A Dynamic Connectionist Model Of Problem Solving

James Lundell and Bernice Laden
University of Washington

Connectionist computing offers a flexible approach to modelling automatic cognitive processes, while production systems provide reasonable models of controlled processes involved in problem solving. Our initial motivation for the development of this problem solving simulation was to build a model which could predict the time required to solve a problem and the types of errors people are likely to make under time pressure. Aside from its predictive ability, we believe the simulation offers a unique approach to modelling cognitive tasks. We refer to the model as a Dynamic Connectionist Model of Problem Solving (DCOMPS). It is connectionist because working memory consists of propositions which are organized into a connected activational network, and it is dynamic because this network changes its organization over time as a result of the firing of production rules. Its architecture is designed to handle a variety of problem solving tasks, including arithmetic word problems, two-term series and a stroop task.

Our purpose in this project is to develop a model of problem solving that predicts the time required to solve a problem and the errors which people are likely to make under time pressure. We assume that problem solving occurs as a series of mental events, each of which consists of the recognition of a pattern and a response. The time required to respond to a pattern is a function of the interaction between automatic and controlled processes (Neely, 1977). The two most commonly used architectures in modelling cognitive processes, connectionist networks and production systems, closely parallel automatic and controlled processes, respectively. Connectionist networks are used to simulate the pattern-matching capabilities of the cognitive system - they are particularly good at resolving incomplete or ambiguous patterns. These networks, however, do not easily accomodate the goal-driven, serial nature of human cognitive processing. Production systems, on the other hand, are goal-driven and serial in nature. Connectionist networks do not easily deal with instantiation of several concurrent concepts; production systems can accommodate instantiation and variable binding.

We would like to thank Earl Hunt, Aura Hanna, Simon Farr and Penny Yee for their comments on earlier drafts. This research was supported by a grant from the Office of Naval Research Grant N0014-84-K-003 to the University of Washington, Earl Hunt Principal Investigator.

Taken together, production systems and connectionist models compensate for each others' weaknesses. The human cognitive system may have evolved these two types of processing primarily because they compensate for each others' deficiencies. Thus there are theoretical as well as pragmatic reasons for attempting to integrate the two approaches to cognitive modelling.

Our work builds on two attempts to combine models of automatic and controlled processing: Anderson's ACT* (1983) and Hunt and Lansman's (1986) production-activation model. In the ACT* approach, production rules are embedded in a connected network, and the rules fire when they attain a sufficient level of activation.

Hunt and Lansman's theory is similar to ACT* but focuses upon simple cognitive tasks such as the Stroop task, four and eight-choice reaction time tasks, and has been extended to simple arithmetic tasks (Richardson and Hunt, 1986). Production rules are matched to features in working memory. A production rule is fired when activation strength exceeds that of all competing productions by a parameter *delta*. The time required for a given production to fire is a decreasing function of the degree of match between the production pattern and the contents of working memory, and an increasing function of the number of competing productions.

Our simulation is based upon the Hunt and Lansman system, but has several basic changes. Hunt and Lansman represented objects as vectors of features; we chose a propositional representation. In the Hunt and Lansman model, activation spreads through a network of production rules. In our model, activation spreads through a network of propositions in which the links between the propositions are determined by a few general principles, and can be modified by production rules.

DCOMPS: An Overview

The program is called DCOMPS, which stands for Dynamic Connectionist Model of Problem Solving. DCOMPS is based upon three assertions: 1) automatic processes can be modelled by the parallel activation of related nodes in a propositional network, 2) controlled processes can be modelled by a production system and, 3) the time required to solve a problem is a function of the interaction between the two processes.

The following points describe some of the more salient features of DCOMPS:

First, we have loosely adapted the system described by Kintsch (1974), for the propositional representation of problems. Complex problem solving tasks are represented by simple propositions, where a proposition consists of a predicate and one or more arguments.

Second, following Hunt and Lansman we distinguish between internal and external channels. Working memory consists of a semantic (internal) channel and one or more external channels. Each channel may contain several linked propositions. Each proposition has an activation level that is determined by the activity in the network. Representations in the external channel have been processed to a certain extent, but have not been semantically encoded. In the current version, there is only one external channel which receives propositional representations of visually presented words or sentences. Other external channels, such as auditory and tactile channels, could be added. Connections across channels are allowed, although they are generally weaker than within-channel connections.

