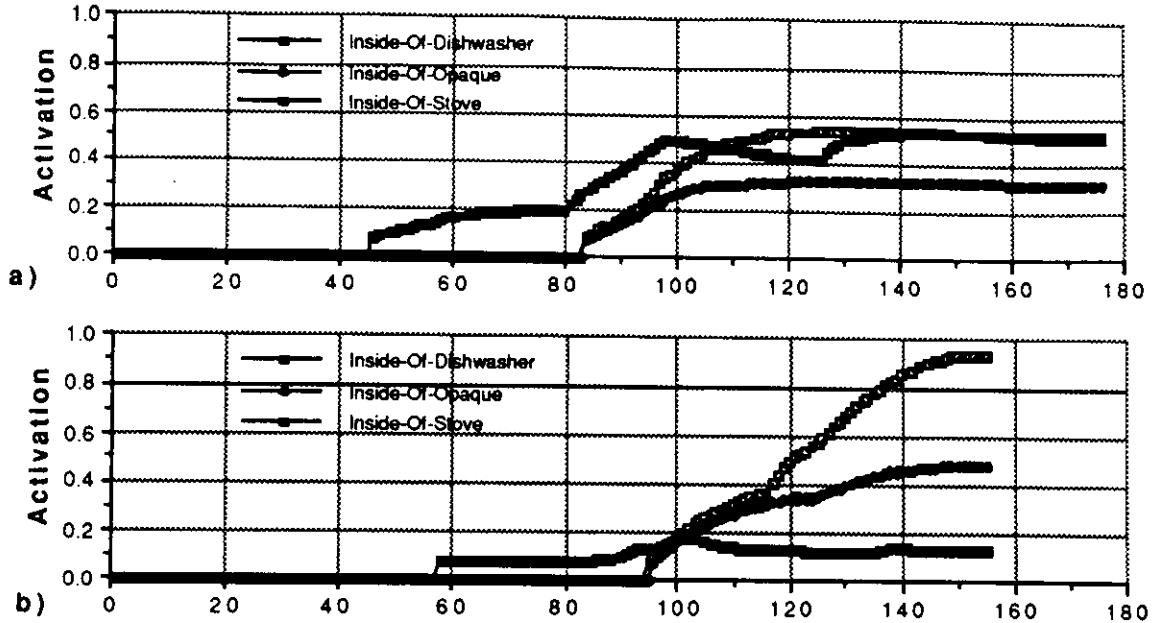


Figure 21. Activations of refinements of Inside-Of after being presented with input for “After Bill put the omelette on the stove, he put the bowl inside the dishwasher.” For a) a ROBIN network without structure enforcing selectional restrictions, and b) a ROBIN network with selectional restrictions.

4.2. The Effect of Selectional Restrictions

ROBIN’s network structure that controls the spread of activation based on selectional restrictions is crucial to controlling crosstalk from logically unrelated inferences. As an example, consider the sentence:

“After Bill put the omelette on the stove, he put the bowl inside the dishwasher.”
(Cook-and-Clean)

The most likely interpretation of **Cook-and-Clean** is that Bill put the bowl in the dishwasher so that he could eventually clean it after cooking his omelette. In the network, the instance of Inside-Of-Dishwasher with Bill as the Actor and Bowl as the Object should thus be the winning refinement of Inside-Of. However, if Inside-Of-Stove is allowed to receive activation from Inside-Of even though its binding constraints are violated, there is a good chance that it could become more activated than Inside-Of-Dishwasher, due to the combined activation from Inside-Of and Stove. As can be seen in Figure 21a, this in fact happens in a network without the structure enforcing selectional restrictions. This “dumb” network has arrived at the decision that Bill was trying to cook something in the bowl when he put it in the dishwasher.

Of course, ROBIN could use the solution of marker-passing systems and evaluate the paths returned by the propagation of signatures. This evaluation would reveal that Inside-Of-Stove’s selectional restrictions had been violated (since the Location was actually a Dishwasher). ROBIN could then reject that path and choose the next most-highly activated one, which in this case is the correct Inside-Of-Dishwasher path.

However, one of ROBIN's main goals is to avoid the use of a separate path evaluation mechanism by returning a single, logically-possible inference path. And as can be seen in Figure 21b, this is exactly what happens for **Cook-and-Clean** in a normal ROBIN network with the structure enforcing selectional restrictions. With the selectional restrictions, **Inside-Of-Stove** still becomes partially activated because of evidence from its prototypical fillers (particularly **Stove**). But since it does not receive any signatures or evidential activation from **Inside-Of**, its activation does not come close to competing with **Inside-Of-Dishwasher**. The network thus settles on the correct interpretation of the sentence.

5. DISCUSSION AND CONCLUSIONS

ROBIN's inferencing, plan/goal analysis, schema instantiation, disambiguation, and reinterpretation abilities have been successfully tested on **Hiding Pot** and a number of other episodes, in two domains, using syntactically preprocessed inputs of one and two sentences in length. Table 2 lists some of the other episodes that ROBIN is able to disambiguate and analyze in terms of their plan/goal structure.

The **Hiding Pot** example's knowledge base currently has 121 conceptual frames defined (18 of which are shown in Figure 5), with a total of 133 different roles (see Appendix A). The average number of roles per frame is less than that of Figure 5 because 33 object frames (e.g. Physical-Object, Cooking-Pot, John, etc.) and 20 simple lexical entries (e.g. "pot" and "police") have no roles defined other than their relations to other frames. The network encodes 208 rules in the form of corresponding roles over which inferences can be made by propagation of signatures.

The network that is constructed from the **Hiding Pot**'s knowledge base (Appendix A) has a total of approximately 20,000 nodes, the vast majority of which are used to calculate selectional restrictions (see discussion in chapter 5.1.1). Each role in the network has three binding nodes. Because of this, there are the equivalent of 399 variables in the network (3 binding nodes per role x 133 roles) and 624 binding path "rules" (3 x 208 rules) over which inferences can be made.

The network typically takes between about 100 and 250 cycles to generate all candidate inference paths and settle into a stable state in which a single inference path is most highly-activated (e.g. Figure 19). The number of cycles generally required for reaching quiescence has remained in approximately that range for all sizes of the network tested. This is primarily true because the gating and selectional restrictions within the network's structure stop activation from spreading to the (sometimes) large areas of the network that are logically unrelated to the input.

5.1. Future Work

In the future, there are six main areas that we would-like to explore: (1) signatures as distributed patterns of activation, (2) realization of signatures as temporal frequencies, (3) the ability to handle embedded role-bindings, (4) increasing the network's capacity, (5) formation of long-term episodic memory, and (6) the addition of lexical information to the networks.

5.1.1. Signatures Using Distributed Representations

Currently, each signature is a single arbitrary activation value that uniquely identifies its concept. Large models could conceivably have thousands or hundreds of thousands of separate concepts that they could recognize (such as Marijuana, Cooking-Pot, Catfish, Guppy, John, John-Wayne, John-Kennedy, etc). It is untenable to expect a single binding node to have enough precision to accurately distinguish between such a large number of signatures¹.

A better solution is that each signature be a *distributed* pattern of activation which uniquely-identifies its concept, as proposed in [Lange & Dyer, 1989]. As shown in Figure 22, distributed signatures would be propagated for inferencing over paths of binding *banks* in exactly the same way as ROBIN's current single-valued signatures. Similar concepts would have similar distributed patterns of constant values as their signatures, so that each signature would carry some semantic meaning. A first pass at this might entail the use of microfeature-like patterns, as in the distributed model of [McClelland & Kawamoto, 1986], but it would be preferable to have the signature patterns learned over time, as done by the model of [Miikkulainen & Dyer, 1988, 1989].

¹The normal coding capacity of connectionist elements is usually in the range of 1-5 bits.

"John smoked the pot."	"pot" = Marijuana, "smoke" = \$Burn-And-Inhale
"Bill smoked the meat."	"smoke" = \$Smoke-Food
"Jerry put the pot on the stove."	"pot" = Cooking-Pot
"Ron put the pot on the stove. He picked it up and smoked it."	"pot" = Cooking-Pot, "put" = \$Stove-Cooking → "pot" = Marijuana "put" = \$Light-Object
"John put the flower in the pot, and then watered it."	"pot" = Planting-Pot, "put" = \$Grow-Potted-Plant
"Cheech watered the pot, but the police saw him, so he was arrested."	"pot" = Planting-Pot → "pot" = Marijuana-Plant
"Jerry grew the pot."	"pot" = Marijuana-Plant
"The CIA searched for bugs." [Granger <i>et al.</i> , 1986]	"bugs" = Microphone, goal = Remove-Listening-Device
"Safeway searched for bugs." [Granger <i>et al.</i> , 1986]	"bugs" = Insects, goal = Remove-Health-Hazard
"Fred proposed to Wilma. Wilma began to cry. Wilma was saddened by the proposal" [Eiselt, 1987]	cry-tears = Happy-Event → cry-tears = Sad-Event

Table 2. Examples ROBIN handles using activation clamping from syntactically pre-processed input.

One of the most important results of using distributed signatures would be a vast simplification of the network structure calculating whether individual signature bindings match a role's selectional restrictions. Because signatures currently carry no semantic meaning, the binding constraint nodes must have a *separate signature comparator node* for each and every one of the concept's signatures that are legal role-bindings. The number of comparator nodes required to calculate a single binding constraint can vary from one (a Dishwasher constraint on *filler* needs to check whether the signature matches Dishwasher) to extremely many (any person known to the system can match a Human constraint on a role, so there must be a separate comparator node for each).

This is clearly not an acceptable solution for large networks. However, if signatures are distributed patterns of activation that are *similar for similar concepts* and that themselves carry semantic information, then the entire structure of comparator nodes for a role could be replaced with a bank of nodes that does a simple *similarity threshold* between the signature binding and the distributed signature of the logical binding constraint. Another possibility is that the binding constraint nodes could be a small distributed ensemble of nodes *trained* to recognize the constraints that a role has on its filler.

5.1.2. Signatures as Temporal Frequencies

Another realization of signatures that we are currently exploring utilizes the time dimension, in which signatures are uniquely-identifying *frequencies* of output spikes generated by artificial neural oscillators. We have already illustrated how *signature frequencies* representing bindings can be propagated across a network for inferencing by phase-locking of relaxation oscillators [Lange *et al.*, in press]. Linkage of distant oscillators through common output frequencies represents shared role-bindings and inferences. [Tomabechi & Kitano, 1989] have also suggested the use of frequency modulation of oscillator pulses for this task.

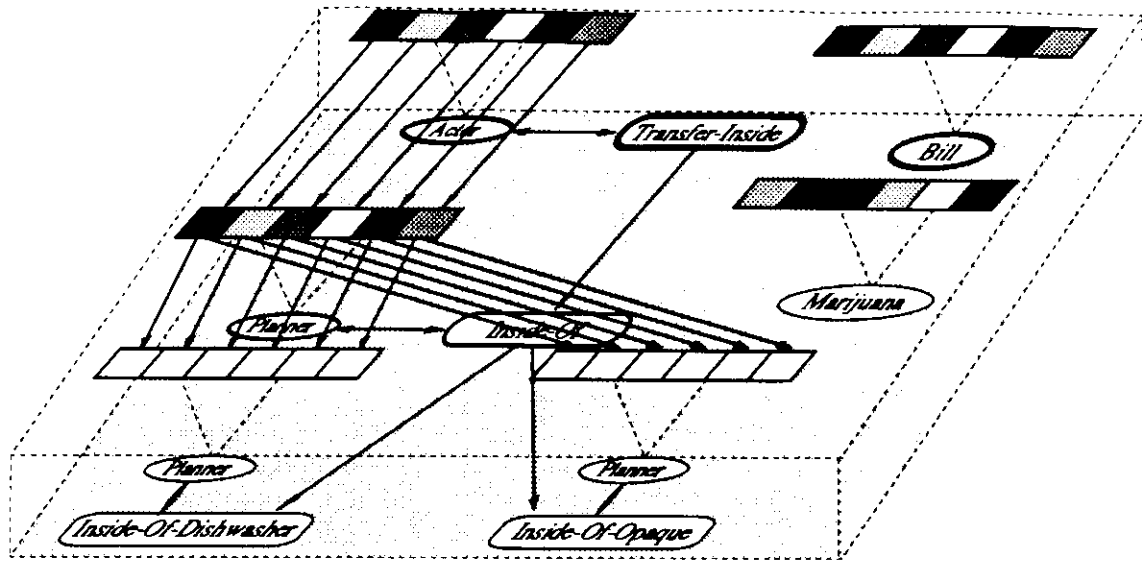


Figure 22. Possible future use of *distributed* signatures, where each signature is a unique pattern distributed over a *bank* of nodes. Here each signature or binding bank is made up of six nodes, with increasing levels of activation represented by increasing darkness of shading (ranging from white = 0 to black = 1). Shown is the (desired) state of the network after Bill's distributed signature has propagated from the binding bank of Transfer-Inside's Actor to the binding bank of Inside-Of's Planner, but before reaching the Planner banks of Inside-Of-Dishwasher and Inside-Of-Opaque.

