DIVISION OF COMPUTER SCIENCE

The Representation of Rules in Connectionist Models

Michael St Aubyn

Technical Report No.227

June 1995

The Representation of Rules in Connectionist Models

MICHAEL ST AUBYN

Abstract: Three models of connectionist rule processing are presented and discussed: Shastri and Ajjanagadde's SHRUTI system, which uses connectionist mechanisms to overcome the inefficiency of classical rule-based models; Sun's CONSYDERR model, which combines rule-processing and similarity-matching in a two-level architecture; and Sumida and Dyer's Parallel Distributed Semantic (PDS) Network, which adds generalisation and inferencing capabilities to a semantic network. The Sun and Sumida and Dyer models are illustrated with implementations and example runs. Aspects of rule-based reasoning not addressed by these models (rule learning, the encoding of rules without hard-wired structure, and holistic rule application) are discussed in the context of a proposed new model.

1. Introduction

Much of human cognitive performance has the appearance of being rule-driven. At the most obvious level, we are capable of internalising and applying sets of instructions (the directions that someone gives us in the street, for example); our use of language seems to be governed by rules of grammar; and, more generally, the existence of systematic relations in human thought has been taken as evidence for a grammar of mental representations (Fodor 1975, Fodor and Pylyshyn 1988). Such considerations settle comfortably into a sententialist account of cognition, one in which complex structured representations are modified by rules in an essentially serial symbol-processing mechanism. This is the theory of cognition that underlies most of classical (here meaning pre-connectionist and pre-artificial life) AI.

The problems of classical cognitivism are well known. They derive largely from the inflexibility of rule-based systems and their consequent impracticality outside artificial domains; and from the difficulties of reconciling a high-level cognitive model with the base-level neural mechanisms in which it is supposedly grounded. A classical model of natural language processing, for example, may apply a sequential algorithm to a string of words to produce a representation of the syntactic structure of the sentence, while in the process creating and discarding a number of temporary structures representing instantiations of variables, aborted interpretations of phrases, etc. The

techniques employed here - serial processing, nested representations, dynamic memory allocation and garbage collection - are natural and familiar features of most computer languages (a fact which makes it easy to overlook how un-brainlike they are); yet if a program which employs them is claimed to be a model of cognition, as many classical AI systems are, it is reasonable to ask for an account of these mechanisms along with the rest of the model.

In one view, this is the role of connectionism: to provide a plausible explanation of the basic subskills of cognition, which can then be employed in a high-level (and essentially classical) model. If some of the 'nice' features of connectionism (generalisation, pattern completion, graceful degradation, etc.) percolate upwards in the process, so much the better. Thus, the processing of sequential information has been modelled in recurrent networks (Elman 1988); complex nested representations in recursive auto-associative memory (RAAM) (Pollack 1988); and dynamic memory allocation in neural net models using node recruitment techniques (e.g., Diederich 1988). This might be called the *bridging* approach to connectionism, since it uses connectionism to span the divide between top-level cognition and base-level mechanism.

In another view, connectionist models such as those just mentioned represent a theory-level challenge to classical cognitivism. By showing how simple tasks may be accomplished without classical structure, they raise the possibility that large areas of cognition might be performed in this way. Chalmers (1990), for example, shows how the transformation of a sentence from the active to the passive form (e.g., *John loves Mary -> Mary is loved by John*) may be performed by a connectionist model which operates *holistically* upon RAAM-encoded distributed representations of the sentences, i.e., without the usual compositional analysis and reconstruction. Such models encourage an *eliminativist* view of connectionism, one which sees connectionism as an alternative theory at the *cognitive* level, rather than simply a new mechanism for implementing the old, sententialist cognitive structures.

The three models discussed in this paper hardly support so radical a view. To a large extent each is an *implementation* of a classical rule-based system, though one which raids the toolbox of connectionism for mechanisms which add interesting and desirable features to the reasoning process. Shastri and Ajjanagadde's model (Section 2), for example, makes use of the high level of parallelism in a neural network and the efficiency of activation propagation to overcome some of the practical limitations of a knowledge-based system; Sun (Section 3) uses feature-based representations to add similarity-matching to a reasoning system; and Sumida and Dyer (Section 4) increase the functionality of a semantic network by exploiting the ability of connectionist models to classify and generalise. To some degree, each of the three systems is claimed to be a

cognitively plausible model of certain aspects of human reasoning. Such claims are questioned and discussed throughout this paper.

2. A Connectionist Model of High-Level Inferencing (Shastri and Ajjanagadde)

Shastri and Ajjanagadde's SHRUTI system (1991) employs mechanisms borrowed from connectionism to directly implement the short episodes of rapid, unconscious inference which, the authors argue, underlie a wide variety of intelligent behaviour. In modelling this kind of reasoning (which the authors call *reflexive*, as distinct from *reflective*, pencil-and-paper reasoning) three kinds of knowledge are represented and used: long-term (hard-wired) facts, short-term facts, and long-term rules. No account is given of how long-term facts and rules are acquired, nor is it possible to modify long-term knowledge without cutting or adding connections or adding or deleting nodes (which the system does not allow). In terms of its behaviour, the model is, in fact, a fairly simple (though potentially very efficient) backward-chaining question-answering system in the style of Prolog. Where it differs significantly from other backward-reasoning systems is in its method of implementation. By using a mechanism consisting only of computationally-simple autonomous nodes propagating activations within a network, Shastri and Ajjanagadde attempt to produce a model of reflexive reasoning that is neurally and psychologically more plausible than a 'classical' system with a central controller running a serial program.

2.1. The Dynamic Binding Problem

The challenge of implementing an essentially classical process in a non-classical mechanism raises some interesting technical issues, foremost among which is the *dynamic binding problem*. This is the problem of assigning temporary values to predicate arguments, as when the general relationship *X owns Y* is instantiated to the short-term fact *Mary owns Book1*; using a rule (e.g., *If X owns Y then X can sell Y*) may require propagating the assigned values from the variables in the antecedent to those in the consequent (to infer that *Mary can sell Book1*). In addition, *multiple instantiations* of the same predicate (e.g., *Mary owns Book1* and *John owns Ball3*) should be allowed to coexist without crosstalk; and categorisation restrictions should be available to prevent nonsensical bindings (e.g., *Book1 owns Mary*).

In a classical reasoning system such as Prolog, dynamic binding is performed by means of a look-up table which holds the current value of each variable: the program simply retrieves the appropriate values from the table whenever they are needed (this

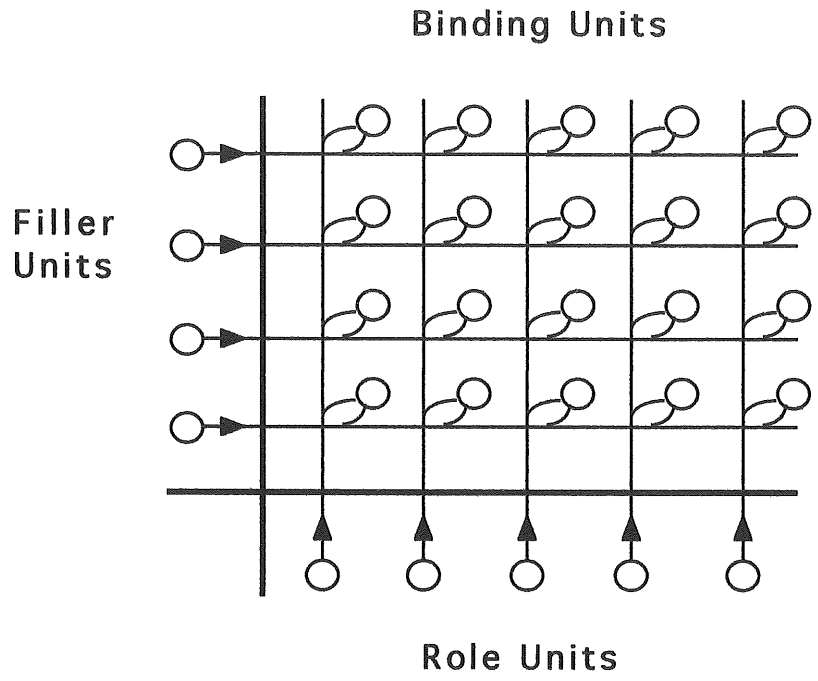

Figure 1. A network to perform tensor product binding (Smolensky 1990).

may be a recursive process if variables are allowed to take other variables as their values). Multiple instantiations are usually handled by renaming. Prolog, for example, creates a copy of a clause each time that clause is invoked, renaming all the variables at the same time to avoid conflict with other copies of the same clause.

Some connectionist models of dynamic binding use methods similar to that of the look-up table. For example, Smolensky's (1990) *tensor product* system is implemented using a grid of vertical and horizontal links: each vertical link is connected to a *role unit* and each horizontal link to a *filler unit*, with a *binding unit* at each point of intersection (Figure 1). A binding is stored by activating a pair of role and filler units and propagating the activations to the binding unit at the junction. To retrieve a binding, a role unit is activated; this activity, combined with that in the binding unit, sets up activity in the corresponding filler unit. The model can handle multiple instantiations (by multiplying the number of links and points of intersection) and variables with variable values (by recursively feeding filler unit activations back into the grid as role unit activations). The obvious limitation of this method is the sheer quantity of links and units that are required: $n*m$ binding units in a model with n variables and m values. Since each potential binding requires a dedicated unit, yet only a small proportion may ever actually be used, there is huge redundancy in a model of this sort. A less wasteful approach might involve the dynamic creation of pathways and units in the grid as they

become necessary. However, the extent to which such a system could be regarded as a cognitive model is questionable: dynamic structural changes are difficult to map onto a physical model of the brain.