Third, control of the problem solving process is governed by the pattern of activation of propositions resident in working memory over time. This in turn is governed by simple rules which dictate how propositions may be linked together, and by the incorporation and transformation of propositions in working memory.

The incorporation of propositional information is based on Kintsch and van Dijk's (1978) model of discourse understanding. Kintsch and van Dijk distinguish between a microstructure and a macrostructure representation of text. The microstructure consists of the part of the discourse which is currently being comprehended. The macrostructure consists of propositions which have been extracted from the microstructure and which are retained in working memory as an aid to understanding incoming propositions. Because we eventually intend to simulate many different types of visual and auditory tasks, we have chosen to call the macrostructure and microstructure the semantic and external channels, respectively. Propositions are extracted from the external channel and incorporated into the semantic channel based on the concept of *referential coherence*. That is, the propositions which contain arguments that are most often referred to are extracted first, while other propositions tend to be discarded. Kintsch (personal communication, July, 1986) has suggested how this theory may be implemented in a connectionist framework.

Propositions are linked to themselves by a connection strength of 1.0. Propositions that share arguments are linked by a connection strength of 0.5. Propositions that are both referenced by a third proposition are linked by a strength of 0.3. For example, P1(agent (John)) and P3 (Object (ball)) would be linked to each other by a strength of 0.3 if proposition P2(has (P1 P3)) were also present. Propositions that are related more distantly are linked by a connection strength of 0.2. For example, if P4 were (Color (red P3)), it would be linked to P1 by 0.2. Thus for the propositions P1-P4 we have the following connection matrix:

	P1	P2	P3	P4
P1	1.0	0.5	0.3	0.2
P2	0.5	1.0	0.5	0.3
P3	0.3	0.5	1.0	0.5
P4	0.2	0.3	0.5	1.0

We have chosen these link strengths arbitrarily. The actual values themselves are not as important as the idea that the connection strengths decrease with the distance from related propositions. Propositions placed in the external channel are given an initial activation value of 1.0. Each proposition's activation value at time t is computed synchronously by the following formula:

$$A_{nt} = \Sigma A_{i(t-1)} C_{i/N}$$

where A_{nt} is the activation of proposition n at time t, $A_{i(t-1)}$ is the activation of proposition i at time t-1, and C_i is the connection strength between proposition n and proposition i. N is a normalizing factor which prevents activation in the network from becoming unacceptably high. The activation level of each node in the network is iteratively computed until the network becomes sufficiently "stable", i.e. when the change in activation levels for each node drops below a criterion amount. Once the network reaches stability, the propositions which are most highly referenced are the most active. In this way, a hierarchy of activation levels is formed, and the least active propositions may be dropped from the network. The most active propositions will then be transformed into a semantic code and placed into the semantic channel. The macrostructure will eventually contain a parsimonious representation of the topic of discourse, based on the concept of referential coherence.

As Kintsch and van Dijk (1978) have noted, referential coherence only partially accounts for discourse comprehension. Since comprehension requires the additional use of information resident in long term memory, the system contains two other types of nodes: schema nodes and rule nodes. These can be conceived of as knowledge the system already has. Schema nodes reside in the semantic channel of working memory where they differentially activate certain types of propositions, and thus help control the activation of the important aspects of a statement. For example, if we were to read *John has three dollars* in the context of an arithmetic problem, we would probably attend to the quantity *three*. This is because our schema for arithmetic problems activates all propositions which pertain to numbers. If we were to read the same sentence at a time when we happen to need money, we would attend more strongly to the object of the sentence, *dollars*.

Rules are part of the knowledge base of the system, and are linked to classes of propositions or schema nodes which appear in

working memory. The activation algorithm for rules is simple: rules inherit the activation of the nodes to which they are connected. All rules are negatively linked to all other rules by a value of -0.5. Each time the network stabilizes, the most active rule fires. When a rule fires, it may change the connection strength between propositions or it may insert new propositions into working memory. This method of inserting new propositions or 'nodes' into the network constitutes a break with traditional connectionist modeling.

A Specific Example: The Two-term Series Problem

The DCOMPS' architecture will be illustrated by showing how the simulation works on two-term series problems.The two-term series problem is a deductive reasoning task containing a relational statement and a question. An example is: *John is better than Pete. Who is best?* In this type of problem people make inferences based upon linguistic, rather than logical interpretations. Performance has been shown to be influenced by three factors: 1) lexical marking, 2) use of negations and 3) statement-question congruence (Clark,1969). We have incorporated these aspects into the simulation.