5.1.3. Embedded Role-Bindings

Using signatures of pre-existing concepts, ROBIN can create and infer novel network instances. However, ROBIN currently cannot dynamically generate and propagate *new* signatures for one of these instances. This ability is crucial for recursive structures, such as in: "John told Bill that Fred told Mary that..." Here each Object of the telling is itself a novel frame instance not having a pre-existing signature. We are currently exploring a solution in which the signatures of the newly-instantiated frames themselves are propagated, a solution that is analogous to that of [Ajjanagadde, 1990]. Until a solution for embedded signatures is found, ROBIN's inferencing capabilities will be somewhat limited in comparison to symbolic rule-based systems¹.

5.1.4. Network Capacity

ROBIN currently only has the capacity to understand examples of from one to three sentences in length, such as **Hiding Pot** and **Marriage**. A major limitation of the model as described is that each frame can only have one instance at any given time, since binding units can only hold a single signature activation at once. Because of this, ROBIN cannot represent or interpret any stories involving two different seeing or eating events, for instance. We are currently exploring a solution in which each frame will have more than one set of conceptual and binding units, each capable of holding a separate dynamic instantiation.

However, even with the capacity to hold multiple instances of each frame, the network's capacity will still be limited by the fact that stories' interpretations are represented as activation across a fi-

¹But not limited in comparison to other structured connectionist models of disambiguation, which cannot handle even simple role-bindings.

nite network. The evidential activation of original parts of a story that are not bolstered by new context will decay away and be lost as time progresses. This is not a problem in marker-passing networks, since they can simply store the generated inference paths incrementally in a separate symbolic buffer¹. Of course, we could use such a solution for ROBIN, but we would prefer to find a purely connectionist solution. Part of the question will be exploring how much of a story can or should be held by the short-term memory of activation in the network before it decays away. A couple of sentences? A couple of paragraphs? On the order of psychological seconds, minutes, or hours?

5.1.5. *Formation of Long-Term Episodic Memory*

Signatures allow ROBIN to create novel network instances over its pre-existing structure, but the activation of these instances is transient. Over time, repeated instantiations should cause modification of weights and recruitment of underutilized nodes [Diederich, 1990] to alter network structure. Possible methods of storing the inferred instances in long-term episodic memory by some kind of distributed learning mechanism must also be explored, likely in conjunction with the use of distributed signatures.

5.1.6. *Lexical Information*

ROBIN does not currently address the problem of deciding upon the original syntactic bindings, i.e. that "pot" is bound to the Object role of the phrase. Rather, ROBIN's networks are given these initial bindings and use them for high-level inferencing. To handle natural language input entered as text, the network must somehow contain and use syntactic and phrasal information to create the initial role-bindings that ROBIN is currently given by hand.

5.2. Comparison to Related Connectionist Models of Variable Binding and Inferencing

There are currently a very limited number of connectionist models besides ROBIN that have attempted to emulate the symbolic abilities of variable binding and rule firing.

5.2.1. *Distributed Connectionist Models*

Distributed connectionist models represent knowledge as patterns of activation across nodes, rather than the single unit representation of individual concepts used in structured networks such as ROBIN. Each of the distributed connectionist models described below uses the energy minimization metaphor to "settle" into individual variable bindings or rule firings.

DCPS is a distributed connectionist production system described by [Touretzky & Hinton, 1988] that uses *coarse codings* to represent variable bindings. Each node in DCPS's working memory has associated with it a receptive field that can represent many possible Frame-Slot-Filler "triple" combinations. When a variable binding exists in the memory, all of the nodes with a receptive field that contains that binding triple (approximately 28 out of their working memory of 2000 nodes) are activated. Multiple triple bindings are represented by superimposing their receptive fields.

DCPS also uses coarse-coding to represent rules of the form:

$$(\text{=x A B}) (\text{=x C D}) \implies +(\text{G =x P}) -(\text{=x R =x})$$

where the capital letters are constants and =x is a variable. When the left-hand side of a rule is matched, the triples on the right-hand side of the rule are either added (+) or deleted (-) from

¹Though they still face the problem of determining when to remove individual markers from the network.

working memory. The rules in DCPS are limited to two clauses on the left-hand side of the rule, with every rule having only a single variable that must appear as the first element of both clauses. Another restriction is that there can never be more than one rule with one variable binding that matches the contents of working memory at any given time.

To execute the rule firing cycle, DCPS first performs energy minimization to find a rule whose left-hand side matches two of the triples in the working memory. Once a rule is settled upon, gated connections from the rule space add and remove receptive fields from the working memory to “fire” the right-hand side of the rule. The cycle is then repeated and another production is matched, just as in a serial computer.

Another distributed connectionist approach is CRAM [Dolan, 1989], a hybrid natural language processing system in which the parsing modules are symbolic, but in which the memory and binding modules use *conjunctive coding*. In this distributed representation scheme, a cube of nodes is allocated to encode Frame-Slot-Filler triples by superimposing the binary pattern of each triple element across a dimension of the cube. This method has been generalized to scalar values by representing the superimposition as the outer product of two or more tensors [Derthick, 1988] and [Dolan & Smolensky, 1989].

Each of these models has successfully demonstrated that distributed connectionist models have the ability to represent and use explicit rules. Furthermore, their use of distributed representations allows their models to be damage-resistant and use far fewer units than needed in traditional localist networks that represent each potential fact with a single unit.

The primary problem with each of these distributed connectionist models is that although they *select* their rules through massively-parallel constraint satisfaction, they actually behave *serially* at the knowledge level, since they can select and fire only one rule at a time. This becomes a major problem when the tasks are complex and involve high-level inferencing. In natural language understanding and planning tasks it is generally necessary to explore many possible inference paths. Making conceptual connections between two or more facts effectively amounts to an intersection search which can quickly involve a very large number of rules. This has proven to be a debilitating problem to serial symbolic rule-based systems, and has in fact motivated a large amount of research in marker-passing models that are better able to approach these problems due to their massively-parallel symbolic mechanisms.

Because current rule-handling distributed models are serial at the knowledge level, they will continue to be plagued with many of the same problems that traditional symbolic rule-based systems face. ROBIN's structured networks, on the other hand, are able to fire many rules at once, dramatically decreasing the time required to “search” through the rule space to find inference paths connecting the inputs. Equally important is that with the propagation of signatures, ROBIN has no need to use (relatively slow) constraint satisfaction to select and fire its rules. ROBIN instead uses the network's smooth constraint satisfaction abilities to perform an even more difficult part of high-level inferencing: *selecting the most plausible of alternative inference paths and allowing for reinterpretation if contexts change*.

5.2.2. Ajjanagadde and Shastri's Structured Connectionist Model

The connectionist model which is most closely related to ROBIN [Lange & Dyer, 1988] is that proposed recently in [Ajjanagadde & Shastri, 1989]. Their model also solves the knowledge-level parallelism problem by being able to maintain and propagate multiple variable bindings in a structured network, but does so instead by using a multi-phase clock.

Their networks encode predicates and rules of the sort:

```
forall (x,y,z) (orderhit (x,y,z) => hit (y,z))  
  (If X orders Y to hit Z, then Y hits Z)
```

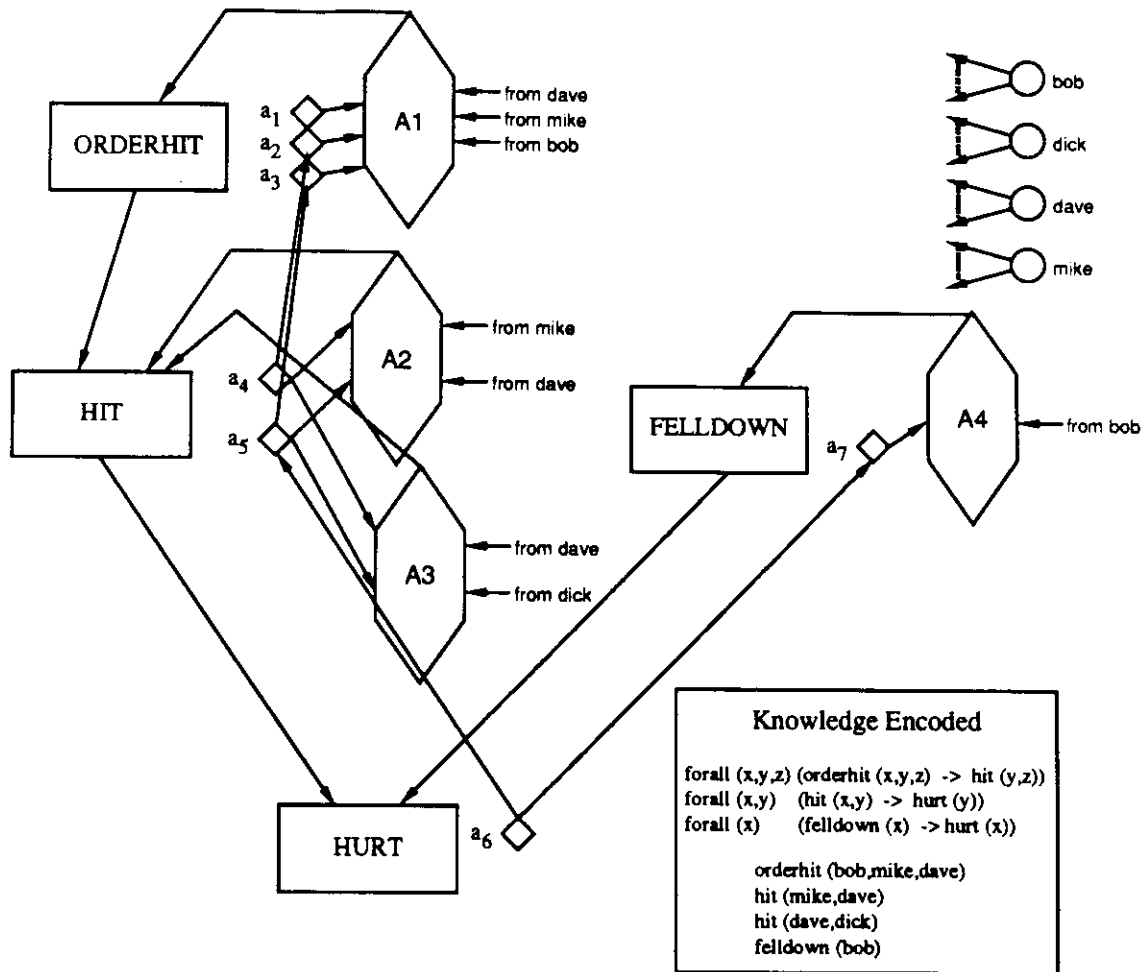


Figure 23. Example network from [Ajjanagadde & Shastri, 1989]. Diamonds a_1 thru a_7 are *arg*-nodes, while circles for bob, dick, dave, and mike are *const*-nodes.

The networks also store long-term facts with individual instance nodes, such as a `orderhit(dave,mike,bob)`. They then pose queries to the network which can only be answered by inferencing over rules like the above, such as the query `?hit(mike,bob)`.

In their model, each network cycle is broken up into a fixed number of phases. A variable binding is represented when an *arg*-node (analogous to ROBIN's binding nodes) is active on the same phase of the clock as the *const*-node (analogous to ROBIN's concept nodes) bound to it. As in ROBIN, there are connections between corresponding *arg*-nodes as defined by rules in the knowledge base. Figure 23 shows an example network illustrated in [Ajjanagadde & Shastri, 1989].

To provide the initial bindings to the network and pose a query, they allocate (by hand) a phase of the clock for each binding in the query. For example, to pose the query `?hit(mike,bob)` to the network of Figure 23, they activate the first *arg*-node of `hit` (a_4) and the *const*-node `mike` in the first clock phase, and activate the second *arg*-node of `hit` (a_5) and the *const*-node `bob` in the second clock phase. Because *arg* and *const*-nodes always become active on the phase of the clock that they were originally activated in, these bindings will hold indefinitely.

Once the bindings are set, they propagate over the paths between arg-nodes. In the first phase of the second cycle, for example, the activation of arg-node a_4 causes arg-node a_2 to become active. A_2 will then continue to become activated on every first phase. Because const-node mike is still active in the first phase, a_2 (the second argument of *orderhit*) is now also bound to mike, and the inference has been made. Similarly, in the second phase of the second cycle, arg-node a_5 causes a_3 to become active, so that both are bound to bob. They are thus able to propagate bindings across the network for inferencing.

Ajjanagadde and Shastri's model succeeds in illustrating an alternative mechanism for maintaining variable bindings in a structured connectionist network. Like ROBIN, their model can match and fire multiple rules in parallel, as opposed to the serial limitations (one rule at a time) of current distributed connectionist models.