For this reason, connectionist methods of dynamic binding typically involve some form of activation encoding within a static structure. An example is Lange and Dyer's (1989) system of *signatures*. A signature is an activation magnitude which uniquely identifies a concept. Thus to represent the dynamic binding of a variable to a value, a node representing the variable is activated with the value's signature. Passing a value from one variable to another in an episode of reasoning is then equivalent to propagating an activation along the link connecting two nodes.

2.2. Temporal Synchrony

Shastri and Ajjanagadde's solution to the dynamic binding problem is also a form of activation encoding, though one which exploits the waveform characteristics of an activation (in particular, its phase position) rather than its magnitude solely. A predicate argument is bound to a value if the nodes representing the argument and value are firing synchronously. They call this approach *temporal synchrony*.

As an example, a representation of the binary predicate *owns* contains two argument nodes labelled *owner* and *o-obj*. The entities *Mary* and *Book1* are represented by two additional nodes, each firing with a unique rythmic pattern of activation. The dynamic fact *owns(Mary, Book1)* may then be represented by synchronising the activation of the *owner* node with the *Mary* node, and the *o-obj* node with the *Book1* node.

A rule is represented by connecting the argument nodes of the antecedent to those of the consequent: the rule *owns(x,y) -> can-sell(x,y)* is illustrated in Figure 2. The directed links and the activation functions of the nodes ensure that the activation patterns of the *owner* and *o-obj* nodes are copied into the *p-seller* ('potential seller') and *cs-obj* nodes respectively. This means that the instantiation of the fact *owns(Mary, Book1)* will result in the simultaneous representation of the fact *can-sell(Mary, Book1)*. Longer chains of inference are represented by linking up more predicates in the same way.

There are two important consequences of this form of rule implementation: (1) a large number of rules can fire in parallel, and (2) the time taken to process a chain of inference is proportional to the length of the chain and independent of the total number of rules. This is in contrast to a serial rule-processing system based on linear search and backtracking, whose performance in general decreases as the magnitude and complexity of the rule base is increased. (Of course, these benefits of Shastri and Ajjanagadde's

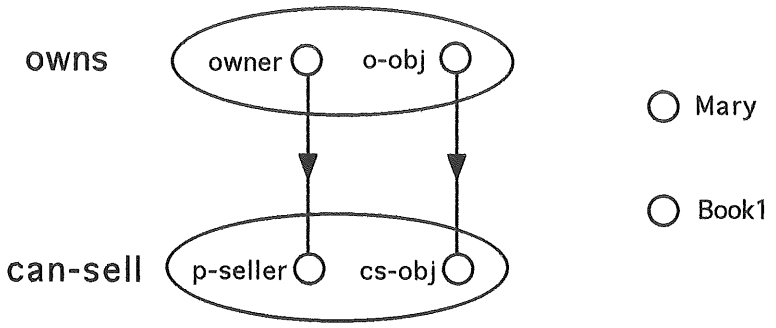

Figure 2. A connectionist representation of the rule *owns(x,y) -> can-sell(x,y).*

model only take effect when it is physically implemented, rather than merely simulated on a serial computer.)

A long-term fact (a static binding of a predicate) is encoded in the model as a *temporal pattern matcher*. A cluster of hardwired links and logic gates connects the predicate to the relevant entity nodes, and transmits a signal to a special node in the predicate (the *collector* node) if a dynamic binding of the predicate arguments corresponds to the static fact. This allows the network to behave as a questioning-answering system. For example, the question *Does John own Car3?* is presented by synchronising the *owner* node with *John* and the *o-obj* node with *Car3*; if *owns(John, Car3)* is encoded as a static binding, the collector node of the *owns* predicate becomes active, signifying a *yes* response. Partially instantiated facts may also be represented and compared: e.g., the dynamic fact *give(x, John, Book1)* (*giver* node left inactive) will match the static fact *give(Mary, John, y)* (*g-obj* node not connected to any entity node), and the collector node of the *give* predicate will fire.

2.3. A Backward-Reasoning System

In Shastri and Ajjanagadde's SHRUTI system, the mechanisms of temporal sychrony, activation propagation and temporal pattern matching are combined in a backward-reasoning system for question answering. The system encodes rules of the form:

$$\forall x_1,\ldots,x_m\{P_1(\ldots)\wedge P_2(\ldots)\ldots\wedge P_n(\ldots)\supset\exists z_1,\ldots z_j Q(\ldots)\}$$

where the arguments of each P_i are elements of $\{x_1,x_2,...,x_m\}$ and each argument of Q is an element of $\{x_1,x_2,...,x_m\}$ or an element of $\{z_1,z_2,...,z_j\}$ or a constant. (The system requires that a variable occurring in multiple argument positions in the antecedent also appear in the consequent.)

Facts are atomic formulae of the form $P(t_1,t_2,...,t_k)$, where each t_i is a constant or a distinct existentially quantified variable. Questions take the same logical form as facts but are encoded dynamically, as described above. A question containing a variable may be interpreted as a yes-no question or a wh-query: e.g., *owns(x,Book1)* could mean *Does someone own Book1?* or *Who owns Book1?*

Since the system is a *backward* reasoner, the links between predicates are reversed (i.e., they lead from the consequent to the antecedent). A question is posed by activating a set of predicate nodes to instantiate the consequent of a rule or rules. The activations are then propagated to the antecedents, which may in turn be the consequents of other rules. The backward propagation continues until a predicate matches its dynamic instantiation with its static binding. This causes the collector node to fire, which in turn activates the collector node of the previous predicate in the chain, and so on back to the predicate where the question was first instantiated, indicating a *yes* response.

As a simple example, suppose the system encodes the rule *owns(x,y) -> can-sell(x,y)* and the static fact *owns(Mary,Book1)*. The question *Can Mary sell Book1?* is asked by activating the argument nodes of *can-sell* in synchrony with *Mary* and *Book1*. The activations are then passed to the argument nodes of the *owns* predicate, dynamically instantiating the question *owns(Mary,Book1)?* This matches the static fact stored at this predicate, so the *owns* collector node fires. Finally, the activation of the collector node is propagated back to the collector of *can-sell*, signalling a *yes* reply to the initial question.

To answer a wh-question with the value or values of the unbound arguments (i.e., to reply *Mary* to the question *Who can sell Book1?*) requires a more elaborate encoding of static facts. For each long-term fact, extra nodes and connections are added which ensure that when a successful match is performed, the value of any variables bound in the process are temporarily stored. An answer extraction phase reads these values when a *yes* response is returned after a wh-question.

A problem for the system as described so far is that it cannot handle multiple dynamic instantiations of the same predicate: e.g., it cannot represent the dynamic fact *loves(Mary,John)* at the same time as the dynamic fact *loves(John,Mary)* (since each argument node encodes just one filler). Shastri and Ajjanagadde's proposed solution is to augment each predicate with *banks* of nodes, along with a complex switching mechanism that channels instantiations into the appropriate bank. So, for example, bank 1 of the *loves* predicate could hold the bindings {*lover* = *Mary*, *loved-one* = *John*}

while bank 2 holds {*lover* = *John, loved-one* = *Mary*}. The reasoning efficiency of the system is reduced in proportion to the number of banks per predicate. Details of this mechanism can be found in (Mani and Shastri 1992).

2.4. Discussion

The mechanism of temporal sychrony does not appear to offer any distinct *computational* advantage over alternative methods of dynamic node marking that exploit, for example, the magnitude of an activation (as in Lange and Dyer's model) or sets of nodes encoding binary values. Indeed, the technique presents significant implementational difficulties (involving the propagation and enforcement of synchrony) that do not arise in other, functionally-equivalent methods. As a computational device, temporal synchrony may prove to be most powerful when used as a supplement to other encoding techniques, to provide an additional dimension to the information content of an activated node.

Shastri and Ajjanagadde's principal justification of their approach is the 'biological plausibility' of temporal synchrony. They cite research on the cat visual cortex which suggests that the binding of visual features is realised by synchronous activity (Eckhorn et al. 1990). However, the plausibility of their model diminishes as they elaborate it with mechanisms that have no clear biological precedent: e.g., the *temporal and-nodes* needed to perform dynamic pattern matching, and the switched banks of nodes that hold multiple instantiations. Furthermore, the existence of certain mechanisms at the cellular level (the level of Eckhorn's research) is, arguably, poor evidence of their significance at the cognitive level (the level which Shastri and Ajjanagadde attempt to model). To implement high-level processes directly in a base-level mechanism is to gloss over the many intermediate levels (the *virtual machines* of the brain) by which simple mechanical processes (such as, perhaps, the propagation of sychrony) are built up, layer upon layer, into complex cognitive ones.

As a model of human psychological performance, the reasoning system described above has two advantages over classical models: the parallel firing of rules, which enables multiple inferences to be performed simultaneously; and the fact that the efficiency of any given episode of inference is unimpaired by the size of the rule-base. As a direct implementation of a rule-based system, however, the model suffers from some of the same deficiences as a classical system. In particular, it exhibits brittleness - a failure to cope with situations for which explicit rules have not been provided (if unable to answer a question precisely, Shastri and Ajjanagadde's model simply returns an unhelpful, Prolog-like *no*). Ron Sun's CONSYDERR model, described in the next section, directly addresses this issue.

A second problem is that the phase encoding technique is unlikely to exhibit the graceful degradation that we would expect of a cognitive model: a slight synchronisation error could result in the representation of an entirely different concept from the one intended. In Section 4, I shall discuss a model which, by using activation encodings with semantic content, goes some way towards solving this problem.

