Parallelism in Inheritance Hierarchies with Exceptions

Garrison W. Cottrell

Department of Computer Science
University of Rochester
Rochester, YY.

Abstract

In a recent paper, Etherington & Reiter formalized a simple version of semantic networks with exceptions in terms of Reiter's Default Logic. With this approach they were able to formally characterize the correctness of an inference algorithm in terms of Default Logic, and exhibited an algorithm that was correct in this sense. Finally, they concluded that massively parallel architectures for semantic networks, such as NETL apparently cannot implement this algorithm. In this paper, we present a different massively parallel architecture for the simplified semantic networks outlined in their paper which appears to avoid the objections to NETL. We also present some results of simulations in this framework of the examples presented in Etherington and Reiter.

Introduction

In a recent paper, Etherington & Reiter (1983) (hereafter E&R) formalized the inheritance hierarchy subset *of* semantic networks with exceptions in terms of Reiter's (1980) Default Logic. With this approach they were able to formally characterize the correctness *of* an inference algorithm in terms of Default Logic, and exhibited an algorithm that was correct in this sense. Finally, they concluded that massively parallel architectures for semantic networks, such as NETL (Fahlman, 1979), apparently cannot implement this algorithm. In this paper, we present a different massively parallel architecture for the simplified semantic networks outlined in their paper which appears to avoid the objections to NETL. We also present some results of simulations in this framework of the examples presented in E&R.

The Problem

Semantic networks have been found to be an efficient and useful representation of knowledge by AI researchers for many years. One principal advantage is the ability to store information about objects at appropriate levels of abstraction in the IS-A hierarchy, so that the fact that dogs, elephants, and people nurse their young, for example, can be stored once at the MAMMAL node. Retrieving all of the properties associated with an instance of some class is done by an inference procedure that is particularly simple in these systems, known as *inheritance.*

As Hayes (1977) points out. there is an obvious correspondence between IS-A hierarchies and simple collections of FOPC formulas. For example, "Clyde is an instance of an Elephant" corresponds to the assertion Elephant(Clyde). Statements about classes, such as "Elephants are Gray", correspond to first-order formulae, in this case, (x).Elephant(x)=>Gray(x). Inheritance can then be seen as a repeated application *of* modus ponens. One nice property of inheritance hierarchies is that, since they are acyclic, modus ponens can only be applied a finite number of times, no more than the depth of the hierarchy. Also, as pointed out by E&R, the node labels in such hierarchies are unary predicates, e.g. MAMMAL(x). Finally, no exceptions are permitted to inheritance. A dog is a mammal, no matter what.

Unfortunately, the real world is not as simple as a taxonomic hierarchy. Often it is useful to abandon the tree structure in favor of multiple inheritance hierarchies, and to allow *exceptions* to inheritance relations. This introduces non-monotonicity into the representation, as well as ambiguity. An common example *of* a non-monotonic rule is: "assume a particular Elephant is Gray unless proven otherwise." This is often known as *default reasoning* and has been formalized by Reiter (1980). When combined with multiple inheritance, default reasoning can lead to ambiguity. A well-known example is:

0) Nixon is a Quaker.
(2) Nixon is a Republican.
(3) Republicans are normally non-pacifists.
(4) Quakers are normally pacifists.

Reiter's formalization of the above facts would be (assuming, for convenience, that Nixon is a type):

(1) (x).Nixon(x)=>Quaker(x)
(2) (x).Nixon(x)=>Republican(x)

(3) $$\frac{\text{Republican(x)f Pacifist(x)}}{\text{~Pacifist(x)}}$$

(4) $$\frac{\text{Quaker(x):Pacifist(x)}}{\text{Pacifist(x)}}$$

(1) and (2) are just the first order rules corresponding to (1) and (2) above. (3) is an example of a default rule. The formula to the left of the colon is called the *prerequisite of* the default. If this is known, and the part to the right of the colon, (the *justification)* can be consistently assumed (i.e., its negation isn't provable from what we know), then we can infer ~Pacifist(x), the *consequent.* Often, the justification contains all of the exceptions to the rule we know about. In this case, we might add "NRAmember(x)" to the justification of (4).

Is an individual b for which Nixon(b) holds a pacifist or not? In Reiter's terminology, there are two *extensions'* consistent with our knowledge. An extension contains the first order facts and is closed under the default rules as well as first order theorem-hood. One contains Pacifist(b), the other -Pacifist(b). In general, the problem we want to solve is: Given an individual b, and a predicate P known to be true of b, we want to compute P_1b), • • • ,P_n(b) such that the P_i's all he within a single extension. As noted by E&R, we can ignore the unary predicate argument, and the default theory is purely propositional. Fortunately, then, non-provability is computable.

Etherington and Reiter's Algorithm

We briefly review E&R's inference algorithm in intuitive terms. Those interested in the formal details may refer to their paper. The purpose of the algorithm is to "derive conclusions all of which lie within a single extension of the underlying default theory." When faced with multiple extensions, the algorithm randomly chooses one. The algorithm operates by successive approximations to an extension. Starting with the first order facts as a first approximation to an extension, it successively chooses (randomly) default rules which are not blocked by the current approximation *or the previous approximation,* and adds their consequents to the current approximation, until all of them are used. The constraints derived in previous approximations thus propagate to the current approximation. It iterates on this, starting with the first order facts ag. n, until two successive approximations are the same (convergence). Etherington (1983) has proved that this algorithm will always converge on an extension. The randomness is essential to the algorithm's ability to derive any possible extension, if it is run "enough" times. An important point about the algorithm as given is that it can be viewed as a relaxation-style constraint propagation technique.