The basic kinds of inferences that can be performed by ROBIN's propagation of signatures and by Ajjanagadde and Shastri's multi-phase clock seem to be about equivalent. Both mechanisms allow the binding of any previously known concept to any role in the network, and cause those bindings to be propagated along arbitrarily long inference paths defined by their knowledge base's rules.

One of the clearest advantages that Ajjanagadde and Shastri's model has over both distributed connectionist and traditional symbolic systems is its ability to perform deductive inference with extreme efficiency. Their model is in fact able to draw conclusions in time proportional to the length of the proof [Ajjanagadde & Shastri, 1989]. ROBIN does not aim for such optimal efficiency, instead using the constraint satisfaction process of the evidential portion of its networks to resolve ambiguities. However, if signatures were to be applied solely to the task of deductive retrieval that Ajjanagadde and Shastri's model handles, ROBIN could be stripped of its constraint-relaxation evidential network. In this case, propagation of signatures could also perform deductive retrieval in time proportional to the length of the proof. In fact, because a single cycle in ROBIN is functionally equivalent to a single phase of the clock in Ajjanagadde and Shastri's model¹, signature propagation would complete the proof a factor of p times faster than their model, where p equals the number of phases in their clock cycle.

One potential problem with using a phase-clock mechanism for variable binding (as opposed to signatures) lies in selecting the number of clock phases when there is sequential input. For example, in natural language processing systems new bindings are constantly being created as every word is read in. This fact will force Ajjanagadde and Shastri's system to continually modify the number of phases in their clock cycle. For example, consider the following story:

"The short and fat man bought a red Corvette. He called the police when a thief stole it from his mother's garage in Fresno."

In the first sentence short and fat must be propagated and bound to the Height and Weight roles of Human, followed by the propagation of Man to the Actor of Buy, Red to the Color of Automobile, and Corvette to the Object of Buy. At some point, Ajjanagadde and Shastri's system must decide upon the system's number phases per clock cycle in order to represent those bindings. The number of phases per clock will have to be changed, however, to handle the new bindings that sequentially arise from the second sentence and any subsequent sentences. The only apparent way to avoid having to continually modify the phase clock to accommodate new bindings in such cases is to constantly operate the clock at a high enough number of phases per cycle to maintain the

¹As in most connectionist models, activation can propagate from one node to its neighbor in a single ROBIN cycle. Ajjanagadde and Shastri's model defines that a single *phase* of their clock cycle possesses this same ability, and so require that each phase have the same computational complexity as a normal connectionist cycle.

maximum number of bindings the system will ever require. This solution, however, would decrease efficiency when there are only a small number of bindings. This kind of problem does not occur with signatures in ROBIN, since new signatures are simply added to the network and propagated along with the old ones.

Looking beyond explicit comparisons between signature and phase-clock binding mechanisms, a primary functional ability that Ajjanagadde and Shastri's model possesses that ROBIN does not is a pre-existing storage of instances. Facts are hardwired into their networks with *instance* nodes, using a single instance node per fact (such as `orderhit(dave,mike,bob)`). Instance nodes could be duplicated in ROBIN and accessed with signatures, but they are not likely a good final solution to the problem of modelling long-term episodic memory, because of the huge number of instances (and thus instance nodes) involved. One possible future advantage of using *distributed* signatures instead of Ajjanagadde and Shastri's phase-clock is that once signatures are propagated, the distributed representations of the bindings would be immediately available for use by a connectionist learning mechanism to form long-term distributed memories. Such a mechanism is needed to solve the *multiple instance problem* [Sumida & Dyer, 1989].

While both models seem to have nearly equivalent variable binding and rule-firing abilities, ROBIN goes beyond Ajjanagadde and Shastri's model in three major ways: (a) integration of its variable binding mechanism within a connectionist semantic network, (b) multiple binding sites per role, and (c) gating and selectional restrictions. The most important difference is that ROBIN's signature role-binding structure is integrated within an *evidential* connectionist semantic network that performs smooth constraint satisfaction to select a most plausible interpretation from several generated inference paths¹. Nearly as important is that having multiple binding sites per role allows ROBIN to evaluate ambiguous role-bindings in parallel, which is key to handling disambiguation and meaning reinterpretations without backtracking. And finally, gating and selectional restrictions within ROBIN's network structure control the spread of activation and eliminate crosstalk between logically unrelated inferences. These capabilities allow ROBIN to perform much of the high-level inferencing required for natural language understanding, where not only must inference paths be dynamically instantiated, but in which alternative paths must be evaluated and selected among in changing contexts.

5.3. Comparison to Related Connectionist Models of Disambiguation and Reinterpretation

5.3.1. Structured Spreading-Activation Models

ROBIN's disambiguation performance is similar to that of Cottrell & Small [1982] and Waltz & Pollack's [1985] structured spreading-activation models on the simple examples that they handle successfully. Of course, signatures allow ROBIN to perform disambiguation on more complex text that requires inferencing, and so is not limited to performing disambiguation based upon the surface semantics of the input.

To compare ROBIN's disambiguation abilities to that of other spreading-activation models and show how important its virtual role-filler structure is for even simple examples that do not require inferencing, consider what happens in the extended Waltz & Pollack network of Figure 1 when it is presented with input for the sentences "The astronomer saw the star" and "The star saw the astronomer." As shown in Figure 24a, the network disambiguates "star" to mean a Celestial-Body in the sentence "The astronomer saw the star" because of activation flowing over the path Astronomer ↔ Astronomy ↔ Celestial-Body. This is a reasonable disambiguation, since it is

¹[Ajjanagadde & Shastri, 1989] mentions integration with connectionist semantic networks as an area of their own future research.

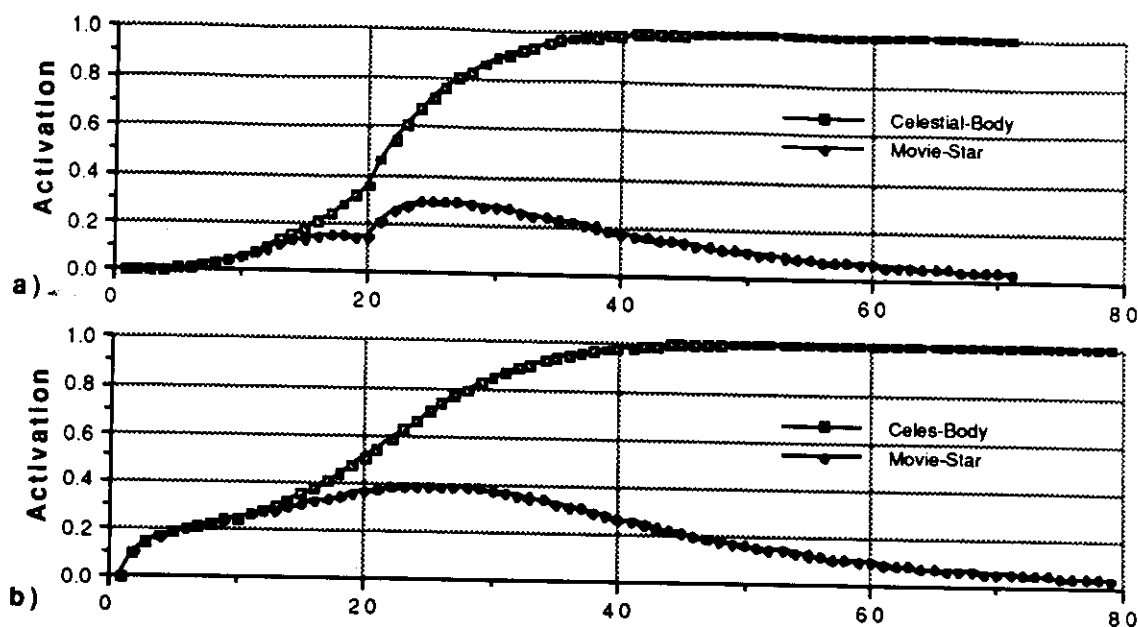


Figure 24. Activations of Celestial-Body and Movie-Star in the network of Figure 1 based on [Waltz & Pollack, 1985] after presentation of input for a) "The astronomer saw the star.", and b) "The star saw the astronomer."

the job of astronomers to study celestial bodies. However, when presented with input for "The star saw the astronomer", the network again disambiguates "star" to Celestial-Body (Figure 24b)¹. This is quite unfortunate, since celestial bodies lack eyes and so cannot see. The problem is that the network of Figure 1 has no way to recognize the difference between the two sentences, and so provides activation evidence to Celestial-Body through See's Object role, even though it is the Astronomer that is (or should be) bound to it.

This is a simple example where the combination of ROBIN's signatures, selectional restrictions, and virtual role-filler structure are crucial for successful interpretation. As seen in Figure 25a, ROBIN also disambiguates "star" to Celestial-Body for "The astronomer saw the star." However, when input for "The star saw the astronomer" is clamped, only the signature of Movie-Star propagates to the Actor role of frame See, since Celestial-Body violates its selectional restrictions. Because of this, only Movie-Star receives activation feedback from See, thus becoming the clear winner and disambiguating the sentence correctly (Figure 25b).

Another difference between ROBIN and other spreading-activation models is the final levels of activation when the network settles. As can be seen in Figures 3 and 24, the direct inhibitory connections of Waltz & Pollack's model drive the non-winner's activations down to zero. This makes it nearly impossible for new context to overcome the inhibition from the winning concept and allow reinterpretation. ROBIN's global inhibition mechanism, on the other hand, serves only to control the spread of activation by normalizing the evidential activations of each concept, leaving their final levels of activation at a value relative to the amount of evidence available for them in that context (e.g. Figure 25).

¹It takes slightly longer since "star" is clamped first and thus provides early evidence to Movie-Star.

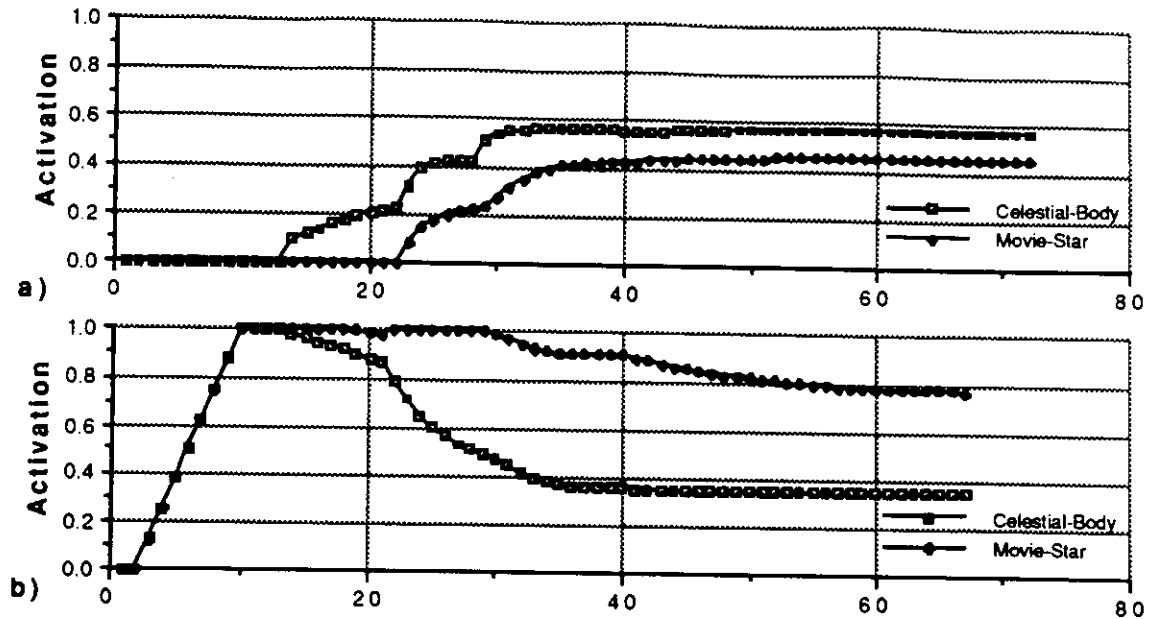


Figure 25. Evidential activations of Celestial-Body and Movie-Star in ROBIN network after presentation of input for a) *"The astronomer saw the star"*, and b) *"The star saw the astronomer."*

In the case of **Hiding Pot**, the global inhibitory normalization is what allowed the word *"pot"* to eventually be reinterpreted to **Marijuana**, despite **Cooking-Pot**'s early dominance. We did not even attempt to run the network with direct inhibitory connections between those competing concepts, because it was clear from the activation plots of Figures 19c and 19d that the new context from *"because the police were coming"* served only to eventually push **Marijuana**'s net input above that of **Cooking-Pot**, and would never have been able to overcome the large amount of inhibition that would have been emanating from **Cooking-Pot** with direct inhibitory connections.