3. Similarity-Based Reasoning in a Two-Level Model (Sun)

One reason for implementing rule-based systems as neural networks is to soften the rigid formalism of such systems by introducing some of the 'nice' features of connectionism (pattern completion, graceful degradation, generalisation, etc.). A model which attempts to combine the power of rule-following with the more psychologically plausible features of connectionism is Ron Sun's CONSYDERR system (Sun 1992a, 1993, etc.). The network is designed to model a variety of commonsense reasoning patterns (including property inheritance) by treating them as instances of rule-following, similarity-matching or a combination of both. Here are two examples of commonsense reasoning (from Collins and Michalski 1990) that Sun models in this way:

1. Q: Is the Chaco the cattle country?
   R: It is like Western Texas, so in some sense I guess it's cattle country.

2. Q: Are there roses in England?
   R: There are a lot of flowers in England. So I guess there are roses.

The first example illustrates the use of similarity-matching to overcome a lack of explicit knowledge: *Western Texas -> cattle country, Chaco is-like Western Texas,* therefore *Chaco -> cattle country*. The second example illustrates a form of top-down property inheritance: *England has flowers, rose is-a flower*, therefore *England has roses*. Neither of these conclusions is certain (as indicated by the respondent's 'I guess'); each depends (or can be made to depend) on a similarity judgment.

Similarity is defined in Sun's model in terms of overlapping feature descriptions. Two concepts are similar if they share a sufficiently large proportion of their features, the degree of similarity (and the degree of certainty of any judgments based upon that similarity) depending on the extent of the overlap of the feature sets.

Given this definition, category membership may be regarded as a special case of similarity. For example, we can represent the fact that *rose is-a flower* by making the feature set of *flower* a subset of the feature set of *rose*. We can do this because *flower*, being the more general term, requires a less precise, less detailed description; in

defining a rose, we begin by defining a flower and then add extra features (e.g., *has-thorns*) that distinguish it from other flowers. It is possible in this way to construct an ISA-hierarchy using sets and subsets of features. The advantages of this approach become apparent when the network is used to implement property inheritance.

3.1. The CONSYDERR Architecture

The CONSYDERR network consists of two interconnected levels: a localist level (CL) for explicit, rule-based reasoning, in which nodes represent concepts and links represent inference rules; and a distributed level (CD) in which nodes represent features of the concepts in CL. An example (adapted from Sun 1992a) is illustrated in Figure 3.

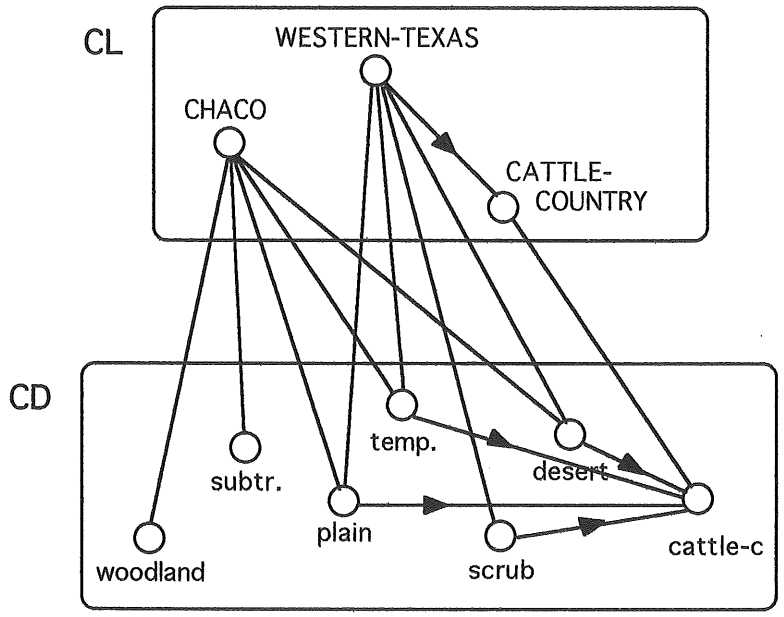

Figure 3. The CONSYDERR network for the Chaco protocol (from Sun 1992a).

Here, the localist concept *CHACO* is represented in the CD level by a set of five features (*woodland, subtropical, plain, temperate,* and *desert*); *WESTERN-TEXAS* is represented by four features (three shared by *CHACO*); and *CATTLE-COUNTRY* by one feature. Each CL node is connected to each of its CD feature nodes by two weighted links (here represented by a single line), one passing activations top-down from CL to CD, the other propagating bottom-up. The rule *WESTERN-TEXAS* ->

CATTLE-COUNTRY is represented by the directed link between the two nodes in CL. It is also represented distributedly in CD by links from the feature nodes of *WESTERN-TEXAS* to the feature node of *CATTLE-COUNTRY*.

Having defined the network and established the weights of the connections (as explained below), we activate and clamp the CL node that we are interested in, and propagate its activation through the network in three phases. Phase 1 enables the top-down links, activating (and clamping) the CD feature nodes corresponding to the active CL concept. Phase 2 enables the intra-level (rule) links, allowing the CL and CD activations to propagate and settle. Finally, phase 3 enables the bottom-up connections. The results of the enquiry are read off as activations in CL.

In the example network of Figure 3, it is the *CHACO* concept that interests us, so we begin by clamping this node in CL. The top-down phase then activates and clamps the five nodes in CD that correspond to the feature description of *CHACO* (incidentally activating three quarters of the feature description of *WESTERN-TEXAS* at the same time). Nothing happens in the CL settling phase, since there are no links from the activated *CHACO* node to either of the two other CL nodes. In the CD settling phase, however, the activations of three of the feature nodes (*plain*, *temperate* and *desert*) are propagated to the *cattle-country* CD node along links that distributedly represent the rule *WESTERN-TEXAS*. Finally, in the bottom-up phase, the CL nodes *WESTERN-TEXAS* and *CATTLE-COUNTRY* are activated by their feature nodes. We read the answer to the original question (*Is the Chaco the cattle country?*) by examining the *CATTLE-COUNTRY* node in CL: the activation of this node indicates a *yes* reply.

Two points should be noted here. First, the similarity-matching and rule-following are each done in parallel, and in time independent of the number of concepts, features and rules involved - an important property in a model of spontaneous reasoning processes. Secondly, a node activation can take a range of values, allowing a measure of confidence to be attached to each conclusion: e.g., a result derived from a marginal similarity-match is indicated by a low level of activity in the corresponding node.

In the following sections, I describe an implementation of Sun's model, and discuss the issues arising from it.

3.2. Implementation Details

A certain amount of detail must be added to this basic description before we have a working model. Most importantly, we must define the activation functions of the CL and CD nodes, and the formulae for calculating the link weights. The functions and formulae given here are slightly simplified versions of those in (Sun 1993).

In the top-down phase, the activation of each CD node i is found by calculating ($WEIGHT_{Ai}$ * $ACTIVATION_A$) for each CL node A that is linked to i by a top-down connection, and then choosing the maximum weighted input as the activation value.

In the settling phase, the activation of a CL or CD node is the sum of its weighted inputs from activated nodes in the same level.

In the bottom-up phase, the activation of a CL node is EITHER its activation from the CL settling phase OR the sum of its weighted inputs from the CD nodes to which it is connected. The larger of these two values is chosen as the activation value of the node (since we do not want an uncertain result from one level to diminish a more certain result from the other).

The connection weights are defined as follows. The rule weights in CL are specified by the user, and indicate the certainty of a rule. In this implementation, rule weights are floating point numbers in the range 0 to 1, so if it is 90% certain that X implies Y, this could be indicated by a weight of 0.9 on the directed link from X to Y.

The CL rules are represented distributedly in CD by connecting each feature node of the antecedent to each feature node of the consequent. The weight on each connection is the same, and is found by dividing the rule weight in CL by the number of features in the antecedent.

The top-down weights from a concept X to its feature nodes are all set at 1, and the bottom-up weights from the feature nodes to X are each (1 / the number of features in X). This arrangement ensures that in passing the activation of X to the distributed layer and back again, we end up with the same activation value at X that we started with.

The configuration of the rule weights and the activation functions involves a number of simplifying assumptions about the knowledge domain. In particular, it is implied that each feature contributes equally in the representation of a concept and in the rules in which the concept is involved. The problems arising from this assumption will be discussed later.

In this implementation, the top-down, bottom-up, and CD intra-level weights are calculated automatically in the initial set-up procedure of the program. The user supplies (within the program code) a list of the concepts to be represented in CL, together with their feature definitions. A list of CL rules and rule strengths must also be provided, along with the activation level of the initial enquiry node. The program then constructs the network and runs the three phases, as described above.