TABLE 1

Eight Versions of the Two-Term Series Problem

	Proposition	Question
Unmarked	John is better than Pete.	Who is best?
	John is better than Pete.	Who is worst?
	Pete is not as good as John.	Who is best?
	Pete is not as good as John.	Who is worst?
Marked	Pete is worse than John.	Who is best?
	Pete is worse than John.	Who is worst?
	John is not as bad as Pete.	Who is best?
	John is not as bad as Pete.	Who is worst?

There are eight versions of the two-term series problem if one takes into consideration lexical marking, negation and congruence. These are summarized in Table 1. We have selected one of them to illustrate the details of our model. In text form the problem is:

John is worse than Pete. Who is best? In propositional form the text is represented as follows:

P1(Agent John)
P2(Worse_than P1 P3)
P3(Agent Pete)
P4(Question P5 P6)
P5(Agent unknown)
P6(Best P5)

Initially, working memory contains a reasoning schema and an instruction node, I1, in the semantic channel. The semantic channel is organized as depicted in Figure 1. I1 is the initial instruction to read the problem. S1 is linked to the rule R1. When R1 is fired the system will read the first sentence of the problem. S2 is linked to rule R2 which when fired will attempt to solve the problem. S4 is linked to R5. R5 will be fired if the text contains a negation, such as *John is not as good as Pete*, and will transform those propositions to a positive interpretation, *Pete is better than John*. S3 is a special node which will look for any relational propositions, such as "Better_than" or "Worse_than" in the external channel and link itself to that proposition. Thus any relational propositions will tend to be more active, and so will receive a greater amount of 'attention'.

S0 is initialized with an activation level of 1.0, and the system iterates until the network stabilizes. All positive links in the superschema are given a value of 0.5, and all negative links are given a value of -0.5. When the system settles, rule R1, read_sentence, is highly active. The system reads in the first sentence, *John is worse than Pete* by placing the first three propositions in the external channel. The worse_than relationship is lexically marked relative to the better_than relationship. In the current example, lexical marking is simulated through the addition of inferences that are made by the system whenever a worse_than relationship is encountered. DCOMPS infers that Pete is bad, and that John is very bad. These inferences are constructed as propositions and linked to P1(Agent John) and P3(Agent Pete) respectively. All inference propositions are linked to their related propositions by a connection strength of 0.2. Inferred propositions are given an initial activation level of 0. According to Clark, lexically marked adjectives take longer to recover from memory. In our interpretation, lexically marked adjectives are adjectives which cause a number of inferences to be made. This means that a sentence containing a lexically marked adjective will link to more (inferred) propositions. It will take longer, on average, for the larger network to stabilize.

At this point, node S3 in the term schema is linked to proposition P2, (Worse_than P1 P3). In order to keep the computational overhead down, rules from the rulebase are only added