5.3.2. Marker-Passing Models

One way to look at ROBIN's signatures is as a connectionist implementation of a restricted class of symbolic markers, since both signatures and markers allow variable-bindings to be represented and propagated in parallel across semantic networks. Markers, however, can be much more complex than signatures, since they are true symbolic pointers often holding structured information and which can be operated on by symbolic functions on marker-passing nodes. Marker-passing systems also often use multiple types of markers in different stages of the spreading process (e.g. "activation" and "prediction" markers in [Riesbeck & Martin, 1986]) and use named links which may act differently depending upon the type of marker. The converse is that although signatures are less powerful than symbolic markers, they allow ROBIN to perform inferencing using simpler, activation-based connectionist units that implement a single domain and knowledge-independent mechanism.

Propagation of markers and signatures allow both marker-passing systems and ROBIN to generate candidate interpretations of input text in parallel, a crucial advantage over symbolic rule-based models that do so serially. The most important difference between ROBIN and marker-passing systems in terms of disambiguation and reinterpretation is that ROBIN does not need to use a separate path evaluator to select the most plausible interpretation of the many paths generated. As an example of how ROBIN performs the disambiguation and reinterpretation of a marker-passing sys-

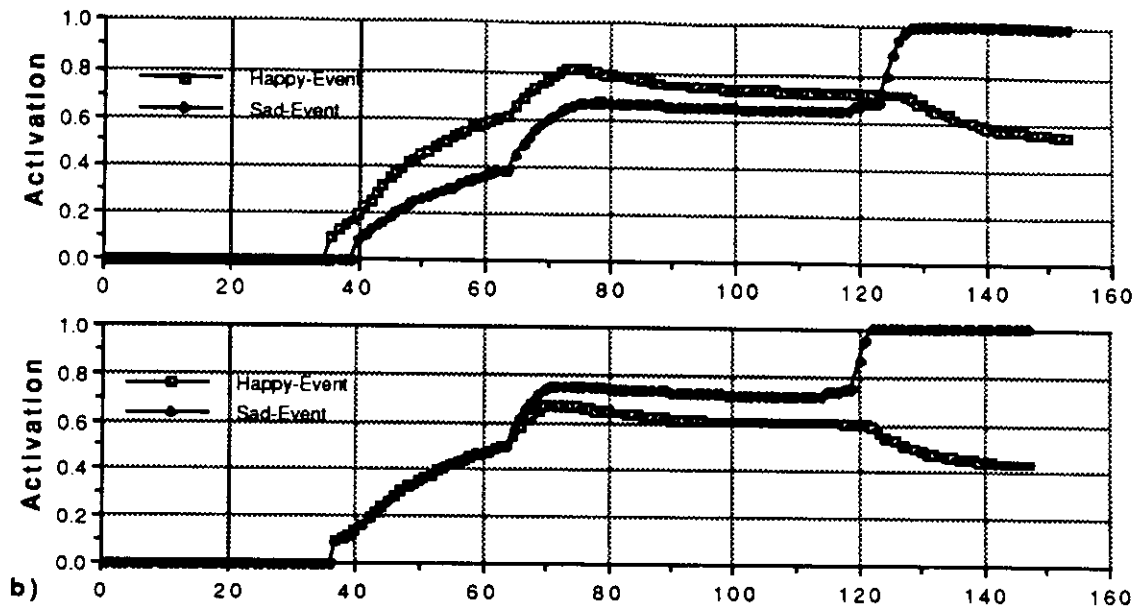


Figure 26. Activations of Happy-State and Sad-State in ROBIN network encoding network of Figure 3 from [Eiselt, 1987], after presentation of input for “Fred asked Wilma to marry him. Wilma began to cry. She was saddened by the proposal.” a) network biased towards marriage proposals being happy events (weight from Propose-Marriage to Happy-Event = 0.7 and to Sad-Event = 0.3). b) network in which marriage proposals are equally likely to be sad events (weight from Propose-Marriage to Happy-Event and Sad-Event = 0.5).

tem, consider how a ROBIN network built from Eiselt’s [1987] knowledge base of Figure 1 processes input for the text:

“Fred proposed to Wilma. Wilma began to cry. Wilma was saddened by the proposal.”
(Marriage)

After activation is spread, both major inference chains between Propose-Marriage and Cry-Tears are instantiated with signatures, just as they are with Eiselt’s markers. Of course, each instantiated frame in ROBIN’s network also has a level of evidential activation. The levels of activation for the linchpin frames Happy-Event and Sad-Event are shown in Figure 26a. As can be seen, Happy-Event initially is the winning interpretation. However, after the “Wilma was saddened” phrase is input to the network at about cycle 120, Sad-Event gets more evidence and thus becomes the more highly-activated of the two, reinterpreting the text. No resort to separate evaluation metrics or symbolic buffers to store discarded paths is necessary.

Another advantage of ROBIN’s connectionist networks over marker-passing networks is that its weighted links and graded activation levels allow the most predictive connections to bias the interpretation more than others. For example, the result shown in Figure 26a was in a network biased to consider marriage proposals as happy events (by having a stronger weight from Propose-Marriage to Happy-Event than to Sad-Event). A more “cynical” network can be modelled simply with modified weights that provide different amounts of evidence, thus leading to a different interpretation without changing the actual knowledge or structure of the network.

Typical Marker Propagation Rules	Propagation of Signatures
Only propagate markers across a certain distance of nodes.	Evidential activation diminishes the further away it gets from an input source. Once it goes below threshold, it and signatures stop propagating.
Typical Path Evaluation Rules	Evidential Activation Effect
Reject paths whose frames' binding constraints have been violated.	Selectional restriction structure stops signatures and evidential activation from spreading to frames whose binding constraints would be violated.
Select paths that include most of the input.	Each clamped input is a maximal source of evidential activation — so paths that include more of the input generally have more activation.
For paths explaining the same inputs, select the shortest path.	Shorter paths have less distance for activation to propagate away and diminish, so generally have more activation.

Table 3. Typical marker-passing rules for propagation of markers and evaluation of inference paths contrasted with a similar gross effect from the spread of activation in ROBIN.

For example, the result shown in Figure 26b is from a network in which marriage proposals are equally as likely to be sad as happy. On hearing of the proposal, it cannot “decide” between Happy-Event and Sad-Event, so their two activations increase at the same rate. However, when the network is presented with the information that Wilma cried (cycle 60), there is more evidence that she was sad (because of a slight weight bias towards Sad-State from Cry-Tears). The final information that “*Wilma was saddened*” at cycle 120 only confirms the conclusion. Thus, in ROBIN (as in all structured spreading-activation networks), the same network can return different interpretations to the same input when its weights are biased towards different concepts — as opposed to the binary nature of most marker-passing networks’ paths.

One interesting thing to note is the gross similarities between the evaluation rules used by marker-passing systems and the kinds of inference paths favored by ROBIN’s spread of evidential activation. For example, most marker-passing systems have a rule that selects paths that include more of the input than others. This naturally tends to occur in ROBIN, because each clamped input is a maximal source of evidential activation — so paths that include more of the input will generally have more activation. Other typical marker-passing path evaluation rules and how they seem to correspond to the properties of spreading activation in ROBIN are shown in Table 3.

Of course, while marker-passing evaluation heuristics are hard and fast rules, the corresponding tendencies in ROBIN are soft constraints that emerge from the spreading-activation process. These emergent constraints can be and often are “overruled” by other activation tendencies and biases from connection strengths and priming. Most important of all is that ROBIN’s disambiguation and reinterpretation happens within the network at the same time as inference paths are generated. In marker-passing systems, on the other hand, path evaluators are a symbolic mechanism separate from the spreading-activation process that operates in serial after paths have been generated, a huge disadvantage as the size of the networks increase and the number of generated inference paths to be evaluated explodes.

5.4. Conclusions

High-level inferencing is a fundamental problem in cognitive tasks such as natural language understanding and planning. Symbolic, rule-based models can make high-level inferences necessary for understanding text, but handle ambiguity poorly, especially when later context requires a reinterpretation of the input. Distributed connectionist models, on the other hand, are able to learn to perform disambiguation, but only for simple sentences that can be understood based on one-step inferences from the surface semantics of the input or on script-based stories that they have been previously trained to recognize.

Marker-passing networks and structured spreading-activation networks seem to understand stories that require high-level inferencing. Marker-passing systems use their built-in symbolic abilities to perform inferencing and generate possible interpretations of the text in parallel. However, they must employ separate path evaluation mechanisms to disambiguate between the multiple paths generated by propagation of markers. Worse yet, the number of inference paths generated explodes as the size of knowledge-bases increase, slowing them down dramatically. Structured spreading-activation networks, on the other hand, use their weighted connections and graded levels of activation to select a single most-plausible interpretation in a given context through the spreading-activation process. Unfortunately, because of their inability to represent variable bindings and perform inferencing, they have been unable to go beyond disambiguation based on the surface semantics of the input.

We have described a structured spreading-activation model, ROBIN, that is able to perform much of the massively-parallel inferencing of marker-passing systems by propagating activation patterns serving as concepts' *signatures*. Using structure that holds signature activations, ROBIN is able to dynamically create novel frame instances by binding roles with any previously known concepts in the network. Since signatures are simply activation patterns that uniquely identify the concept bound to roles, they can be propagated in parallel across separate paths of binding nodes that preserve their activation, thus performing inferencing.

Just as importantly, the ambiguous candidate interpretations generated by the propagation of signatures are selected between by the spread of activation along ROBIN's *evidential* semantic network structure, which is similar to that of normal structured spreading-activation networks. The network thus combines evidence from context for each inference path and settles upon a single most-highly activated path by constraint satisfaction. Furthermore, because each concept in the network retains a level of activation that corresponds to the amount of evidence in its favor, reinterpretation occurs automatically if new context causes another inference path to become more highly-activated than the original winner.

Once structured spreading-activation networks are extended to handle inferencing, potential problems from crosstalk make it vital for their static evidential structure to interact with the dynamic structure represented by the current variable bindings of the network. We have identified two reasons why this is especially important: to assure that activation feedback is between frames and their actual (rather than prototypical) role-fillers, and to stop activation from spreading to frames whose selectional restrictions have been violated. ROBIN assures this *virtual structure* by connections of nodes and links between the signature and evidential portions of the network, thus eliminating large sources of potential crosstalk.

ROBIN is thus able to handle many of the high-level inferencing tasks not addressed by other connectionist models, while at the same time perform disambiguation and semantic reinterpretation often difficult for symbolic systems.

REFERENCES

- Ajjanagadde, V. (1990): Reasoning With Function Symbols in a Connectionist System. *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*. Boston, MA, July 1990.
- Ajjanagadde, V. & Shastri, L. (1989): Efficient Inference With Multi-Place Predicates and Variables in a Connectionist System. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*. Ann Arbor, MI, August 1989.
- Barnden, J. (1990): The Power of Some Unusual Connectionist Data-Structuring Techniques. In J. A. Barnden and J. B. Pollack (eds.), *Advances in Connectionist and Neural Computation Theory*. Norwood, NJ: Ablex.
- Charniak, E. (1986): A Neat Theory of Marker Passing. *Proceedings of the National Conference on Artificial Intelligence (AAAI-86)*, Philadelphia, PA, August, 1986.
- Cottrell, G. & Small, S. (1982): A Connectionist Scheme for Modeling Word-Sense Disambiguation. *Cognition and Brain Theory*, Vol. 6, p. 89-120.
- Derthick, M. (1988): *Mundane Reasoning by Parallel Constraint Satisfaction*. (Tech Rep. No CMU-CS-88-182). Doctoral Dissertation, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA.
- Diederich, J. (1990): Steps Toward Knowledge-Intensive Connectionist Learning. In J. Barnden and J. Pollack (eds.), *Advances in Connectionist and Neural Computation Theory*, Ablex Publishing. In press.
- Dolan, C. P. (1989). *Tensor Manipulation Networks: Connectionist and Symbolic Approaches to Comprehension, Learning, and Planning*. (Tech Rep. No UCLA-AI-89-06), Doctoral Dissertation, Computer Science Department, University of California, Los Angeles.
- Dolan, C. P. & Smolensky, P. (1989). Tensor Product Production System: A Modular Architecture and Representation. *Connection Science*, Vol. 1 (1), p. 53-68.
- Dyer, M. G. (1983): *In-Depth Understanding: A Computer Model of Integrated Processing for Narrative Comprehension*. Cambridge, MA: The MIT Press.
- Dyer, M. G. (1990): Symbolic Neuroengineering for Natural Language Processing: A Multi-Level Research Approach. In J. Barnden and J. Pollack (eds.), *Advances in Connectionist and Neural Computation Theory*. Norwood, NJ: Ablex.
- Eiselt, K. P. (1987): Recovering From Erroneous Inferences. *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*. Seattle, WA, July 1987.
- Feldman, J. A. (1989): Neural Representation of Conceptual Knowledge. In L. Nadel, L. A. Cooper, P. Culicover, and R. M. Harnish (eds), *Neural Connections, Mental Computation*, p. 68-103. Cambridge, MA: The MIT Press.
- Feldman, J. A. & Ballard, D. H. (1982): Connectionist Models and Their Properties. *Cognitive Science*, Vol. 6 (3), p. 205-254.
- Fodor, J. A. & Pylyshyn, Z. W. (1988): Connectionism and Cognitive Architecture: A Critical Analysis. In S. Pinker and J. Mehler (eds.), *Connections and Symbols*, p. 3-71. Cambridge, MA: The MIT Press.
- Granger R. H., Eiselt, K. P., & Holbrook, J. K. (1986): Parsing with Parallelism: A Spreading Activation Model of Inference Processing During Text Understanding. In J. Kolodner and C. Riesbeck (eds.), *Experience, Memory, and Reasoning*, p. 227-246. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