3.3. Some Example Runs

The first example implements the network in Figure 3. The output of the program is given below:

```
Nodes in CL, and their features:
   WESTERN-TEXAS = {temperate, plain, desert, scrub}
   CHACO = {subtropical, woodland, plain, temperate, desert}
   CATTLE-COUNTRY = {cattle-country}

Rules in CL:
   WESTERN-TEXAS -> CATTLE-COUNTRY (Rule strength = 1.000)

Activated nodes in CL, initially:
   CHACO (1.000)
```

The program begins by displaying the information supplied by the user. It now constructs the two-level network and displays the connection strengths within CD:

```
Links in CD:
   temperate -> cattle-country (Link weight = 0.250)
   plain -> cattle-country (Link weight = 0.250)
   desert -> cattle-country (Link weight = 0.250)
   scrub -> cattle-country (Link weight = 0.250)
```

Notice that each of the four link weights in CD is a quarter of the value of the rule weight in CL: the implication, that each feature contributes equally to the high-level properties of a concept, is a convenient, though unrealistic, assumption (Western Texas may be cattle-country *in spite of* being desert). The top-down phase is now performed, activating the distributed representation of *CHACO*:

```
Activated nodes in CD, after top-down phase:
   temperate (1.000)
   plain (1.000)
   desert (1.000)
   subtropical (1.000)
   woodland (1.000)
```

The settling phase now enables the internal connections in each level, and the activations are propagated according to the rule-based knowledge represented in the links:

Activated nodes in CL, after settling phase:

CHACO (1.000)

Activated nodes in CD, after settling phase:

temperate (1.000)
plain (1.000)
desert (1.000)
subtropical (1.000)
woodland (1.000)
cattle-country (0.750)

The settling phase has had no effect in CL (there are no explict rules involving *CHACO*), but in CD the *cattle-country* node has become partially active. The activation of 0.75 recognises that the use of the CD representation of the rule *WESTERN-TEXAS -> CATTLE-COUNTRY* is uncertain (since only three quarters of the feature description of *WESTERN-TEXAS* have been activated). Finally, the bottom-up phase converts the distributed representations into localist activations:

Activated nodes in CL, after bottom-up phase:

WESTERN-TEXAS (0.750)
CHACO (1.000)
CATTLE-COUNTRY (0.750)

Here the *cattle-country* activation has simply been propagated to the CL node, where its value may be interpreted as the certainty of the conclusion 'Chaco is cattle country'. The activation value of *WESTERN-TEXAS* is not exactly a measure of the similarity of *WESTERN-TEXAS* to *CHACO*, but rather an indication of the extent to which the former may be regarded as a superclass of the latter. (If the *CHACO* feature description had contained all four features of *WESTERN-TEXAS*, the activation of *WESTERN-TEXAS* (and *CATTLE-COUNTRY*) would have been 1.)

The second example run implements top-down property inheritance in an ISA hierarchy.

Nodes in CL, and their features:

BIRD = {flies, bipedal, has-feathers, lays-eggs, has-beak}
CANARY = BIRD + {is-yellow, sings}

TWEETY = CANARY + {lives-in-cage, belongs-to-john}

PENGUIN = {bipedal, has-feathers, lays-eggs, has-beak, likes-cold, eats-fish}

PONGO = PENGUIN + {lives-in-london-zoo}

SCARED-OF-CATS = {scared-of-cats}

CHASES-MICE = {chases-mice}

Rules in CL:

BIRD -> SCARED-OF-CATS (Rule strength = 1.000)

Activated nodes in CL, initially:

TWEETY (1.000)

The network constructed from this data represents the following facts: *TWEETY isa CANARY*, *CANARY isa BIRD*, *PONGO isa PENGUIN*, *PENGUIN isa BIRD*, and *BIRD has-property SCARED-OF-CATS*. Each of the *isa* facts is represented in terms of overlapping feature sets. Thus the *BIRD* feature set is a subset of the *CANARY* feature set. Notice that one member of the *BIRD* feature set (*flies*) is not included in the set of *PENGUIN* features, indicating that *BIRD* is not an exact superclass of *PENGUIN* and that *PENGUIN isa BIRD* does not have the same degree of certainty as the other *isa* facts. (This does not seem entirely unnatural if we interpret *BIRD* as meaning 'typical bird'.) The *has-property* fact is represented by a link between two CL nodes. In activating the *TWEETY* node, we effectively ask the question *What can we infer about Tweety?*

Activated nodes in CD, after top-down phase:

flies (1.000)

bipedal (1.000)

has-feathers (1.000)

lays-eggs (1.000)

has-beak (1.000)

is-yellow (1.000)

sings (1.000)

lives-in-cage (1.000)

belongs-to-john (1.000)

Activated nodes in CL, after settling phase:

TWEETY (1.000)

Activated nodes in CD, after settling phase:

flies (1.000)
bipedal (1.000)
has-feathers (1.000)
lays-eggs (1.000)
has-beak (1.000)
is-yellow (1.000)
sings (1.000)
lives-in-cage (1.000)
belongs-to-john (1.000)
scared-of-cats (1.000)

Activated nodes in CL, after bottom-up phase:
BIRD (1.000)
CANARY (1.000)
TWEETY (1.000)
PENGUIN (0.667)
PONGO (0.571)
SCARED-OF-CATS (1.000)

The firing of the *SCARED-OF-CATS* node with activation 1.0 indicates that Tweety, as a typical bird, is scared of cats. It is an important feature of this network that the retrieval of this fact was achieved without the kind of path-following activity that is typical with inheritance hierarchies. Class membership is decided simultaneously at all levels of the hierarchy as an automatic consequence of the feature set system of representation.

It is interesting to see what can be inferred about Pongo, as an atypical bird. We do this by reinitialising the network with *PONGO* as the activated node, and performing the three-phase propagation as before. The final result is as follows:

Activated nodes in CL, after bottom-up phase:
BIRD (0.800)
CANARY (0.571)
TWEETY (0.444)
PENGUIN (1.000)
PONGO (1.000)
SCARED-OF-CATS (0.800)

The atypicality of penguins means that the properties of a typical bird cannot be assigned to Pongo with complete certainty.

The final example run implements the second example of commonsense reasoning in the introduction to this section:

Q: Are there roses in England?
R: There are a lot of flowers in England. So I guess there are roses.

Nodes in CL, and their features:

ENGLAND = {england}

FLOWER = {has-stem, has-petals, grows-in-soil, needs-sunlight, needs-water, scented}

ROSE = FLOWER + {has-thorns, fragrant}

RED-ROSE = ROSE + {is-red}

WHITE-ROSE = ROSE + {is-white}

Rules in CL:

ENGLAND -> FLOWER (Rule strength = 1.000)

Activated nodes in CL, initially:

ENGLAND (1.000)

Activated nodes in CD, after top-down phase:

england (1.000)

Activated nodes in CL, after settling phase:

ENGLAND (1.000)

FLOWER (1.000)

Activated nodes in CD, after settling phase:

england (1.000)

has-stem (1.000)

has-petals (1.000)

grows-in-soil (1.000)

needs-sunlight (1.000)

needs-water (1.000)

scented (1.000)

Activated nodes in CL, after bottom-up phase:

ENGLAND (1.000)

FLOWER (1.000)

ROSE (0.750)

RED-ROSE (0.667)

WHITE-ROSE (0.667)

3.4. Discussion

Many of the problems with Sun's model are problems of microfeature representation in general. There is not just the considerable difficulty of constructing a feature set for each concept; we must also make decisions about the relevance of each feature for a given task. Sun's approach, though suitable for the highly specific examples illustrated here, cannot easily be generalised to handle a range of tasks in which the notion of relevance may change from one episode of reasoning to the next. As a simple example, the CONSYDERR representations of John Major and Tony Blair would each contain the whole of the *politician* feature set, which in turn would contain all the features of *human*, and so on. The very large number of shared features would encourage the model to assume that anything true of John Major is also true of Tony Blair; whereas in fact most questions involving these two would require the reasoner to focus on their political differences.

Sun hints at a solution to this problem when he talks of an 'attention-focusing module external to the system' (Sun 1991 p.441). He gives no details, but one can imagine a device that masks out parts of the CD layer (perhaps by sending a strong negative bias to selected CD nodes), forcing the model to focus its similarity judgments on a limited set of microfeatures - those concerned with political policy, for example. There remains, of course, the question of how the external module would itself determine relevance (also the problem of predefining the feature sets to cope adequately with all possible contexts in which they might be required).

An aspect of microfeature representation that is highlighted in the example runs is that some features are clearly more central to the definition of a concept, and contribute more to its high-level properties, than others. This idea is not captured in Sun's model, which automatically assigns equal importance to all features of a concept representation. A simple remedy might be to give each feature an importance rating or *centrality* (Sutcliffe 1992), represented by the weights of the interlevel links. But the issue of relevance again raises problems: features may need to assume different centralities in different contexts.

Even greater difficulties arise when we consider concepts that do not readily lend themselves to microfeature representation. What, for example, are the microfeatures of SCARED-OF-CATS, which we chose to represent as a high-level concept in the second example run? To say that there are no microfeatures because we

cannot name them seems unreasonable: we should not expect microfeatures, being subconceptual, to have names anyway. But if we cannot even name the features of a concept, how can we assign plausible centralities? These sorts of questions plague Sun's model, and indeed any model which relies heavily on feature representation.

One of the properties of the model that Sun draws attention to is its ability to merge concepts by simultaneously activating their feature descriptions. The example he gives (Sun 1991 p.440) is that by activating the description of *utility-vehicle* at the same time as *passenger-vehicle*, we end up with a description of *van*. However, this tendency of the model to merge everything it represents can also be regarded as a serious limitation. Suppose we wished to ask a question involving more than one concept, e.g., *Is Tweety scared of Sylvester?* Activating Tweety and Sylvester simultaneously and propagating the activations to CD would not produce two separate representations but one merged representation of the two entities together. Unless some form of activation encoding (temporal synchrony?) is employed to identify members of separate distributed representations, the model is restricted to reasoning about a single concept (or set of semantically related concepts) at a time.