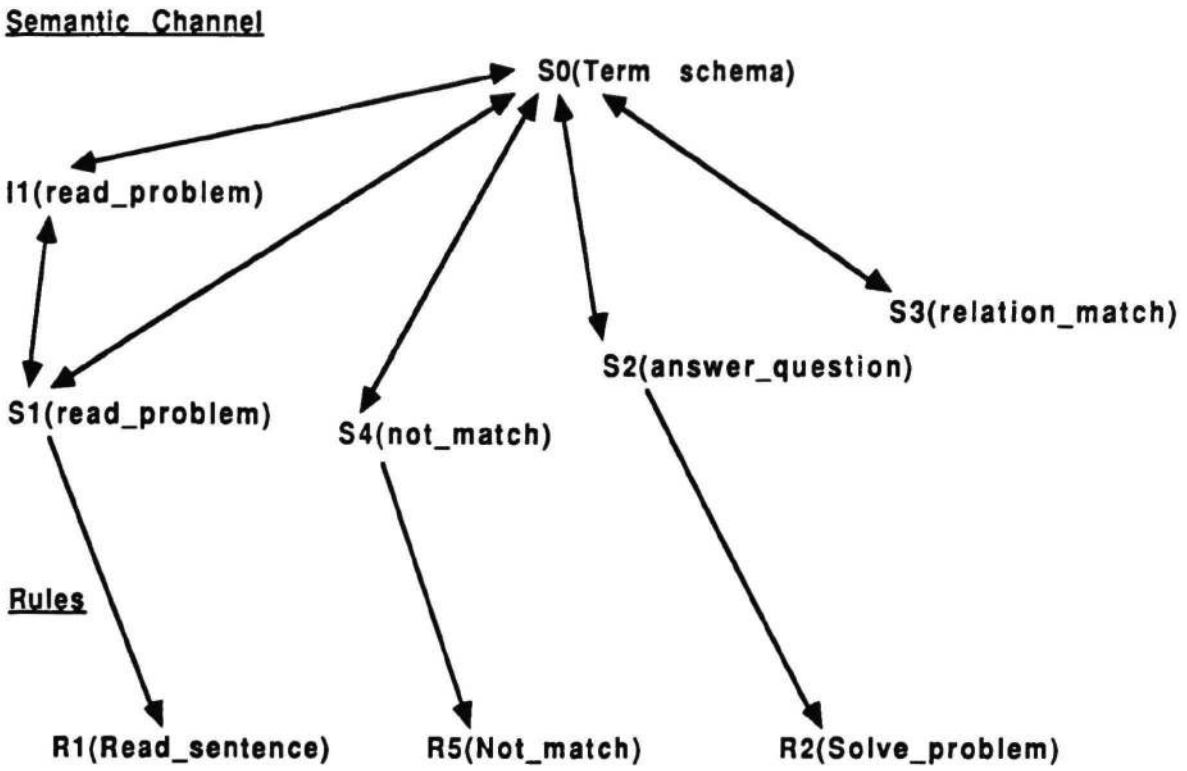


Figure 1. Initially, working memory contains a schema organized in the semantic channel.

to the network if there is a corresponding proposition match in working memory. Rule R4, worse_than_relation, is added to the network because it now has a corresponding match in P2. The state of the network at this point is described in Figure 2.