- Hendler, J. (1988): *Integrating Marker-Passing and Problem Solving: A Spreading Activation Approach to Improved Choice in Planning*, Lawrence Erlbaum Associates, Hillsdale, New Jersey.
- Holldobler, S. (1990): A Structured Connectionist Unification Algorithm. *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-90)*. Boston, MA, July 1990.
- Kitano, H., Tomabechi, H. & Levin, L. (1989): Ambiguity Resolution in DMTRANS PLUS. *Proceedings of the Fourth Conference of the European Chapter of the Association of Computational Linguistics*. Manchester University Press.
- Lange, T. (1990): Simulation of Heterogeneous Neural Networks on Serial and Parallel Machines. *Parallel Computing*, 14.
- Lange, T. (in press): Lexical and Pragmatic Disambiguation and Reinterpretation in Connectionist Networks. *International Journal of Man-Machine Studies*.
- Lange, T. & Dyer, M. G. (1988): Dynamic, Non-Local Role-Bindings and Inferencing in a Localist Network for Natural Language Understanding. In David S. Touretzky (ed.), *Advances in Neural Information Processing Systems I*, p. 545-552. San Mateo, CA: Morgan Kaufmann (Collected papers of the IEEE Conference on Neural Information Processing Systems — Natural and Synthetic, Denver, CO, November 1988).
- Lange, T. & Dyer, M. G. (1989a): Frame Selection in a Connectionist Model of High-Level Inferencing. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society (CogSci-89)*, Ann Arbor, MI, August 1989.
- Lange, T. & Dyer, M. G. (1989b): High-Level Inferencing in a Connectionist Network. *Connection Science*, 1 (2), p. 181-217.
- Lange, T., Hodges, J., Fuenmayor, M., & Belyaev, L. (1989a): DESCARTES: Development Environment For Simulating Hybrid Connectionist Architectures. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society (CogSci-89)*, Ann Arbor, MI, August 1989.
- Lange, T., Vidal, J., & Dyer, M. G. (1989b): Artificial Neural Oscillators For Inferencing. *Proceedings of the International Workshop on Neurocomputers and Attention*, Moscow, USSR, September 1989. (Tech Rep. No. UCLA-AI-89-11, Computer Science Department, University of California, Los Angeles).
- Lytinen, S. (1984): Frame Selection in Parsing. *Proceedings of the National Conference on Artificial Intelligence (AAAI-84)*, August 1984.
- McClelland, J. L. & Kawamoto, A. H. (1986): Mechanisms of Sentence Processing: Assigning Roles to Constituents of Sentences. In McClelland & Rumelhart (eds.), *Parallel Distributed Processing: Vol 2*, p. 272-325. Cambridge, MA: The MIT Press.
- Miikkulainen, R. & Dyer, M. G. (1988): Forming Global Representations with Extended Back-propagation. *Proceedings of the IEEE Second Annual Conference on Neural Networks (ICNN-88)*, San Diego, CA.
- Miikkulainen, R. & Dyer, M. G. (1989): Encoding Input/Output Representations in Connectionist Cognitive Systems. In D. Touretzky, G. Hinton, and T. Sejnowski (eds.). *Proceedings of the 1988 Connectionist Models Summer School*. San Mateo, CA: Morgan Kaufmann.
- Miikkulainen, R. & Dyer, M. G. (1989): A Modular Neural Network Architecture for Sequential Paraphrasing of Script-Based Stories. *Proceedings of the International Joint Conference on Neural Networks*. Washington, DC, June, 1989.

- Minsky, M. (1975): A Framework for Representing Knowledge. In P.H. Winston (ed.), *The Psychology of Computer Vision*, p. 211-277. New York: McGraw-Hill.
- Norvig, P. (1986): *A Unified Theory of Inference For Text Understanding*. Doctoral Dissertation, Computer Science Department, University of California, Berkeley.
- Riesbeck, C. K. & Martin, C. E. (1986): Direct Memory Access Parsing. In J. Kolodner and C. Riesbeck (eds.), *Experience, Memory, and Reasoning*, p. 209-226. Hillsdale, New Jersey: Lawrence Erlbaum Associates.
- Rumelhart, D. E., Hinton, G. E., & McClelland, J. L. (1986): A General Framework for Parallel Distributed Processing. In Rumelhart & McClelland (eds.), *Parallel Distributed Processing: Vol 1*, p. 45-76. Cambridge, MA: The MIT Press.
- Schank, R. C. & Abelson, R. (1977): *Scripts, Plans, Goals and Understanding*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.
- Shastri, L. (1988): A Connectionist Approach to Knowledge Representation and Limited Inference. *Cognitive Science*, Vol. 12 (3), p. 331-392.
- St. John, M. F. (1990): *The Story Gestalt: Text Comprehension by Cue-based Constraint Satisfaction*. Technical Report No. 9004 (Doctoral Dissertation), Department of Cognitive Science, University of California, San Diego.
- Sumida, R. A. & Dyer, M. G. (1989): Storing and Generalizing Multiple Instances While Maintaining Knowledge-Level Parallelism. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, Detroit, MI, August 1989.
- Tomabechi, H. and Kitano, H. (1989): Beyond PDP: The Frequency Modulation Neural Network Architecture. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, Detroit, MI, August 1989.
- Touretzky, D. S. (1990): Connectionism and Compositional Semantics. In J. A. Barnden and J. B. Pollack (eds.), *Advances in Connectionist and Neural Computation Theory*. Norwood, NJ: Ablex.
- Touretzky, D. S. & Hinton, G. E. (1988): A Distributed Connectionist Production System. *Cognitive Science*, 12 (3), p. 423-466.
- Waltz, D. & Pollack, J. (1985): Massively Parallel Parsing: A Strongly Interactive Model of Natural Language Interpretation. *Cognitive Science*, 9 (1), p. 51-74.

APPENDIX A

The complete knowledge base definitions used to construct the network that processes inputs for episodes such as **Hiding Pot** and others in its domain (some shown in Table 2) are listed below. An overview of part of this knowledge base was shown in Figure 5. The networks built to process episodes from other models (e.g. **Star Marriage** from [Waltz & Pollack, 1985] and **Marriage** from [Eiselt, 1987]) were defined in separate knowledge bases that encode knowledge similar to that in their networks.

```

;*****
;*   Simple phrase entries, i.e. words defining kinds of Physical-Objects. *
;*****

(Frame <animal>      Phrase :Phrase-For (Animal          1.00))
(Frame <cigarette>   Phrase :Phrase-For (Cigarette       1.00))
(Frame <dishwasher>  Phrase :Phrase-For (Dishwasher      1.00))
(Frame <flower>      Phrase :Phrase-For (Flower          1.00))
(Frame <fork>        Phrase :Phrase-For (Eating-Utensil 1.00))
(Frame <gun>         Phrase :Phrase-For (Firearm         0.5))
(Frame <human>       Phrase :Phrase-For (Human           0.5))
(Frame <John>        Phrase :Phrase-For (John            1.00))
(Frame <man>         Phrase :Phrase-For (Human           0.5))
(Frame <marijuana>   Phrase :Phrase-For (Marijuana        0.4))
(Frame <Mary>        Phrase :Phrase-For (Mary            1.00))
(Frame <meat>        Phrase :Phrase-For (Meat            1.00))
(Frame <plant>       Phrase :Phrase-For (Plant           1.00))
(Frame <police>      Phrase :Phrase-For (Police          1.00))
(Frame <pot>         Phrase :Phrase-For (Marijuana        0.6)
                          (Cooking-Pot          1.00)
                          (Planting-Pot         1.00))
(Frame <skillet>     Phrase :Phrase-For (Skillet         1.00))
(Frame <stove>       Phrase :Phrase-For (Stove           1.00))
(Frame <thing>       Phrase :Phrase-For (Phys-Obj        1.00))
(Frame <toilet>      Phrase :Phrase-For (Toilet          1.00))
(Frame <water>       Phrase :Phrase-For (Water           1.00))

```

```

;*****
;*   Action verb phrase entries representing actions.   *
;*****

; e.g. "The police arrested John."
(Frame <arrest> Phrase
  :Roles      (Actor   (Police  0.50))
              (Object  (Human  0.05))
  :Phrase-For (Police-Arrest  1.00))

; e.g. "John cleaned the pot inside the dishwasher."
(Frame <clean> Phrase
  :Roles      (Actor   (Human  0.05))
              (Object  (Phys-Obj 0.05))
              (Location (Phys-Obj 0.05))
  :Phrase-For ($Dishwasher-Cleanin g 1.0 (Actor   Actor)
              (Object  Object)
              (Location Location)))

; e.g. "John cooked the meat in the dishwasher"
(Frame <cook> Phrase
  :Roles      (Actor   (Human  0.05))
              (Object  (Edible-Obj 0.20))
              (Location (Phys-Obj 0.05))
  :Phrase-For ($Stove-Cooking  1.00)
              ($Baking         1.0))

; e.g. "John came to the store."
(Frame <come> Phrase
  :Roles      (Actor   (Human  0.05))
              (Location (Phys-Obj 0.05))
  :Phrase-For (Transfer-Self 1.0 (Actor   Actor)
              (Location Location)))

; e.g. "John grew the flower in the pot."
(Frame <grow> Phrase
  :Roles      (Actor   (Human  0.05))
              (Object  (Animate 0.05))
              (Location (Phys-Obj 0.05))
  :Phrase-For ($Grow-Potted-Plant 1.00))

; e.g. "John hide the marijuana in the toilet."
(Frame <hide> Phrase
  :Roles      (Actor   (Human  0.05))
              (Object  (Phys-Obj 0.05))
              (Location (Container-Obj 0.05))
  :Phrase-For (Avoid-Detection 1.0 (Actor   Actor)
              (Object  Object)
              (Location Location)))

; e.g. "John lit the cigarette."
(Frame <lit> Phrase
  :Roles      (Actor   (Human  0.05))
              (Object  (Flammable-Obj 0.20))
              (Location (Phys-Obj 0.05))
  :Phrase-For (On-Top-Of-Heat-Source 1.00))

```



```

; e.g. "John put the pot inside the dishwasher."
(Frame <put_inside> Phrase
  :Roles      (Actor   (Human   0.05))
              (Object  (Phys-Obj 0.05))
              (Location (Container-Obj 0.05))
  :Phrase-For (Transfer-Inside 1.0 (Actor Actor)
              (Object Object)
              (Location Location)))

; e.g. "John planted the flower in the pot."
(Frame <planted> Phrase
  :Roles      (Actor   (Human   0.05))
              (Object  (Phys-Obj 0.05))
              (Location (Phys-Obj 0.05))
  :Phrase-For ($GP-Plant 1.0))

; e.g. "John put the pot on top of the stove."
(Frame <put_on_top_of> Phrase
  :Roles      (Actor   (Human   0.05))
              (Object  (Phys-Obj 0.05))
              (Location (Phys-Obj 0.05))
  :Phrase-For (Transfer-On-Top 1.00))

; e.g. "The police saw the pot on the stove."
(Frame <see> Phrase
  :Roles      (Actor   (Human   0.05))
              (Object  (Phys-Obj 0.05))
              (Location (Phys-Obj 0.05))
  :Phrase-For (See-Object 1.00))

; e.g. "John shot the Mary with a gun."
(Frame <shoot> Phrase
  :Roles      (Actor   (Human   0.05))
              (Object  (Phys-Obj 0.05))
              (Inst    (Weapon  0.05))
  :Phrase-For ($Shoot-Person 1.00))

; e.g. "John smoked the pot."
(Frame <smoke> Phrase
  :Roles      (Actor   (Human   0.05))
              (Object  (Phys-Obj 0.05))
  :Phrase-For ($Burn-And-Inhale 1.00)
              ($Smoke-Food 1.00))

; e.g. "John watered the pot."
(Frame <watered> Phrase
  :Roles      (Actor   (Human   0.05))
              (Object  (Phys-Obj 0.05))
              (Inst    (Phys-Obj 0.05))
  :Phrase-For ($GP-Water-Plant 1.0))