A further serious restriction of the model as presented here is the propositional nature of the rules in CL. The system is capable of representing facts such as *All birds are scared of cats* (in the second example run), but the reasoning component falls far short of the expressiveness of first-order logic (or even the subset of FOL implemented in Shastri and Ajjanagadde's model). Sun's proposed solution (Sun 1992a, 1992b) involves replacing each CL node with an assembly of nodes representing a predicate (e.g., *IS-BIRD(X)*, *SCARED-OF(X,Y)*), each argument of the predicate being represented by a separate node within the assembly. To associate a predicate argument with an entity, the appropriate node is activated with a value that represents that entity (in this respect, Sun's approach resembles Lange and Dyer's (1989) method of variable binding by activation *signatures*). To perform an inference, as in Shastri and Ajjanagadde's model, the argument node activations are passed along links to the argument nodes of other predicate assemblies in CL. In CD, node assemblies are used to represent predicates and their arguments in distributed form; argument fillers are again represented by node activations, and take their values from the argument nodes of the predicate assemblies in CL during the top-down phase.

In other words, it is predicates such as *SCARED-OF(X,Y)* which are now being represented explicitly in CL and distributedly in CD (as a set of feature predicates: *f1(X,Y)*, *f2(X,Y)*, etc.). But this raises a number of new difficulties. First, the construction of microfeature sets seems even more problematic when applied to two- or more place predicates (Sun gives no concrete examples, and indeed it is hard to think of many relationship predicates which can be adequately deconstructed into microfeatures, as the model requires). Secondly, the communication of bindings between CL and CD

will lead to frequent conflicts in both the top-down and bottom-up phases. Since the feature sets in CD may overlap, there will be a conflict whenever a CD predicate is required to participate in the representation of two differently instantiated CL predicates (do the argument nodes in the feature predicate take bindings from the first CL predicate, from the second, or from a combination of both?). A similar problem arises in the bottom-up phase when the features of a single CL predicate have been instantiated with conflicting bindings in the settling phase. Sun discusses these problems (Sun 1992a p.312), but proposes no satisfactory solutions.

As a cognitive model, the two-level architecture corresponds, to some degree, to Shastri and Ajjanagadde's distinction between *reflective* and *reflexive* reasoning, the former being implemented in the CL settling phase, the latter in the CD settling phase. The top-down phase models the process by which we internalise concepts to prepare them for reflexive (unconscious) reasoning, while the bottom-up phase merges the results of the two reasoning processes in a form suitable for communication to others. One point at which the correspondence fails, however, is in the duplication of the CL rules in CD. Part of Shastri and Ajjanagadde's thesis is that the rules involved in reflexive reasoning are of a different nature from those used reflectively - not just the same rules represented differently, as in Sun's model. (In fact, the duplication of rules leads to some superfluous processing in the model: in each of the example runs, the removal of the CL settling phase made no difference to the result.) An interesting experiment might be to implement different sets of rules in the two levels, to capture more closely the distinction between the two forms of reasoning.

In the first-order version of Sun's model, as in Lange and Dyer's system, dynamic variable binding is performed by firing a node representing a variable with an activation signature representing an entity. The same principle is employed in the Shastri and Ajjanagadde model, though here the signature is encoded by *phase* rather than *magnitude*: two node activations must be in synchrony if they are to represent the same entity. In all three models, a signature is an arbitrary value chosen more or less at random - the only requirement is that it be distinct from other signatures simultaneously employed in the same model.

There are two difficulties here. First, a reasoning system may need to access a large number of separate items. If each is to be represented by a distinct signature, there must be a very high degree of precision in the activation encoding at each argument node. This is both biologically implausible (if the nodes are intended to correspond in some way to neurons) and technically difficult to enforce in a physical implementation of a network. The technical difficulties are particular severe in Shastri and Ajjanagadde's model, since the propagation of activations inevitably involves slight delays, leading to corruption of the phase signatures. In fact, Shastri and Ajjanagadde

estimate that their model can unambiguously represent only about ten distinct individuals at one time. Their proposed solution to this problem is a system of dynamic allocation of phase signatures from a small pool, to which each signature is returned when no longer needed. The recycled signature is then made available for reallocation. If the system is required to reason simultaneously about more items than it has signatures to label them with, it becomes 'forgetful' (in fact, Shastri and Ajjanagadde note the close correspondence of their figure of ten to the number 7 ± 2, the estimated measure of short-term memory capacity (Miller 1956)). The signature pool is an ingenious solution, but it is difficult to imagine how a mechanism of this sort could be realistically incorporated into the model.

A second problem arises when a signature is corrupted, perhaps by transmission delays (in the case of Shastri and Ajjanagadde's model) or by unavoidable noise interference. We would like the model to exhibit graceful degradation in such circumstances: i.e., to make plausible errors such as confusing one object for another that is very similar. But this is unlikely to happen in a system which uses arbitrary values as its signature encodings. One way around this problem is to use signatures which have semantic content - which are, in effect, reduced descriptions of the things they represent. This is the approach adopted in the model described in the next section. It will be shown also that this approach allows other desirable features of connectionism (generalisation and pattern completion) to be incorporated into the process of reasoning.

4. Parallel Distributed Semantic Networks (Sumida and Dyer)

A Parallel Distributed Semantic (PDS) Network (Sumida and Dyer 1989) combines two forms of knowledge representation: a semantic network and a parallel distributed processing (PDP) system. A semantic network is a localist representation of a hierarchically structured knowledge base in which concepts are represented by nodes and the relationships between concepts by labelled arcs connecting the nodes. An example is given in Figure 4.

Clearly this network is no more than a graphical representation of the following facts:

Bill is an instance of a *human*;
Mary is an instance of a *human*;
John is an instance of a *human*;
a *human* is the agent of a *hit-act*;
a *human* is the patient of a *hit-act*;
a *hit-act* is an instance of *competitive-activity*;

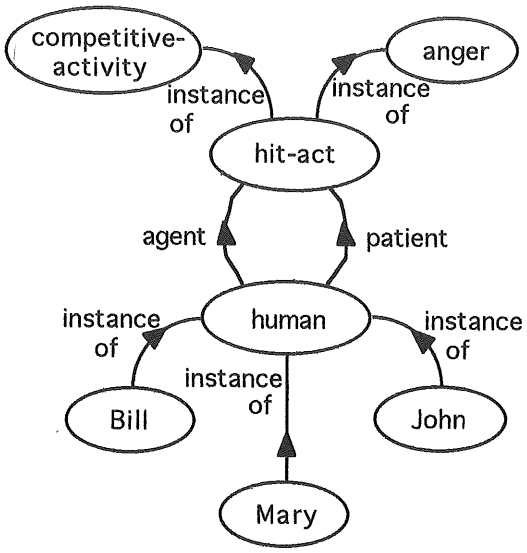

Figure 4. A semantic network.

a *hit-act* is an instance of *anger*.

It is possible to imagine how a simple reasoning system might be developed from this network of nodes and links. Suppose that the *human* node is activated in some way that represents the individual *Bill*, and then again in some way that represents *John*. The *hit-act* node, taking input from *human*, is then activated to represent *Bill hits John*, and the activation is propagated to the *competitive-activity* node or the *anger* node, representing one of the following inferences:

If Bill hits John then Bill and John are involved in competitive activity

or

If Bill hits John then Bill is angry with John.

There are two main problems here. First, it is not immediately obvious how the activation of a single node can unambiguously represent a composite statement such as *Bill hits John*. Secondly, a semantic network that represents all possible inferences is indeterminate as to which inferences are applicable in any given situation (e.g., it cannot determine whether a particular hit-act is an instance of competitive activity or anger).

In Sumida and Dyer's model, the first of these problems is handled by the use of *reduced descriptions*, the second by *propagation filters*.

4.1. Role Binding by Reduced Descriptions

In a PDS network, the binding of a concept to a particular instance is encoded by a pattern of activation across a set of units representing that concept. For example, the concept *human* might be represented by an ensemble of three units, each taking values in the range -1 to +1. A particular human is then represented by a unique activation pattern across this conceptual ensemble, e.g., [-1, 0, 0.5] for *Bill*, [0.5, -0.3, 1] for *John.*

The activation pattern for each instance is constructed from a feature description of that individual using an auto-associative encoder/decoder network (Rumelhart & McClelland 1986). This is a three-layer PDP net which is trained to recreate its input (a feature description of an instance) on its output layer, creating in the process a reduced feature description in its layer of hidden units. This reduced description is the activation pattern that is used to instantiate the conceptual ensemble. An example network for the *HUMAN* concept is shown in Figure 5.

This is a completely-connected, strictly-layered, 7-3-7 network which relates a 7-unit feature description in its input layer to a 3-unit reduced description in the hidden layer. A feature description is initialised using conversion information such as that in Table 1. So, for example, a boxer who is male, tall, strong and young would be represented by clamping the input layer with the values [1, -1, -1, -1, 0.5, 0.5, -0.5]. (Obviously, a more realistic implementation would need to include many more feature roles than this. The idea is to define each individual uniquely in terms of his or her feature description, so the larger the number of distinct individuals to be represented,

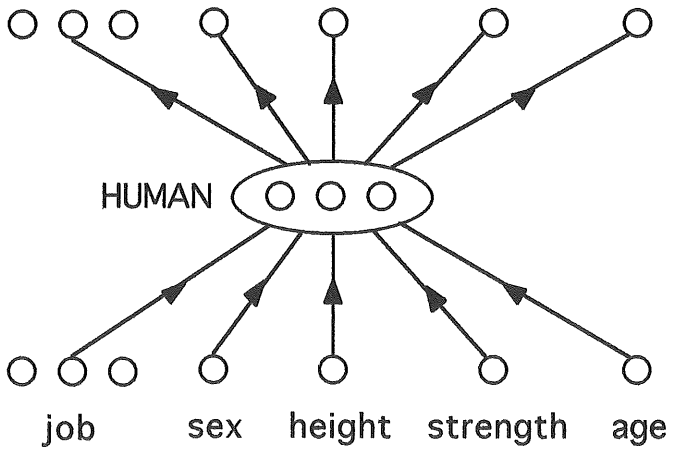

Figure 5. An encoder/decoder net for the *human* concept.

the larger the number of feature roles - or at least feature role values - required to define them.)