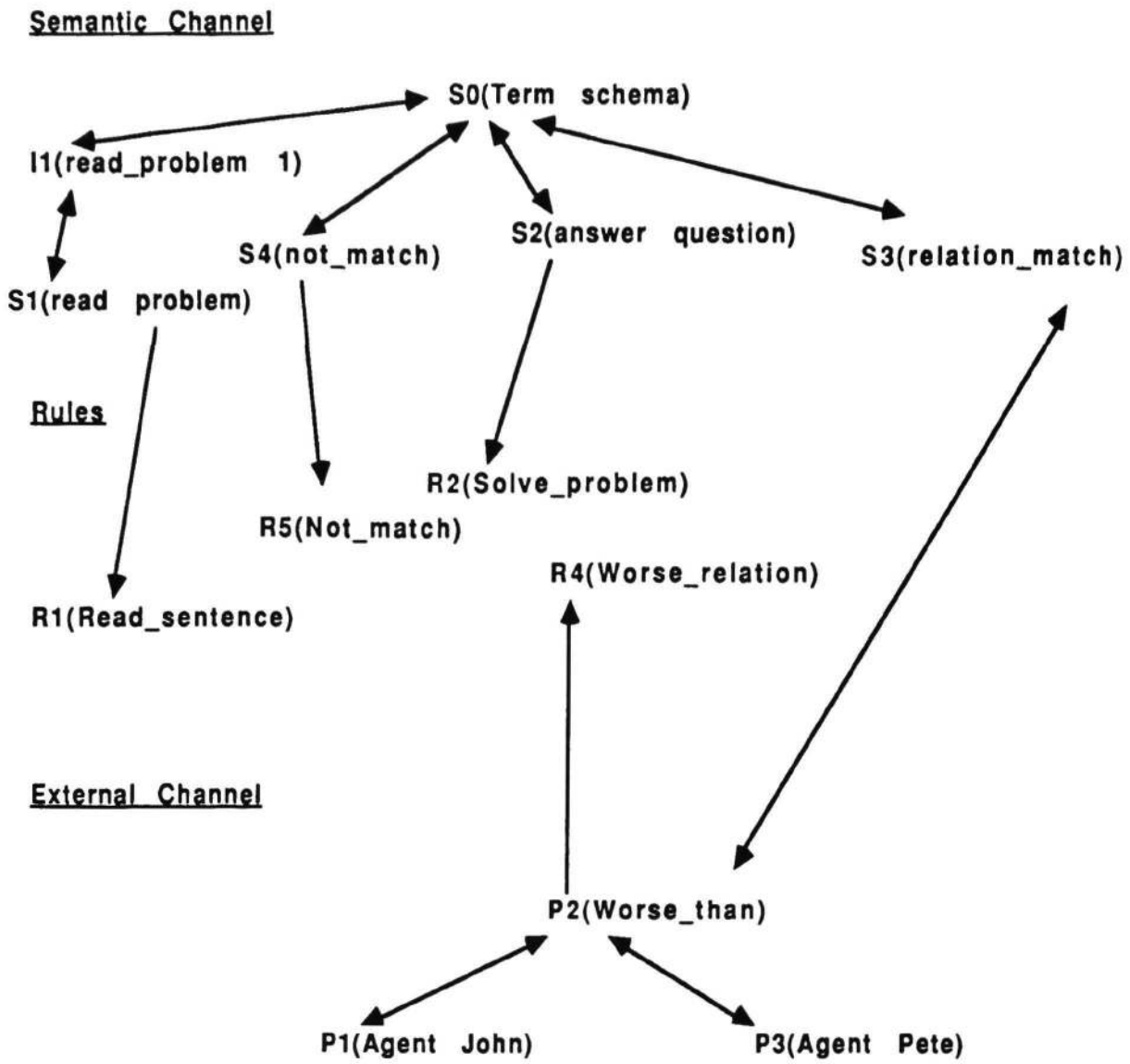
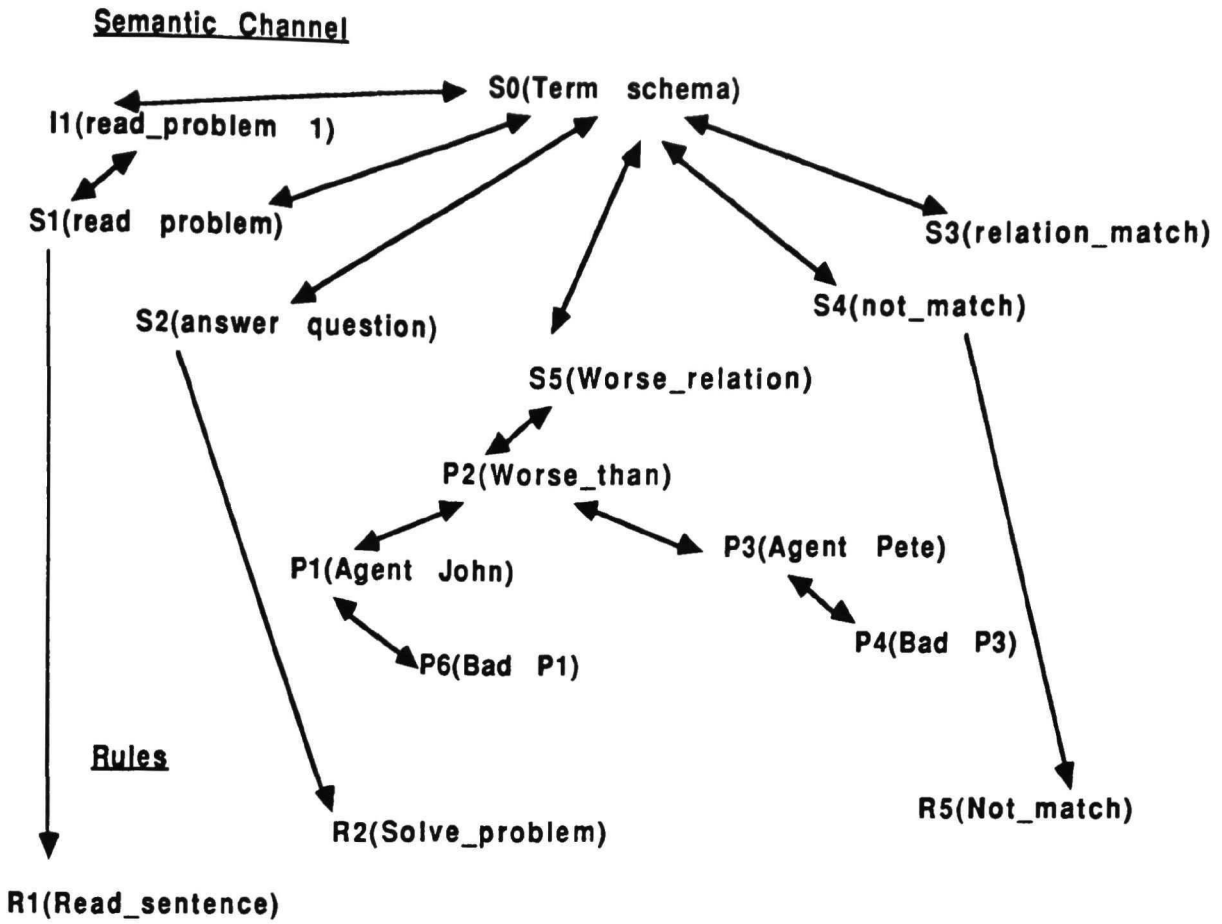