```

```

;*****
;* State verb phrase entries representing actions.
;*****

; e.g. "John has a gun."
(Frame <has>      Phrase
      :Roles      (Actor   (Human   0.05))
                  (Object  (Phys-Obj 0.05))
      :Phrase-For (Possess-Obj 1.00))

; e.g. "John has cancer."
(Frame <has_cancer> Phrase
      :Roles      (Object  (Human   0.05))
      :Phrase-For (Lung-Cancer 1.00))

; e.g. "The pot is clean."
(Frame <is_clean>  Phrase
      :Roles      (Object  (Phys-Obj 0.05))
      :Phrase-For (Clean    1.00))

; e.g. "John is dying."
(Frame <is_dying>  Phrase
      :Roles      (Object  (Animate 0.05))
      :Phrase-For (Dying    1.00))

; e.g. "John is in jail."
(Frame <is_in_jail> Phrase
      :Roles      (Object  (Human   0.05))
      :Phrase-For (In-Jail  1.00))

; e.g. "The pot is inside of the dishwasher."
(Frame <is_inside_of> Phrase
      :Roles      (Actor   (Human   0.05))
                  (Object  (Phys-Obj 0.05))
                  (Location (Phys-Obj 0.05))
      :Phrase-For (Inside-Of 1.00))

; e.g. "The cigarette is lit."
(Frame <is_lit>    Phrase
      :Roles      (Object  (Phys-Obj 0.05))
      :Phrase-For (Burning  1.00))

; e.g. "The police are next to John."
(Frame <is_next_to> Phrase
      :Roles      (Object  (Human   0.05))
                  (Location (Phys-Obj 0.05))
      :Phrase-For (Proximity-Of 1.00))

; e.g. "The pot is on top of the stove."
(Frame <is_on_top_of> Phrase
      :Roles      (Object  (Human   0.05))
                  (Location (Phys-Obj 0.05))
      :Phrase-For (On-Top-Of  1.00))

; e.g. "The meat is ready."
(Frame <is_ready>  Phrase
      :Roles      (Object  (Human   0.05))
      :Phrase-For (Food-Prepared 1.00))

```

```

;*****
;*   Frames representing objects in the world in an is-a heirarchy.
;*****

```

```

(Frame Phys-Obj  Object
:Phrase          (<thing> 1.00)
:Refinements     (Animate 1.00)
                  (Inanimate 1.00))

(Frame Animate  Object
:Refinement-Of  (Phys-Obj 0.3)
:Refinements     (Human 1.00)
                  (Animal 1.00)
                  (Plant 1.00))

(Frame Inanimate Object
:Refinement-Of  (Phys-Obj 0.7)
:Refinements     (Edible-Obj 1.00)
                  (Man-Made-Obj 1.00))

(Frame Human    Object
:Phrase          (<human> 1.00)
                  (<man> 1.00)
:Refinement-Of  (Animate 0.7)
:Refinements     (Police 1.00)
                  (John 1.00)
                  (Mary 1.00))

(Frame Animal   Object
:Phrase          (<animal> 1.00)
:Refinement-Of  (Animate 0.2))

(Frame Plant    Object
:Phrase          (<plant> 0.5)
:Refinement-Of  (Animate 0.1)
:Refinements     (Flower 1.00)
                  (Marijuana 1.00)
:Uses           ($Grow-Potted-Plant^Plant 1.00))

(Frame Edible-Obj Object
:Refinement-Of  (Inanimate 0.3)
:Refinements     (Meat 1.00)
                  (Water 1.00))

(Frame Man-Made-Obj Object
:Refinement-Of  (Inanimate 0.6)
:Refinements     (Household-Item 1.00)
                  (Weapon 1.00)
                  (Drug 1.00))

(Frame Police   Object
:Phrase          (<police> 1.00)
:Refinement-Of  (Human 0.05)
:Uses           (Police-Capture^Police 1.00))

(Frame John     Object
:Phrase          (<John> 1.00)
:Refinement-Of  (Human 0.05))

(Frame Mary     Object
:Phrase          (<Mary> 1.00)
:Refinement-Of  (Human 0.05))

(Frame Flower   Object
:Phrase          (<flower> 1.00)
:Refinement-Of  (Plant 0.1))

```

```

(Frame Meat      Object
:Phrase         (<meat> 1.00)
:Refinement-Of (Edible-Obj 0.5)
:Uses          ($Smoke-Food^Object 1.00))

(Frame Water    Object
:Phrase         (<water> 0.6)
:Refinement-Of (Edible-Obj 0.1))

(Frame Household-Item Object
:Refinement-Of (Man-Made-Obj 0.1)
:Refinements  (Cooking-Utensil 1.00)
               (Dishwasher 1.00)
               (Eating-Utensil 1.00)
               (Stove 1.00)
               (Planting-Pot 1.00)
               (Toilet 1.00))

(Frame Weapon   Object
:Refinement-Of (Man-Made-Obj 0.1)
:Refinements  (Firearm 1.00)
:Uses        (Police-Capture^Evidencer 0.5))

(Frame Drug     Object
:Refinement-Of (Man-Made-Obj 0.1)
:Refinements  (Cigarette 1.00)
               (Marijuana 1.00))

(Frame Cooking-Utensil Object
:Refinement-Of (Household-Item 0.1)
:Refinements  (Cooking-Pot 1.0)
               (Skillet 1.0))

(Frame Eating-Utensil Object
:Phrase         (<fork> 1.00)
:Refinement-Of (Household-Item 0.1))

(Frame Cooking-Pot Object
:Phrase         (<pot> 0.3)
:Refinement-Of (Cooking-Utensil 0.3)
               (Container-Obj 0.1)
               (Opaque-Obj 0.1))

(Frame Skillet  Object
:Phrase         (<skillet> 1.0)
:Refinement-Of (Cooking-Utensil 0.3)
               (Container-Obj 0.1))

(Frame Dishwasher Object
:Phrase         (<dishwasher> 1.00)
:Refinement-Of (Household-Item 0.1)
               (Container-Obj 0.1)
               (Opaque-Obj 0.1)
:Uses          ($Dishwasher-Cleaning^Cleaner 1.00))

(Frame Stove    Object
:Phrase         (<stove> 1.00)
:Refinement-Of (Household-Item 0.1)
               (Container-Obj 0.1)
               (Opaque-Obj 0.1)
               (Heat-Source 0.1)
:Uses          ($Stove-Cooking^Stove 1.00)
               ($Baking^Stove 1.00))

(Frame Planting-Pot Object
:Phrase         (<pot> 0.4)
:Refinement-Of (Household-Item 0.1)
               (Container-Obj 0.1)
               (Opaque-Obj 0.1)
:Uses          ($Grow-Potted-Plant^Pot 1.00))

```

```

(Frame Toilet      Object
:Phrase          (<toilet>      1.00)
:Refinement-Of  (Household-Item 0.1)
                (Container-Obj  0.1)
                (Opaque-Obj   0.1))

(Frame Firearm    Object
:Phrase          (<gun> 1.00)
:Refinement-Of  (Weapon 0.1)
:Uses           ($Shoot-Person^Firearm 1.00))

(Frame Cigarette  Object
:Phrase          (<cigarette> 1.00)
:Refinement-Of  (Drug 0.1)
                (Flammable-Obj 0.1)
:Uses           ($Burn-And-Inhale^Drug 0.6))

(Frame Marijuana  Object
:Phrase          (<marijuana> 1.00)
                (<pot> 0.3)
:Refinement-Of  (Drug 0.1)
                (Plant 0.1)
                (Illegal-Possess-Obj 0.3)
                (Flammable-Obj 0.1)
:Uses           ($Burn-And-Inhale^Drug 0.4))

(Frame Illegal-Possess-Obj Object
:Refinements   (Marijuana 1.00)
:Uses          (Possess-Illegal-Obj^Object 1.00)
                (Police-Capture^Evidencer 0.50))

(Frame Opaque-Obj Object
:Refinements   (Cooking-Pot 1.00)
                (Dishwasher 1.00)
                (Planting-Pot 1.00)
                (Stove 1.00)
                (Toilet 1.00)
:Uses          (Inside-Of-Opaque^Object 1.00))

(Frame Container-Obj Object
:Refinements   (Cooking-Pot 1.00)
                (Dishwasher 1.00)
                (Planting-Pot 1.00)
                (Skillet 1.00)
                (Stove 1.00)
                (Toilet 1.00)
:Uses          (Inside-Of^Object 1.00))

(Frame Flammable-Obj Object
:Refinements   (Marijuana 1.00)
                (Cigarette 1.00))

(Frame Heat-Source Object
:Refinements   (Stove 1.00))

```

```

;*****
;** Script $Dishwasher-Cleaning ("clean"):
;
;
; ** A person (Planner) uses a dishwasher (Cleaner) to make his
; ** dishes(Dishes) clean (Clean-Dishes).
;
; ** Precondition: The Dishes are inside of the Cleaner
;                  (Inside-Of-Dishwasher).
; ** Results-In:   The Dishes are clean (Clean-Dishes).
;
; ** Roles for Script $Dishwasher-Cleaning. The roles are:
;
; ** Planner: The person planning and doing the $Dishwasher-Cleaning.
;             [Human]
; ** Dishes:  The things being cleaned.
;             [Cooking-Pot or Eating-Utensil]
; ** Cleaner: The thing doing the cleaning.
;             [Dishwasher]
;
(Role $Dishwasher-Cleaning^Planner :Prototypes (Human 0.05)
:Uses ($Dishwasher-Cleaning Actor)
      (Inside-Of-Dishwasher Planner))
(Role $Dishwasher-Cleaning^Dishes :Prototypes (Cooking-Utensil 0.2)
:Uses ($Dishwasher-Cleaning Object)
      (Eating-Utensil 0.2)
      (Inside-Of-Dishwasher Object))
(Role $Dishwasher-Cleaning^Cleaner :Prototypes (Dishwasher 0.9)
:Uses ($Dishwasher-Cleaning Location)
      (Inside-Of-Dishwasher Location))

(Frame $Dishwasher-Cleaning Script
:Roles (Actor $Dishwasher-Cleaning^Planner)
       (Object $Dishwasher-Cleaning^Dishes)
       (Location $Dishwasher-Cleaning^Cleaner)
:Phrase (<clean> 1.0 (Actor Actor)
        (Object Object)
        (Location Location))
:Precond (Inside-Of-Dishwasher 1.00)
:Results-In (Clean-Dishes 1.0 (Actor Planner)
            (Object Object)))

(Frame Inside-Of-Dishwasher State
:Roles (Planner $Dishwasher-Cleaning^Planner)
       (Object $Dishwasher-Cleaning^Dishes)
       (Location $Dishwasher-Cleaning^Cleaner)
:Refinement-Of (Inside-Of 1.0 (Planner Planner)
              (Object Object)
              (Location Location))
:Precond-For ($Dishwasher-Cleaning 1.00))

(Frame Clean-Dishes State
:Roles (Planner (Human 0.05))
       (Object (Cooking-Utensil 0.2)
              (Eating-Utensil 0.2))
:Refinement-Of (Clean 1.0 (Planner Planner)
              (Object Object))
:Results-Of ($Dishwasher-Cleaning 1.0
            (Planner Actor)
            (Object Object)))

```

```

;*****
;** Script $Stove-Cooking ("cook"):
;**
;** A person (Planner) cooks his food (Food) on top of a stove (Stove) in
;** a cooking-pot (Pot).
;**
;** Preconditions: The Pot is on top of the Stove (On-Top-Of-Stove).
;** The Food is inside of the Pot (Inside-Of-Cooking-Pot).
;** Results-In: The Food is cooked (Food-Prepared).
;** The Pot is dirty (Dirty-Dishes).

;** Planner: The person (Human) planning and doing the $Stove-Cooking.
;** Stove: The stove (Stove) being used.
;** Pot: The pot (Cooking-Pot) the food is being cooked in.
;** Food: The food (Edible-Obj) being cooked.

(Role $Stove-Cooking^Planner :Prototypes (Human 0.05)
:Uses ($Stove-Cooking Actor)
(On-Top-Of-Stove Planner))

(Role $Stove-Cooking^Stove :Prototypes (Stove 0.9)
:Uses ($Stove-Cooking Location)
(On-Top-Of-Stove Location))

(Role $Stove-Cooking^Pot :Prototypes (Cooking-Pot 0.5)
(Skillet 0.5)
:Uses ($Stove-Cooking Inst)
(On-Top-Of-Stove Object))

(Role $Stove-Cooking^Food :Prototypes (Edible-Obj 0.4)
:Uses ($Stove-Cooking Object))

(Frame $Stove-Cooking Script
:Roles (Actor $Stove-Cooking^Planner)
(Object $Stove-Cooking^Food)
(Location $Stove-Cooking^Stove)
(Inst $Stove-Cooking^Pot)
:Phrase (<cook> 1.00 (Actor Actor)
(Object Object)
(Location Location))
:Precond (On-Top-Of-Stove 1.00)
:Results-In (Food-Prepared 1.0 (Actor Planner)
(Object Object))
(Dirty-Dishes 1.0 (Actor Planner)
(Object Object)))

(Frame On-Top-Of-Stove State
:Roles (Planner $Stove-Cooking^Planner)
(Object $Stove-Cooking^Pot)
(Location $Stove-Cooking^Stove)
:Refinement-Of (On-Top-Of 1.0 (Planner Planner)
(Object Object)
(Location Location))
:Precond-For ($Stove-Cooking 1.00)
:Precond (Inside-Of-Cooking-Pot 1.00))