Role	Value	Encoding
job	boxer	1 -1 -1
	ballerina	-1 1 -1
	schoolchild	-1 -1 1
sex	male	-1
	female	1
height	very tall	1
	tall	0.5
	medium height	0
	short	-0.5
	very short	-1
strength	very strong	1
	strong	0.5
	medium strength	0
	weak	-0.5
	very weak	-1
age	very old	1
	elderly	0.5
	middle-aged	0
	young	-0.5
	very young	-1

Table 1. Encodings of features for the *human* concept.

The network is trained by backpropagation on a set of individuals, each defined in terms of these features, until it is capable of recreating each individual's feature description on the output layer (with an acceptable degree of error). The purpose of this training process is twofold: first, to produce a compact, portable representation of each member of the training set; and secondly, to enable generalisation *beyond* the training set. Generalisation arises because the net, in compacting the feature descriptions, is forced to classify the training data by discovering and exploiting the underlying regularities (Hinton 1986). (These regularities are exploited also by Sumida's propagation filter mechanism, to be described shortly.)

In a simple example run, the network of Figure 5 was trained on a sample of 21 individuals (mostly typical cases, e.g., strong, male, tall young boxers), then tested on a set of nine new individuals. After 120 epochs, the average squared error did not drop significantly below 0.1177 for the training set, and 0.1532 for the testing set; some features in both sets were distorted by the encoding/decoding process. The network's ability to encode and decode the examples in the testing set was, in part, a measure of the extent to which these new instances share the discovered regularities of the training set.

Allowing for these errors, we now have a (less than perfect) mechanism for constructing reduced descriptions of human beings, even those which the network has not seen before. Referring back to the semantic network of Figure 4, and remembering that each concept node in the network is now a *conceptual ensemble* of units, it is possible to see how role binding may be performed, at least in the case of the *HUMAN* node. This node is an ensemble of three units which are activated with the reduced description of the individual that we wish to represent.

Similarly, the *HIT-ACT* node is a conceptual ensemble that is activated with reduced descriptions of particular instances of hitting. As before, these reduced descriptions come from the hidden layer of a PDP network (Figure 6). The *agent* and *patient* unit groups in the input layer are clamped with values taken directly from the *HUMAN* conceptual ensemble. So to find the reduced description of *Mary hits John*, for example, we get reduced descriptions of *Mary* and *John* from the network in Figure 5, clamp the input layer of the *HIT-ACT* network with these values, and pass the activations forward to the hidden layer, where the encoding of *Mary hits John* now appears.

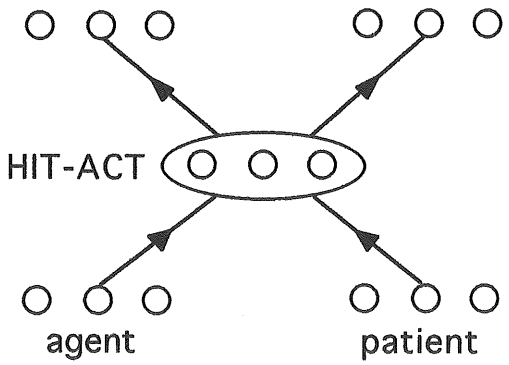

Figure 6. An encoder/decoder network for the *hit-act* concept.

As with the *HUMAN* network, the weights of the *HIT-ACT* network must be learned through training. In the example run, the loss of detail was even worse than in the *HUMAN* net - the mean squared error was 0.2697 for the training set (20 members),

and 0.2956 for the testing set (10 members) after 120 epochs. (These figures did not improve significantly with longer training periods, but there may be other ways of reducing the error rate - see the discussion section.)

The relationship between the two concepts so far defined in the PDS net is illustrated in Figure 7. The narrow lines from the *HUMAN* ensemble to the *agent* and *patient* groups indicate that the reduced descriptions are passed along these connections unaltered. The lines from *agent* and *patient* to *HIT-ACT* are weighted connections corresponding to the weighted links between the input layer and the hidden layer of the *HIT-ACT* network in figure 6.

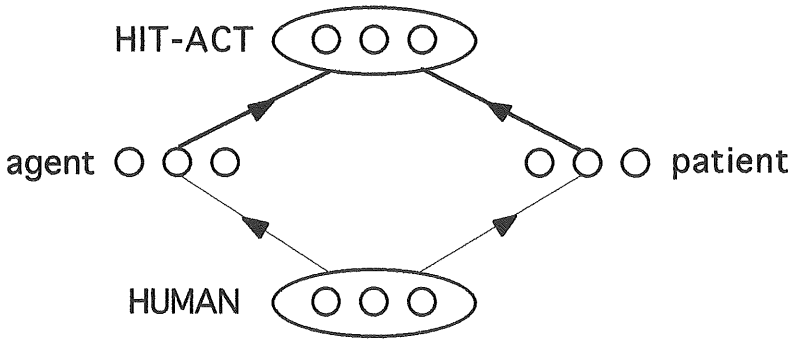

Figure 7. A PDS net fragment.

In theory, this PDS net can be expanded to any number of levels, with reduced descriptions being combined into higher-level reduced descriptions in much the same way as in *recursive auto-associate memory* (Pollack 1988). (The main difference is that RAAM uses just one auto-associative net, rather than several, to encode a nested structure.) We could, for example, set up the concept *KNOWS* to take inputs from *HUMAN* and *HIT-ACT* and train it on instances of *X knows that Y hit Z*; we could then add a *CAUSES* concept to represent instances of *'X knows that Y hit Z' causes X to hit Y* (figure 8). In the first paper on this model (Sumida and Dyer 1989), this is the means by which inference is implemented.

To see how this works, suppose that the *KNOWS* conceptual ensemble of Figure 8 is instantiated with a reduced description of *Bill knows that John hit Mary*. This set of activations is propagated unchanged to the *antecedent* unit group. Meanwhile, the *consequent* unit group is left uncommitted. When the *antecedent* and *consequent* activations are combined to form a reduced description of a *CAUSES* instance, the pattern at the *CAUSES* node completes itself (i.e., it settles to a reduced description of *Bill knows that John hit Mary CAUSES Bill hits John*). We can then extract the details

of the inferred consequent (*Bill hits John*) by decoding the reduced description (using the top halves of the appropriate encoder/decoder networks).

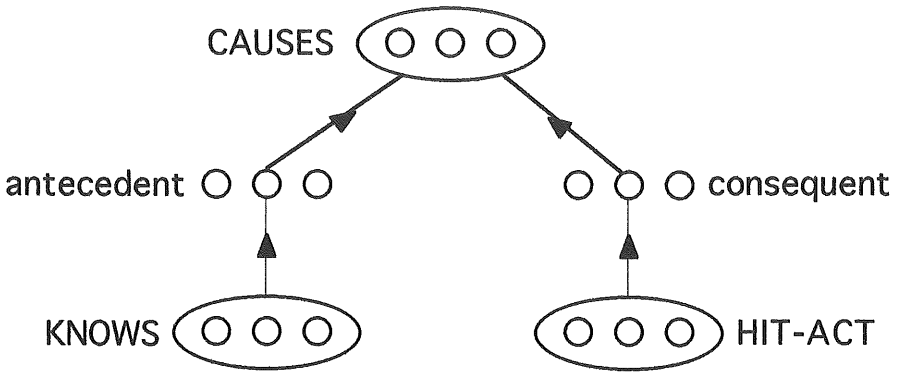

Figure 8. The CAUSES concept.

In practice, this mechanism of inference has severe limitations. First, a PDP net which is designed to perform pattern completion is unlikely to generalise well beyond the specific instances that it has been trained on (it will tend to assume that an unfamiliar instance is an incomplete version of a familiar one, and correct it accordingly). Secondly, as we have already seen, there is a loss of detail in the production of a reduced description. The more levels we add to the PDS net, the worse this problem becomes, and a point is inevitably reached when it is impossible to unambiguously represent complex instances in this way (though some of the errors generated by this loss of definition may nevertheless be cognitively interesting - see discussion).

4.2. Dynamic Inferencing using Propagation Filters

In (Sumida 1991), *propagation filters* are introduced as a new mechanism for performing inference in PDS networks. Propagation filters (inspired by the idea of *skeleton filters* in (Sejnowski 1981) and (Hinton 1981)) are groups of units which gate the connections between other unit groups (Figure 9). When a filter group is open, activation patterns pass freely along the connections; when closed, propagation is inhibited. The filters are opened and closed by a group of *selector units*: the pattern over the selector group opens up the appropriate filter by pushing its units above threshold. For example, in Figure 9, the selector group is set up to open filter group 1 when the pattern [1, 0] appears in selector units 1 and 2. This allows the activation pattern of *source1* to be passed across to *destination1*. The pattern in *source2* is not propagated because the units in filter group 2 remain below threshold.