Figure 2. The first sentence is read in. The relational node of the schema, S3 is linked to the proposition P2. Rule R4 is added to the network.

The system again cycles through the activation updating function until the network stabilizes. This time R4 fires, which does two things: it adds a proposition to the semantic channel asserting that this first sentence contains a Worse_than relation, and it incorporates the propositions P1-P3 into the schema by changing the link strengths between the propositions to 0.3 if one proposition is contained in another's argument list, and 0.2 if two propositions share a common argument. Now the schema looks like Figure 3.



External Channel

Empty

Figure 3. Both text and inferred propositions are incorporated into the schema.

The presence of the term schema was necessary for proper understanding of the first sentence. The schema node S3 directed attention towards the relation stated in the text. As a result, R4 fired and created a new proposition (S5) linking P2 and S0. Without the schema the model would not have been able to extract information from the sentence which is crucial to solving the two-term series problem.

The connection strengths between the propositions are lowered once the propositions are incorporated into the semantic channel.

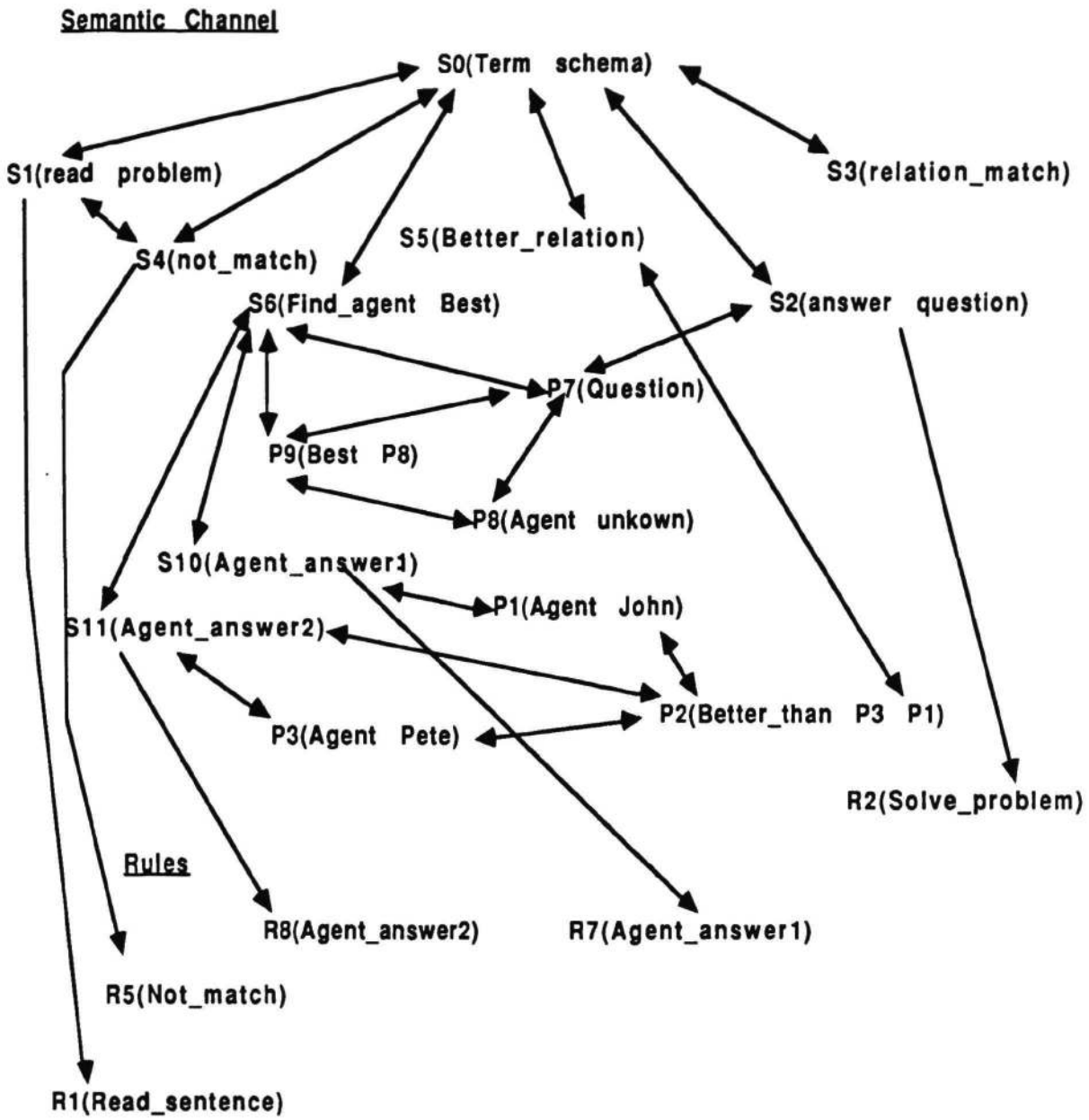
This is a decay mechanism used in the simulation which we found was necessary to allow the system to attend to new propositions placed on the external channel.