```

```

;*****
; ** Script $Baking:
; **
; ** A person (Planner) bakes his food (Food) inside of a stove (Stove) in
; ** a cooking-pot or skillet (Pot).
; **
; ** Preconditions: The Pot is inside of the Stove (Inside-Of-Stove).
; ** The Food is inside of the Pot (Inside-Of-Cooking-Pot).
; ** Results-In: The Food is cooked (Food-Prepared).
; ** The Pot is dirty (Dirty-Dishes).
; **
; ** Planner: The person (Human) planning and doing the $Baking.
; ** Stove: The stove (Stove) being used.
; ** Pot: The pot (Cooking-Pot) the food is being cooked in.
; ** Food: The food (Edible-Obj) being cooked.

(Role $Baking^Planner :Prototypes (Human 0.05)
:Uses ($Baking Actor)
(Inside-Of-Stove Planner))

(Role $Baking^Stove :Prototypes (Stove 0.9)
:Uses ($Baking Location)
(Inside-Of-Stove Location))

(Role $Baking^Pot :Prototypes (Cooking-Pot 0.5)
(Skillet 0.5)
:Uses ($Baking Inst)
(Inside-Of-Stove Object))

(Role $Baking^Food :Prototypes (Edible-Obj 0.4)
:Uses ($Baking Object))

(Frame $Baking Script
:Roles (Actor $Baking^Planner)
(Object $Baking^Food)
(Location $Baking^Stove)
(Inst $Baking^Pot)
:Phrase (<cook> 1.00 (Actor Actor)
(Object Object)
(Location Location))
:Precond (Inside-Of-Stove 1.00)
:Results-In (Food-Prepared 1.0 (Actor Planner)
(Object Object))
(Dirty-Dishes 1.0 (Actor Planner)
(Object Object)))

(Frame Inside-Of-Stove State
:Roles (Planner $Baking^Planner)
(Object $Baking^Pot)
(Location $Baking^Stove)
:Refinement-Of (Inside-Of 1.0 (Planner Planner)
(Object Object)
(Location Location))
:Precond-For ($Baking 1.00)
:Precond (Inside-Of-Cooking-Pot 1.00))

```



```

(Frame Inside-Of-Cooking-Pot
  State
  :Roles      (Planner (Human      0.1))
              (Object  (Edible-Obj 0.2))
              (Location (Cooking-Utensil 0.3))
  :Refinement-Of (Inside-Of      1.0 (Planner Planner)
                 (Object Object)
                 (Location Location))
  :Precond-For  (On-Top-Of-Stove 1.0 (Planner Planner)
                 (Object Object)
                 (Location Location))
                 (Inside-Of-Stove 1.0 (Planner Planner)
                 (Object Object)
                 (Location Location))

(Frame Dirty-Dishes State
  :Roles      (Planner (Human      0.05))
              (Object  (Cooking-Utensil 0.2)
                       (Eating-Utensil 0.2))
  :Refinement-Of (Dirty 1.0 (Planner Planner)
                 (Object Object))
  :Results-Of  ($Stove-Cooking 1.0 (Planner Actor)
                 (Object Object))
                 ($Baking      1.0 (Planner Actor)
                 (Object Object))

```

```

;*****
; ** Script $Burn-And-Inhale ("smoke"):
; **
; **   A person (Smoker) smokes a cigarette or marijuana (Drug) to give himself
; **   lung cancer (S-Lung-Cancer).
; **
; **   Preconditions The Drug is lit ($BAI-Burning).
; **   Results-In:   The Smoker has lung cancer (Lung-Cancer).

; **   Smoker:   The person (Human) doing the smoking.
; **   Drug:     The drug (Cigarette or Marijuana) that the Smoker is smoking.

(Role $Burn-And-Inhale^Smoker   :Prototypes (Human 0.05)
                                :Uses       ($Burn-And-Inhale Actor)
                                           ($BAI-Burning Planner))
(Role $Burn-And-Inhale^Drug     :Prototypes (Cigarette 0.5)
                                (Marijuana 0.5)
                                :Uses       ($Burn-And-Inhale Object)
                                           ($BAI-Burning Object))

(Frame $Burn-And-Inhale        Script
  :Roles (Actor $Burn-And-Inhale^Smoker)
         (Object $Burn-And-Inhale^Drug)
  :Phrase (<smoke> 0.5)
  :Precond ($BAI-Burning 0.8)
  :Results-In (Lung-Cancer 0.5 (Actor Object)))

(Frame $BAI-Burning            State
  :Roles (Planner $Burn-And-Inhale^Smoker)
         (Object $Burn-And-Inhale^Drug)
  :Refinement-Of (Burning 0.9 (Planner Planner)
                 (Object Object))
  :Precond-For ($Burn-And-Inhale 1.00))

```

```

;*****
; ** Script $Grow-Potted-Plant ("grow"):
; **
; **   A person (Planner) grows a plant (Plant) in a planting-pot (Pot).
; **
; **   Precondition: The Plant is inside the Pot (Inside-Of-$GP-Plant).
; **   Plan:         The Planner waters the Plant ($GP-Water-Plant).
; **
; ** Action $GP-Plant ("plant"):
; **   As part of $Grow-Potted-Plant, the Planner puts the Plant inside of
; **   the Pot. Results-In: Inside-Of-$GP-Plant.
; **
; ** State Inside-Of-$GP-Water:
; **   Watering the Plant ($GP-Water-Plant) requires that the Plant be inside of
; **   the Pot. Goal-For: $GP-Water-Plant.

; ** Planner: The person planning and doing the $Grow-Potted-Plant.
; ** Plant:   The plant being grown.
; ** Pot:     The pot the plant is being grown in.
; ** WaterR:  The water the plant is watered with.

(Role $Grow-Potted-Plant^Planner :Prototypes (Human 0.05)
                                :Uses       ($Grow-Potted-Plant Actor)
                                           ($GP-Plant Actor)
                                           ($GP-Water-Plant Actor)
                                           (Inside-Of-$GP-Plant Planner)
                                           (Inside-Of-$GP-Water Planner))

(Role $Grow-Potted-Plant^Plant  :Prototypes (Plant 0.5)
                                :Uses       ($Grow-Potted-Plant Object)
                                           ($GP-Plant Object)
                                           (Inside-Of-$GP-Plant Object)
                                           ($GP-Water-Plant Object))

(Role $Grow-Potted-Plant^Pot    :Prototypes (Planting-Pot 0.9)
                                :Uses       ($Grow-Potted-Plant Location)
                                           ($GP-Plant Location)
                                           (Inside-Of-$GP-Plant Location)
                                           (Inside-Of-$GP-Water Location))

(Role $Grow-Potted-Plant^WaterR :Prototypes (Water 0.05)
                                :Uses       ($Grow-Potted-Plant Water)
                                           ($GP-Water-Plant Inst)
                                           (Inside-Of-$GP-Water Object))

(Frame $Grow-Potted-Plant Script
  :Roles (Actor $Grow-Potted-Plant^Planner)
         (Object $Grow-Potted-Plant^Plant)
         (Location $Grow-Potted-Plant^Pot)
         (Water $Grow-Potted-Plant^WaterR)
  :Phrase (<grow> 1.00 (Actor Actor)
          (Object Object)
          (Location Location))
  :Precond (Inside-Of-$GP-Plant 1.00)
  :Plan    ($GP-Water-Plant 1.00))

(Frame $GP-Plant Action
  :Roles (Actor $Grow-Potted-Plant^Planner)
         (Object $Grow-Potted-Plant^Plant)
         (Location $Grow-Potted-Plant^Pot)
  :Phrase (<planted> 0.6)
  :Refinement-Of (Transfer-Inside 1.0 (Actor Actor)
                (Object Object)
                (Location Location))
  :Results-In (Inside-Of-$GP-Plant 1.00))

```

```

(Frame SGP-Water-Plant Plan
:Roles (Actor $Grow-Potted-Plant^Planner)
      (Object $Grow-Potted-Plant^Plant)
      (Inst $Grow-Potted-Plant^Waterr)
:Phrase (<watered> 0.4)
:Plan-For ($Grow-Potted-Plant 1.00)
:Goal (Inside-Of-$GP-Water 1.00))

(Frame Inside-Of-$GP-Plant State
:Roles (Planner $Grow-Potted-Plant^Planner)
      (Object $Grow-Potted-Plant^Plant)
      (Location $Grow-Potted-Plant^Pot)
:Refinement-Of (Inside-Of 1.0 (Planner Planner)
              (Object Object)
              (Location Location))
:Results-Of ($GP-Plant 1.00)
:Precond-For ($Grow-Potted-Plant 1.00))

(Frame Inside-Of-$GP-Water State
:Roles (Planner $Grow-Potted-Plant^Planner)
      (Object $Grow-Potted-Plant^Waterr)
      (Location $Grow-Potted-Plant^Pot)
:Refinement-Of (Inside-Of 1.0 (Planner Planner)
              (Object Object)
              (Location Location))
:Goal-For ($GP-Water-Plant 1.00)
:Precond-For ($Grow-Potted-Plant 1.00))

```

```

;*****
; ** N-MOP Police-Capture:
; **
; ** The police (Police) capture a criminal (Criminal), based on the
; ** evidence (Evidencer). If they are successful, then he will be in jail.
; **
; ** Precondition: The Police know about the crime that the Criminal has
; ** committed (Police-Know-Crime).
; ** Plan: The Police arrest the Criminal (Police-Arrest).
; ** Results-In: The Criminal is in Jail (S-In-Jail).
; **
; ** Action Police-Know-Crime:
; ** The Police know about the crime that the Criminal has committed.
; **
; ** Precondition: Criminal committed a crime (Crime-Committed).
; ** Plan: The Police see the Evidencer (Police-See-Evidence).
; **
; ** Action Police-Arrest ("arrest"):
; ** The Police arrest the Criminal.
; **
; ** Action Police-See-Evidence:
; ** The Police see the Evidencer of the crime that the Criminal committed.
; **
; ** Action Crime-Committed:
; ** The Criminal commits a crime using the Evidencer.
; **
; ** Instances: The Criminal possesses something illegal (Possess-Illegal-Obj)
; ** The Criminal shoots somebody ($Shoot-Person).
; **
; ** Police: The police trying to do the capturing.
; ** Criminal: The criminal the police are trying to catch.
; ** Evidencer: The evidence (Weapon, Illegal-Possess-Obj) the police use to
; ** nail the Criminal.

(Role Police-Capture^Police :Prototypes (Police 0.7)
:Uses (Police-Capture Actor)
(Police-Know-Crime Actor)
(Police-Arrest Actor)
(Police-See-Illegal Actor))

(Role Police-Capture^Criminal :Prototypes (Human 0.05)
:Uses (Police-Capture Object)
(Police-Know-Crime Object)
(Police-Arrest Object)
(Crime-Committed Actor))

(Role Police-Capture^Evidencer :Prototypes (Illegal-Possess-Obj 0.1)
(Weapon 0.1)
:Uses (Police-Capture Evidence)
(Police-Know-Crime Evidence)
(Police-See-Illegal Evidence)
(Crime-Committed Evidence))

(Frame Police-Capture N-MOP
:Roles (Actor Police-Capture^Police)
(Object Police-Capture^Criminal)
(Evidence Police-Capture^Evidencer)
:Precond (Police-Know-Crime 1.00)
:Plan (Police-Arrest 1.00)
:Results-In (In-Jail 1.0 (Actor Planner)
(Object Object)))

```

```

(Frame Police-Know-Crime Action
:Roles (Actor Police-Capture^Police)
      (Object Police-Capture^Criminal)
      (Evidence Police-Capture^Evidencer)
:Precond-For (Police-Capture 1.00)
:Precond (Crime-Committed 0.50)
:Plan (Police-See-Illegal 1.00))

(Frame Police-Arrest Action
:Roles (Actor Police-Capture^Police)
      (Object Police-Capture^Criminal)
:Phrase (<arrest> 1.00)
:Plan-For (Police-Capture 0.75))

(Frame Police-See-Illegal Action
:Roles (Actor Police-Capture^Police)
      (Evidence Police-Capture^Evidencer)
:Refinement-Of (See-Object 1.0 (Actor Actor)
              (Evidence Object))
:Plan-For (Police-Know-Crime 0.5))

(Frame Crime-Committed Action
:Roles (Actor Police-Capture^Criminal)
      (Evidence Police-Capture^Evidencer)
:Precond-For (Police-Know-Crime 1.00)
:Instances (Possess-Illegal-Obj 1.0 (Actor Actor)
          (Evidence Object))
          ($Shoot-Person 1.0 (Actor Actor)
          (Evidence Inst)))

(Frame Possess-Illegal-Obj State
:Roles (Actor (Human 0.05))
      (Object (Illegal-Possess-Obj 0.5))
:Refinement-Of (Possess-Obj 1.0 (Actor Actor)
              (Object Object))
:Is-A (Crime-Committed 0.5 (Actor Actor)
      (Object Evidence)))

```

```

;*****
;** Script $Shoot-Person ("shoot"):
;**
;**   A person (Shooter) shoots another person (Shootee) with a gun (Firearm)
;** to kill him.
;**
;**   Results-In: The Shootee is dying (S-Dying).