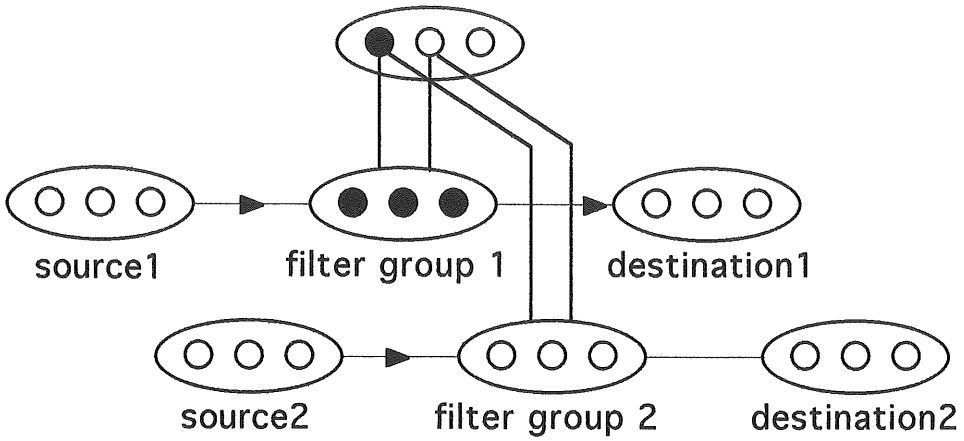

Figure 9. A propagation filter with group 1 open and group 2 closed.

This mechanism may be applied to the PDS net described in the previous section to make decisions about which of several possible inferences is appropriate in any given situation (e.g., whether a hit-act is an instance of competitive activity or anger). It was observed, when describing the PDP components of this model, that the process of squeezing a training set through the hidden layer of a PDP network forces the network to classify the input by discovering and exploiting regular features in the training data. Hinton (1986) showed how it is possible to inspect these regularities by analysing the activations of the hidden units under different inputs. Sumida's proposal is that it is possible to exploit these regularities to control propagation filters.

For example, an analysis of training set results for the *HIT-ACT* concept revealed that in every case where both the agent and patient were boxers, and in no other cases, the first value in the reduced description was strongly positive (> +0.5), and the third was very strongly negative (< -0.9). We may therefore use the *HIT-ACT* conceptual ensemble as a selector group to control the propagation of the agent and patient patterns from *HIT-ACT* to *COMPETITIVE-ACTIVITY* (in the case where a boxer hits a boxer) or from *HIT-ACT* to *ANGER* (in all other cases). This is illustrated in figure 10.

There are three main advantages that this method of inference has over the pattern completion technique. First, it provides an autonomous mechanism for selecting among possible inferences: the opening and closing of the propagation filters is an automatic response of the network when the *HIT-ACT* ensemble is instantiated. Secondly, there is no loss of information in performing the inference: the role bindings are simply passed across from the *HIT-ACT* role groups to those of *COMPETITIVE-ACTIVITY* through the open filters. Thirdly, this method seems to generalise well: an

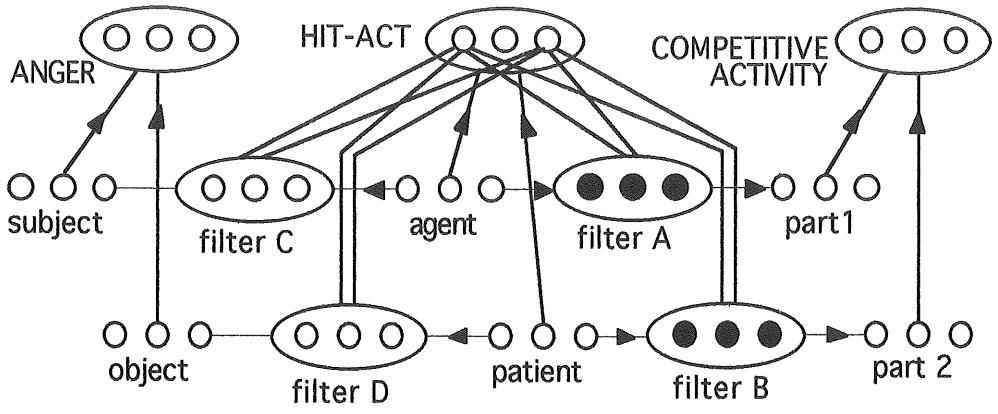

Figure 10. An example of the use of filters to control the propagation of bindings. A strong +ve value (> +0.5) in unit 1 of the *HIT-ACT* ensemble combined with a very strong -ve value (< -0.9) in unit 3 (indicating boxer-hits-boxer) opens up filters A and B, and generates the inference that this hit-act is an instance of competitive activity. Filters C and D remain below threshold.

analysis of the testing set results for the *HIT-ACT* concept revealed that even with examples not included in the training process, the convention of [unit 1 > 0.5, unit 3 < -0.9] was adhered to. This suggests that the PDS net with propagation filters will correctly classify instances of hitting as anger or competitive activity even if it has never seen the participants before.

4.3. Discussion

An obvious problem with PDS nets is the loss of resolution as the data is recursively compressed. Reducing a reduced description may make it impossible to recover the full details of a feature description at some later stage - at the end of an episode of reasoning, for example. This could be a serious problem if, as in this model, individuals are identified solely by their feature descriptions.

There are two responses to this criticism. One is to attempt to patch up the fault by modifying the mechanism - e.g., Sumida and Dyer (1989) suggest the addition of an extra layer of units to each conceptual ensemble, to increase its representational power; other simple improvements might involve expanding the training set, and using a fully binary representation for the feature descriptions (in the example runs, it was the one-unit continuous-valued features that were the most easily distorted).

The second response is to argue that the sorts of errors that arise from this loss of detail are at least cognitively plausible, and may indeed provide useful analogues for

certain aspects of cognitive performance. For example, if Frank the boxer has an 'identical' twin brother Barry, who is also a boxer, we might find it hard to remember which of the two we saw at a boxing match several weeks ago. In much the same way, the PDS network will have a tendency to confuse or blur together individuals and events with near-identical feature descriptions. Also, the fact that the errors worsen in proportion to the unfamiliarity of the data and the depth of encoding seems to correspond to our own difficulties in retaining novel, complex propositions (e.g., *John knew that because Mary wanted Bill to hit Tom ...*).

Similar 'plausible' errors will arise from noise interference, a potential problem in a hardware implementation of a PDS network. Because the binding mechanism employs representations that have semantic content, a slight distortion of the values in a conceptual ensemble will cause the binding to blur into another that is semantically related - e.g., *Barry hit John* instead of *Frank hit John* (an analogue of this in human experience might be the effect of trauma on long-term memory). This is in contrast to the effects that will be observed in models that use more or less arbitrary values to represent bindings. In Shastri and Ajjanagadde's model, for example, a slight distortion in the activation waveform of a role node could bring the node into synchrony with a totally unrelated value node elsewhere in the network.

One practical difficulty with the system of propagation filters as described here and in (Sumida 1991) is the problem of initialising the connections between the selector units and the filter groups. The system designer must perform this task himself by analysing the training results for each PDP network and determining which hidden units are being used to classify which aspects of the training data. A useful development of this model would therefore be the addition of mechanisms which automate or at least assist this process.

5. General Discussion

A common feature of the three models discussed in this paper is the use of hard-wired structure to represent rules. In each system, rules are implemented as links between nodes or groups of nodes, and the process of inference by the propagation of activations along the connections. This is a convenient and potentially powerful form of representation, permitting the direct and precise implementation of a variety of simple rule structures in a way that compares favourably with classical rule-based models. The parallelism of activation propagation allows multiple reasoning processes to proceed simultaneously (subject to the model's ability to handle multiple instantiations of a predicate) and eliminates the search and backtracking that characterises more traditional systems.

The three models elaborate on this mechanism in a variety of ways. For Shastri and Ajjanagadde, the main concern is to overcome the technical limitations of the basic model (e.g., the multiple instantiation problem) and to enhance it with the features that would normally be available in an advanced reasoning system (property inheritance, typed variables, the processing of *wh*-questions, etc). The resulting system is an impressive demonstration of the extent to which essentially classical processes may be implemented in a connectionist model without a central controller. It is questionable, however, whether this conflation of low-level mechanisms and high-level processes really adds anything at the performance level other than the efficiency of parallel processing: as a cognitive model, Shastri and Ajjanagadde's system may be no more plausible and no less brittle than a speeded-up expert system.

Sun addresses the issue of brittleness by augmenting the same structure-based rule representation with feature-based similarity matching. This produces a degree of flexibility, not in the rules themselves, which are permanently and rigidly embedded in the system, but in the decisions about which rules to apply. In Sun's model, a concept X may be partially activated as a side-effect of the activation of another concept Y; thus rules which apply to X may be employed in the response to a question about Y. The problem here is that similarity judgments, such as that between X and Y, are almost always context-dependent, a fact which Sun's model does not adequately acknowledge. In constructing a CONSYDERR network to solve some particular problem, we must effectively solve the problem in advance, in the decisions we make about which features to specify and how the feature sets overlap.

Sumida and Dyer also employ feature-based representations within a fixed, rule-based structure, though here the intention is to induce generalisation at the various levels of the structure. Thus it is hoped that the model will learn how to recognise instances of boxers hitting boxers, and to correctly categorise these examples as instances of competition rather than anger. Some of the practical limitations of this form of generalisation were explored in this report.

There are wider concerns which affect all three of these models, and which derive from their use of hard-wired structure to explicitly represent the rules that they encode. Explicit representation lays each of these models open to Fodor and Pylyshyn's (1988) charge that this sort of connectionist model does no more than implement classical processes, and is therefore uninteresting at the cognitive level. The use of novel mechanisms does not in itself constitute a novel approach or provide a new insight into cognitive processing if the symbol structures (in this case, the rules) are essentially the same as before.