During the next cycle the system proceeds as before which eventually results in the rule R1 firing. The system places the next sentence, *Who is best?*, into the visual channel, setting all activation values for P7-P9 to 1.0. Rule R6(Quest_rule) is linked into the network. Rule R6(Quest_rule) wins out, which creates proposition S6(Find_agent Best). I1, the instruction to read, is removed from the network because the Quest_rule proposition is a signal that this is the last sentence in the problem.

The system cycles through again, now firing R2 because proposition I1 has been removed from the network. Rule R2 is a complex rule which forms various hypotheses about the answer. At this point three hypotheses are possible: *John is best, Pete is best* or the statement is incongruent with the question and must be converted to the proper representation. These hypotheses are linked into the network as S7, S8 and S9. DCOMPS gathers evidence for each hypothesis by linking each one to the propositions in the semantic channel which support it. In this case there are no propositions which support the John and Pete hypotheses, because the proposition terms such as P4(Bad P1) are incongruent with the question, *Who is best*. Hypothesis S9 is most active because there is a worse_than relation and a best question present in the current representation, signalling that a conversion is necessary.

Three rules are linked to the three newly formed hypotheses. As with all other rules which are present in the network, inhibitory links are created between them. When the network stabilizes, rule R9(Convert) fires. This rule will reformat the propositions created by the first sentence so that the relation is congruent with the question. The convert rule changes *worse_than* to *better_than* in P2, as well as S5. It also switches the agents, John and Pete, so that the initial sentence now has the interpretation *Pete is better than John*. The "bad" inferences drop out of working memory as well as the convert hypothesis, S9.

Now that the problem is represented correctly, the conversion rule no longer applies. Two hypotheses remain: John is best or Pete is best. These hypotheses can now gather evidence because the worse_than proposition has been converted to better_than. Figure 4 depicts how these hypothese are now linked to working memory. Two rules, R7(Agent1) and R8(Agent2) are added to the rule list. R8 is most active, since it is linked to P1 (Agent Pete), and P2(better_than P1 P3). Thus, *Pete is best* is determined to be the answer.



External Channel

Empty

Figure 4. Hypotheses linked into working memory

How does DCOMPS compare to empircal results?

Table 2 presents a comparison of the original Clark data and the number of cycles required for DCOMPS to solve the same problem. There is a Spearman correlation of .994 between the rank order of the times required for solution by human subjects and by DCOMPS. A tie occurs between the times required to solve *John is worse than Pete. Who is best?* and *John is not as good as Pete. Who is best?* We attribute this to the fact that DCOMPS takes less time to process *not* statements than do people. Because *John is not as good as Pete. Who is best?* is the quickest of the negatively framed problems and *John is worse than Pete. Who is best?* is the longest of the positively framed problems, a tie occurs between the times required to solve these two problems by DCOMPS.

TABLE 2

A comparison between data from Clark (1969) and DCOMPS for performance on two-term series problems.