```

```

(Frame $Shoot-Person Script
  :Roles      (Actor (Human 0.5))
              (Object (Human 0.05))
              (Inst  (Firearm 0.2))
  :Phrase     (<shoot> 1.00)
  :Is-A       (Crime-Committed 0.5 (Actor Actor)
              (Inst Evidence))
  :Results-In (Dying              0.05 (Actor Planner)
              (Object Object))

```

```

;*****
;** Script $Smoke-Food ("smoke"):
;**
;**   A person (Actor) smokes meat (Object) to prepare it.
;**
;**   Results-In: The meat Object is cooked (S-Food-Prepared).

```

```

(Frame $Smoke-Food      Action
  :Roles      (Actor (Human 0.05))
              (Object (Meat 0.33))
  :Phrase     (<smoke> 0.5)
  :Results-In (Food-Prepared 0.5 (Actor Planner)
              (Object Object))

```

```

;*****
;**                                     **
;*****
;*****
;Action Do-Action-To-Object: A person (Actor) does some kind of action to an
; object (Object). This implies that the person possesses that object.

(Frame Do-Action-To-Object Action
  :Roles      (Actor (Human 0.05))
              (Object (Phys-Obj 0.05))
  :Refinements (Transfer-Inside 1.0 (Actor Actor)
                                   (Object Object))
              (Transfer-On-Top 1.0 (Actor Actor)
                                   (Object Object))
  :Implies    (Possess-Obj 0.1 (Actor Actor)
              (Object Object)))

;*****
;Plan Avoid-Detection ("hide"): A human (Actor) wants to hide an object
; (Object). He thus has the goal of the Object being blocked from
; sight (Block-See).

(Frame Avoid-Detection Plan
  :Roles (Actor (Human 0.05))
        (Object (Phys-Obj 0.05))
  :Phrase (<hide> 1.0 (Actor Actor)
        (Object Object))
  :Goal (Block-See 0.2 (Actor Planner)
        (Object Object)))

;*****
;Action Transfer-Inside ("put_inside"): A person (Actor) puts an object
; (Object) into a container (Location). Results in the object being
; inside of the container (Inside-Of).

(Frame Transfer-Inside
  Action
  :Roles (Actor (Human 0.05))
        (Object (Phys-Obj 0.05))
        (Location (Container-Obj 0.05))
  :Phrase (<put_inside> 1.0 (Actor Actor)
        (Object Object)
        (Location Location))
  :Refinement-Of (Do-Action-To-Object 0.1 (Actor Actor)
        (Object Object))
  :Refinements ($GP-Plant 1.0 (Actor Actor)
        (Object Object)
        (Location Location))
  :Results-In (Inside-Of 0.6 (Actor Planner)
        (Object Object)
        (Location Location)))

```



```

;*****
;Action Transfer-On-Top ("put_on_top_of"): A person (Actor) puts an object
; (Object) on top of another object (Location). Results in the object being
; on top of the container (S-On-Top-Of).

```

```

(Frame Transfer-On-Top
  Action
  :Phrase      (<put_on_top_of> 1.00)
  :Roles       (Actor   (Human 0.05))
                (Object  (Phys-Obj 0.05))
                (Location (Container-Obj 0.05))
  :Refinement-Of (Do-Action-To-Object 0.1 (Actor Actor)
                                                (Object Object))
  :Results-In   (On-Top-Of 0.6 (Actor Planner)
                                                (Object Object)
                                                (Location Location))

```

```

;*****
;Action Transfer-Self ("come"): A person (Actor) transfers himself to Location.
; Results in the person being at the location (S-Prox).

```

```

(Frame Transfer-Self  Action
  :Roles      (Actor   (Human 0.05))
                (Location (Phys-Obj 0.05))
  :Phrase     (<come> 1.0 (Actor Actor)
                (Location Location))
  :Results-In (Proximity-Of 0.5 (Actor Object)
                (Location Location))

```

```

;*****
;Action See-Object ("see"): An animate (Actor) sees an object (Object).
; Precondition: The animate is near the object (Proximity-Of).
; Disabled-By: Disabled if the object cannot be seen (Block-See).

```

```

(Frame See-Object  Action
  :Phrase      (<see> 1.00)
  :Roles       (Actor   (Animate 0.05))
                (Object  (Phys-Obj 0.05))
  :Precond     (Proximity-Of 1.0 (Actor Object)
                (Object Location))
  :Disabled-By (Block-See 1.0 (Object Object))
  :Refinements (Police-See-Illegal 1.0 (Actor Actor)
                (Object Evidence))

```

```

;*****
;**                                     General State Frames                                     **
;*****

```

```

;*****
;State Proximity-Of ("is_next_to"): A person (Actor) is near an object
; (Location).

```

```

(Frame Proximity-Of State
  :Roles      (Object (Human 0.05))
              (Location (Phys-Obj 0.05))
  :Phrase     (<is_next_to> 1.00)
  :Precond-For (See-Object 1.0 (Object Actor)
              (Location Object))
  :Results-Of (Transfer-Self 1.0 (Object Actor)
              (Location Location))

```

```

;*****
;State Inside-Of ("is_inside_of"): An object (Object) is inside of a container
; (Location). A human Planner caused this.

```

```

(Frame Inside-Of State
  :Roles      (Planner (Human 0.05))
              (Object (Phys-Obj 0.05))
              (Location (Container-Obj 0.5))
  :Phrase     (<is_inside_of> 1.0 (Object Object)
              (Location Location))
  :Refinements (Inside-Of-Dishwasher 1.0 (Planner Planner)
              (Object Object)
              (Location Location))
              (Inside-Of-Stove 1.0 (Planner Planner)
              (Object Object)
              (Location Location))
              (Inside-Of-$GP-Plant 1.0 (Planner Planner)
              (Object Object)
              (Location Location))
              (Inside-Of-$GP-Water 1.0 (Planner Planner)
              (Object Object)
              (Location Location))
              (Inside-Of-Opaque 1.0 (Planner Planner)
              (Object Object)
              (Location Location))
  :Results-Of (Transfer-Inside 1.0 (Planner Actor)
              (Object Object)
              (Location Location))

```

```

;*****
;State On-Top-Of ("is_on_top_of"): An object (Object) is on top of an object
; (Location). A human Planner caused this.

```

```

(Frame On-Top-Of State
  :Roles      (Planner (Human 0.05))
              (Object (Phys-Obj 0.05))
              (Location (Phys-Obj 0.05))
  :Phrase     (<is_on_top_of> 1.0 (Object Object)
              (Location Location))
  :Refinements (On-Top-Of-Stove 1.0 (Planner Planner)
              (Object Object)
              (Location Location))
              (On-Top-Of-Heat-Source 1.0 (Planner Planner)
              (Object Object)
              (Location Location))
  :Results-Of (Transfer-On-Top 1.0 (Planner Actor)
              (Object Object)
              (Location Location))

```

```

;*****
;State On-Top-Of-Heat-Source: An flammable object (Object) is on top of a
; heat source (Location). A human Planner caused this.
; Results-In: The flammable object burning (Burning).

```

```

(Frame On-Top-Of-Heat-Source State
  :Roles      (Planner (Human 0.05))
              (Object (Flammable-Obj 0.5))
              (Location (Heat-Source 0.5))
  :Phrase     (<lit> 1.0 (Planner Actor)
              (Object Object)
              (Location Location))
  :Refinement-Of (On-Top-Of 1.0 (Planner Planner)
              (Object Object)
              (Location Location))
  :Results-In  (Burning 0.7 (Planner Planner)
              (Object Object))

```

```

;*****
;State Burning ("is_lit"): An flammable object (Object) is burning. A human
; Planner caused this.

```

```

(Frame Burning State
  :Roles      (Planner (Human 0.05))
              (Object (Flammable-Obj 0.5))
  :Phrase     (<is_lit> 0.6 (Object Object))
  :Refinements ($BAI-Burning 1.0 (Planner Planner)
              (Object Object))
  :Results-Of (On-Top-Of-Heat-Source 1.0 (Planner Planner)
              (Object Object))

```

```

;*****
;State Possess-Obj ("has"): A human (Actor) is in possession of an
; object (Object)

(Frame Possess-Obj State
  :Roles      (Actor (Human 0.05))
              (Object (Phys-Obj 0.05))
  :Phrase     (<has> 1.00)
  :Refinements (Possess-Illegal-Obj 1.0 (Actor Actor)
              (Object Object))
  :Implied-By (Do-Action-To-Object 0.8 (Actor Actor)
              (Object Object))

;*****
;State Clean ("is_clean"): An object (Object) is clean.
; A human Planner caused this.

(Frame Clean State
  :Roles      (Planner (Human 0.05))
              (Object (Phys-Obj 0.05))
  :Phrase     (<is_clean> 1.0 (Object Object))
  :Refinements (Clean-Dishes 1.0 (Planner Planner)
              (Object Object))
  :Goal-For   (Dirty 1.0 (Planner Planner)
              (Object Object))

;*****
;State Dirty ("is_dirty"): An object (Object) is dirty.
; A human Planner caused this.

(Frame Dirty State
  :Roles      (Planner (Human 0.05))
              (Object (Phys-Obj 0.05))
  :Phrase     (<is_clean> 1.0 (Object Object))
  :Refinements (Dirty-Dishes 1.0 (Planner Planner)
              (Object Object))
  :Goal       (Clean 1.0 (Planner Planner)
              (Object Object))

;*****
;State Food-Prepared ("is_ready"): Some food (Object) is ready to be eaten.
; A human Planner caused this.

(Frame Food-Prepared State
  :Roles      (Planner (Human 0.05))
              (Object (Edible-Obj 0.05))
  :Phrase     (<is_ready> 0.8 (Object Object))
  :Results-Of ($Stove-Cooking 1.0 (Planner Actor)
              (Object Object))
              ($Baking 1.0 (Planner Actor)
              (Object Object))
              ($Smoke-Food 1.0 (Planner Actor)
              (Object Object))

```

```

;*****
;State Lung-Cancer ("has_cancer"): An animate (Object) has lung cancer.
; Results-In: The animate is dying (S-Dying)

(Frame Lung-Cancer State
:Roles (Object (Animate 0.05))
:Phrase (<has_cancer> 1.00 (Object Object))
:Results-Of (SBurn-And-Inhale 0.70 (Object Actor))
:Results-In (Dying 0.05 (Object Object)))

;*****
;State In-Jail ("is_in_jail"): An human (Object) is in jail.
; A human (Planner) planned this.

(Frame In-Jail State
:Roles (Planner (Human 0.05))
(Object (Human 0.05))
:Phrase (<is_in_jail> 1.0 (Object Object))
:Results-Of (Police-Capture 0.8 (Planner Actor)
(Object Object))

;*****
;State Dying ("is_dying"): An animate (Object) is dying.
; A human (Planner) may or may not have planned this.

(Frame Dying State
:Roles (Planner (Human 0.05))
(Object (Animate 0.05))
:Phrase (<is_dying> 1.0 (Object Object))
:Results-Of ($Shoot-Person 0.9 (Planner Actor)
(Object Object))
(Lung-Cancer 0.9 (Planner Object)
(Object Object))

;*****
;State Inside-Of-Opaque: An object (Object) is inside of an opaque container
; object (Location). A human (Planner) planned this.
; Results-In: Nobody can see the object (Block-See).

(Frame Inside-Of-Opaque State
:Roles (Planner (Human 0.05))
(Object (Phys-Obj 0.05))
(Location (Opaque-Obj 0.05))
:Refinement-Of (Inside-Of 0.9 (Planner Planner)
(Object Object)
(Location Location))
:Results-In (Block-See 0.7 (Planner Planner)
(Object Object))

```

```

;*****
;State Block-See: An object (Object) cannot be seen. A human (Planner)
; planned this.
; Disables: Anybody from seeing the object (See-Object).

(Frame Block-See      State
                    :Roles      (Planner (Human 0.05)
                                   (Object (Phys-Obj 0.05))
                    :Results-Of (Inside-Of-Opaque 1.0 (Planner Planner)
                                                         (Object Object))
                    :Disables   (See-Object 0.2 (Object Object))
                    :Goal-For   (Avoid-Detection 1.0 (Planner Actor)
                                   (Object Object)))

```