A more practical concern is the inflexibility of structure-based representation, and the consequent difficulty of inducing change within it. It is clear, for example, that none of these models can acquire new rules or delete or modify existing ones, without

intervention at a much higher level than the basic operation of the system. Hadley (1993) argues that dynamic rule acquisition is an essential feature of human cognitive performance, and should be reflected in a model of rule processing. Our ability to acquire arithmetic skills, for example, seems dependent on our capacity for learning the rules by which basic operations are combined into more complex ones.

The ability to represent novel rules of variable structure and content from those defined in the initial model requires a different approach from those discussed here. Some hint may be taken from connectionist approaches to natural language processing, where a similar problem arises - that of representing novel sentences within a fixed architecture. One approach in particular seems relevant here: recursive auto-associative memory or RAAM (Pollack 1988). This is the means by which a complex nested structure may be encoded, level upon level, and using the same auto-associative network, into a compressed representation of fixed bandwith. The principle is similar to that employed in the Sumida and Dyer model, with one important difference: it is *structure* as well as *content* that is being encoded in the reduced descriptions.

Chalmers (1990) has shown that the reduced, RAAM-encoded description of a simple sentence contains sufficient syntactic information for active-passive transformations to be accurately performed by a pattern associator network. This network sees only the reduced descriptions of the sentences and thus operates *holistically* upon the sentences, i.e., without the usual extraction and reconstruction of the parse tree. Applying this principle to rule processing, a model might be constructed to perform holistic application of RAAM-encoded rules. The ability of the model to acquire and correctly apply rules of novel structure and content would be a measure of its ability to generalise. A possible architecture for such a model is given in Figure 11.

This model has two components: a rule encoding component by which rules of arbitrary size and structure are compressed (by RAAM or otherwise) into a fixed-bandwith reduced description; and a rule application component which feeds the rule and its datum into a (trained) backpropagation network to produce a resultant at the output layer. In this example, the rules are designed to perform operations upon number strings. The fact that there is no explicit structural representation in the reduced description of a rule means that the rule application component is not implementing explicit, 'classical' rule-processing but something uniquely connectionist. Notice, in particular, that there is a form of implicit variable binding in the way that the rule applicator processes terms such as *first* and *fifth* which is very different from the explicit mechanisms employed in the Shastri and Ajjanagadde and Sun (first-order) models.

It remains to be seen if such a model is capable of producing the degree of generalisation that would be required for the effective encoding of novel rules. As was observed with the Sumida and Dyer model, the iterative encoding of reduced descriptions leads to a progressive deterioration of the original data. Furthermore, the

rule application component may itself be severely limited in the complexity of the rules that it can handle. These issues are the subject of present investigation.

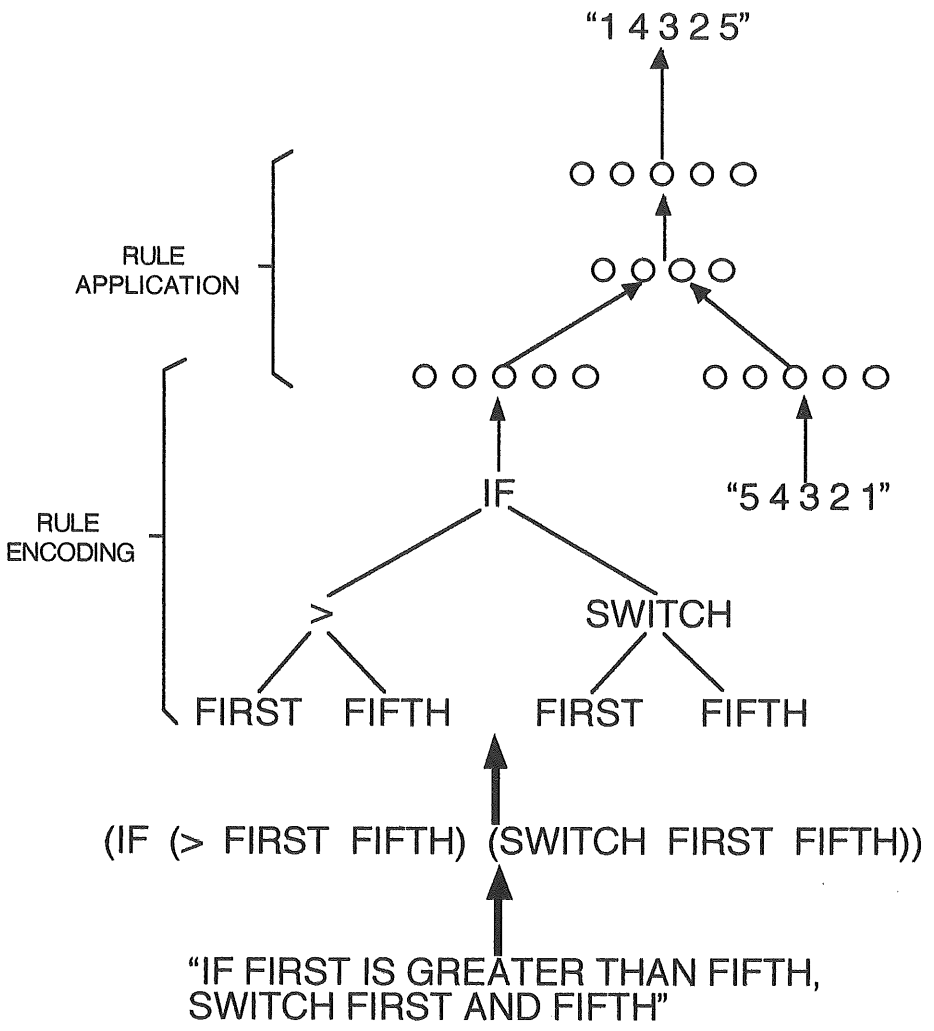

Figure 11. A model of holistic rule application.

References

Collins, A. & Michalski, R. (1990), 'The Logic of Plausible Reasoning', *Cognitive Science*, 13(1), 1-49.

Chalmers, D. J. (1990), 'Syntactic Transformations on Distributed Representations', *Connection Science*, Vol. 2, Nos. 1 & 2, 53-62.

Diederich, J. (1988), 'Connectionist Recruitment Learning', *Proceedings of the Eighth European Conference on Artificial Intelligence*, Munich, Germany.

Eckhorn, R., Reitboeck, H. J., Arndt, M. & Dicke, P. (1990), 'Feature Linking via Synchronisation among Distributed Assemblies: Simulations of Results from Cat Visual Cortex', *Neural Computation*, 2, 293-307.

Elman, J. L. (1988), 'Finding Structure in Time', *Cognitive Science*, 14, 179-211.

Fodor, J. A. (1975), *The Language of Thought*, Thomas Cromwell, New York.

Fodor, J. A. & Pylyshyn, Z.W. (1988), 'Connectionism and Cognitive Architecture: A Critical Analysis', *Cognition*, 28, 3-71.

Hadley, R. F. (1993), 'Connectionism, Explicit Rules, and Symbolic Manipulation', *Minds and Machines*, 3, 201-218.

Hinton, G. E. (1981), 'Implementing Semantic Networks in Parallel Hardware', *Parallel Models of Associative Memory*, Lawrwnce Erlbaum.

Hinton, G. E. (1986), 'Learning Distributed Representations of Concepts', *Proceedings of the Eighth Annual Conference of the Cognitive Science Society.*

Lange, T. E. & Dyer, M.G. (1989), 'High-Level Inferencing in a Connectionist Network', *Connection Science*, Vol. 1, No. 2, 181-217.

Mani, D. R. & Shastri, L. (1992) 'A Connectionist Solution to the Multiple Instantiation Problem Using Temporal Synchrony', *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*.

Miller, G. A. (1956), 'The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information', *Psychological Review*, 63(2), 81-97.

Pollack, J. B. (1988), 'Recursive Auto-Associative Memory: Devising Compositional Distributed Representations', *Proceedings of the Tenth Annual Conference of the Cognitive Science Society.*

Rumelhart, D. E. & McClelland, J. L. (1986), *Parallel Distributed Processing*, Volume 1. MIT Press.

Sejnowski, T. J. (1981), 'Skeleton Filters in the Brain', *Parallel Distributed Processing*, Volume 1, MIT Press.

Shastri, L. & Ajjanagadde, V. G. (1993), 'From Simple Associations to Systematic Reasoning: A Connectionist Representation of Rules, Variables and Dynamic Bindings using Temporal Synchrony', *Behavioral and Brain Sciences*, 16, 417-494.

Smolensky, P. (1990), 'Tensor Product Variable Binding and the Representation of Symbolic Structures in Connectionist Systems', *Artificial Intelligence*, 46, 159-216.

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 Artficial Intelligence.*

Sumida, R. A. (1991), 'Dynamic Inferencing in Parallel Distributed Semantic Networks', *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society.*

Sun, R. (1991), 'Connectionist Models of Rule-Based Reasoning', *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, 437-442.

Sun, R. (1992a), 'A Connectionist Model for Commonsense Reasoning Incorporating Rules and Similarities', *Knowledge Acquisition*, 4, 293-321.

Sun, R. (1992b), 'On Variable Binding in Connectionist Networks', *Connection Science*, 4, 2, 93-124.

Sun, R. (1993), 'An Efficient Feature-Based Connectionist Inheritance Scheme', *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, No. 2, March/April, 512-522.

Sutcliffe, R. F. E. (1992), 'Representing Meaning Using Microfeatures', *Connectionist Approaches to Natural Language Processing*, R. G. Reilly & N. E. Sharkey (Eds.).