Problem	Question	Clark secs.	DCOMPS cycles
John is better than Pete.	Who is best?	.61	91
John is worse than Pete.	Who is worst?	.62	95
John is better than Pete.	Who is worst?	.68	108
John is worse than Pete.	Who is best?	1.00	113
John is not as good as Pete.	Who is best?	1.17	113
John is not as good as Pete.	Who is worst?	1.47	128
John is not as bad as Pete.	Who is worst?	1.58	130
John is not as bad as Pete.	Who is best?	1.73	147

The time required to solve these problems is a function both of controlled, serial processing and automatic, parallel processing. That is, problems involving incongruent questions or negative assertions require the firing of additional productions which transform the appropriate propositions. The time required for these factors is a result of serial operations, and can be predicted by a simple production system model. However, the difference between solving problems with marked and unmarked adjectives is a result of the parallel activation of the inferred propositions which were incorporated into the network. During some steps of the problem solving process, it takes longer for the network to "settle" when these additional propositions are present.

Conclusion

DCOMPS currently performs two other tasks: arithmetic word problems and Stroop tasks. The system solves word arithmetic problems in a manner similar to two-term series problems. DCOMPS begins with an arithmetic schema which aids in interpreting sentences which appear on the external channel. In particular, the arithmetic schema differentially activates any *Quantity* propositions. As each sentence is read, rules fire which incorporate the propositions into the schema, and classify the propositions according to the type of set which is indicated. For example, the sentence *John has five more marbles than Sarah* is classified as a more_than set. When a question is encountered, such as *How many marbles does John have?*, DCOMPS generates hypotheses about the operations required to solve the problem based on the sets which have been identified.

Like the two-term series problems, a theory of the particular problem solving process already exists (Kintsch and Greeno, 1985). Our simulation is an attempt to implement the theory within the DCOMPS architecture, and to compare the times required by the system to solve various problems with the times required by humans.

The time to respond to various types of Stroop conditions compares favorably with human data in three ways. First, color words presented in contrasting colors are not identified as quickly as color words presented in the same color. Second, it takes DCOMPS longer to identify the color of the word than the meaning of the word. Third, DCOMPS displays between-trial effects. It takes the system longer to identify a feature which was inhibited on the previous trial. For example, if the task is to identify the color of the word and the first word presented is the word *blue* printed in the color *red*, it would take longer to identify the color of the next word if its color is *blue*, since that feature was inhibited on the previous trial.

We are currently working on simulating two additional types of tasks using DCOMPS: factual and counterfactual reasoning, and musical chord priming. Future plans for DCOMPS include: 1) pursuing several approaches to relate DCOMPS' performance with human data, 2) studying individual differences in problem solving by manipulating the parameters of the model, and 3) simulating several additional cognitive tasks.

DCOMPS offers a flexible compromise between production systems and connectionist models. Although some researchers have suggested a way in which a static connectionist model might simulate sequential processess (e.g, Rumelhart, Smolensky, McClelland, & Hinton, 1986) this method is very likely to be computationally expensive and, more importantly, to be so complex as to be intractable in simulating problem solving tasks. DCOMPS explores the possibility that a simple system such as the one proposed can

solve problems which involve both sequences of actions and parallel processing.

References

Anderson, J.R.(1983). *The Architecture of Cognition*. Cambridge, MA: Harvard University Press.

Clark, H.H. (1969). Linguistic processes in deductive reasoning. *Psychological Review*, Vol. 76, No. 4, 387-404.

Hunt, E. and Lansman, M. (1986). Unified model of attention and problem solving. *Psychological Review*, 93(4), 445-461.

Kintsch, W. (1974). *The Representation of Meaning in Memory*. Hillsdale, N.J.: Erlbaum.

Kintsch, W. and Greeno, J.G. (1985). Understanding and solving word arithmetic problems. *Psychological Review*, Vol. 92, No. 1, 109-129.

Kintsch, W. and van Dijk, T.A. (1978). Toward a model of text comprehension and production. *Psychological Review*, 85, 363-394.

Neely, J.H. (1977). Semantic priming and retrieval from lexical memory: Roles of inhibitionless spreading activation and limited-capacity attention. *Journal of Experimental Psychology: General*, Vol. 106, No. 3, 226-254.

Richardson and Hunt, E. (1986). Problem solving under time pressure. University of Washington Technical Report.

Rumelhart, D.E., Smolensky, P., McClelland, J.L., and Hinton, G.E. (1986). Schemata and sequential thought processes in PDP models. In J.L.McClelland and D.E.Rumelhart, Eds, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 2: Psychological and Biological Models*. Cambridge: The MIT Press, 7-58.