Unfortunately, NETL is unable to capture such algorithms due to the "one-shot" nature of marker-passing. Markers are propagated through the network to find properties. Cancellation links can block this propagation to implement exceptions to inheritance. The very existence of cancellation links in the version of NETL discussed in E&R (discarded in later versions; see Touretzky, 1984) defeats marker passing because a link can be crossed before it is cancelled from a longer path. See Figures 1(a) and 1(b), reproduced from FAR. In Figure 1(a), F must be reached before B in order to generate the extension properly, and vice-versa in 1(b). It is clear from this that the problem with NETL is not that it is a parallel machine. Rather, the problem is that is a *single pass* marker passing machine.

An Alternate Parallel Approach

An obvious answer to these objections is to relax the "one-shot" nature of the parallel network. Connectionist networks (Feldman & Ballard, 1982), being iterative, have no such restriction. Connectionist models consist of simple processing units connected by links. A unit or node is a computational entity comprised of:

p: a continuous value in [-1,1], called the *potential*
v: an *output,* in the range [0,1.0] in discrete jumps of .1
i:a vector of *inputs,*

and functions for updating these:

$p \leftarrow f(\mathbf{i},p)$
$v \leftarrow g(\mathbf{i},p)$

We will term an application *of* these functions an *update* of the unit. Note that there is no interpreter for a connectionist network; all updates are done locally by each unit in parallel. There are no constraints on the functions that can be used, though they are usually kept simple. It is an important research topic at the moment to discover what constraints on the functions can be reasonably assumed without losing computational ability. In the following model, we show that even with simple updating functions, we can still get fairly powerful results. Finally, note that there is no mention of time in the definition. That is, in simulating such networks, the units could be scheduled for updating in various ways: They could be kept in lock step (synchronous) or they could be updated in random order, with some units perhaps being

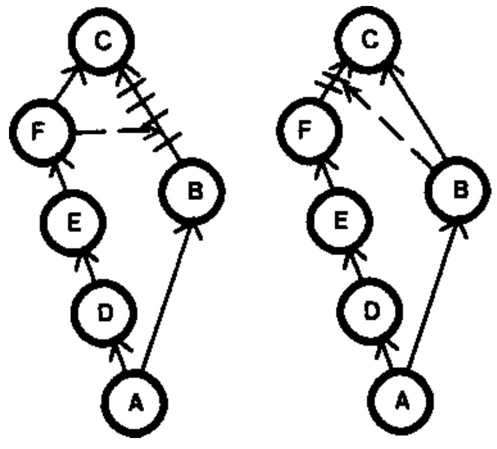


Figure 1. Networks which defeat the shortest path heuristic.

updated several times before another gets a chance to be updated (simulating asynchrony). We use an asynchronous version in the following model.

A *connection* (or link), is an identification of an element of a units input vector with the another unit's output, along with a *weight,* a value between -1 and 1. Any value transmitted on the link is multiplied by the weight before it is passed to the unit. In the following model, we use only use weights of -1 and 1. Links with negative weights are called *inhibitory* links. These are drawn with a small circle at their head in the figures. A special kind of link, *modifier links,* are node-link connections that have the effect that when the unit at their tail has positive output, they block activation from crossing the link at their head. These are also drawn with a small circle at their head, but since they are always incident on other links, there is no confusion between them and inhibitory links.

For convenience, the potential function is then often broken down into two stages: An *evidence* function, which is applied to the inputs, and an *activation* function, which computes the actual potential given the result of the evidence function and the current potential. The activation function usually employs a decay parameter so that if the evidence goes to 0, so does the activation. A *conjunctive connection* is used to refer to two links that must both have non-zero input for the evidence function to pass a non-zero result to the activation function. We will use an output function that thresholds the potential (we use a threshold of 0 in this model, so negative activation is not spread) and rounds it to the nearest tenth. A unit that has non-zero output is called *firing* (or simply, "on").

In the so-called *localist* connectionist models, (Feldman & Ballard, 1982) we represent an object in the domain as a unit or small set of units[1]. The basic idea is that a unit stands for a value of a parameter (the *unit/value principle)* and collects inputs from other units which represent evidence for that value, positive or negative. For example, in vision, (see Ballard, 1984) a unit could represent the presence of an edge at a certain angle at a particular (x,y) coordinate on the retina. The unit's output represents its confidence that there is an edge at the point in the visual field that this unit refers to. Thus, at run time, the unit's output represents a confidence level in a *hypothesis* about the parameter it refers to. An output of 1.0[2] after convergence represents certainty about the parameter value represented by the unit. The links between the units are weighted, reflecting the importance to the receiving unit of the evidence from that link. Much of the information encoded in the network is contained in the connections between units (hence the name "connectionism").

Computation is performed by designating some of the units as *input units.* These are units that may be "clamped on", that is, their output is fixed by the experimenter. Activation is allowed to spread from these, and if the network is well-designed, it converges to a fixed point where no unit's output changes from one iteration to the next At this point the result is read out from the network by the experimenter. Unfortunately, no theory exists at the moment that guarantees convergence of these networks. Analysis is difficult when arbitrary functions are allowed on different units. This is one principle advantage of some connectionist models (Hinton & Sejnowski, 1983) that use uniform functions on all units, where analysis is possible.

Connectionist networks are a natural architecture for solving relaxation style problems. Their "activation passing" is iterative, and constraints between hypotheses can be easily encoded in the networks as positive or negative links between mutually compatible or incompatible hypotheses (represented as processing units). The typical way to go about building connectionist models is to first decide on which elements of the domain we want to model, choose a way to encode those as units, and then to wire the units together in such a way as to encode *constraints* between the elements. Finally, we must choose an appropriate function for combining the evidence. In the following, we present a connectionist model of semantic networks of the kind discussed in E&R. It should be kept in mind that these have a particularly simple form. Properties are not distinguished from type nodes, and there are no two place predicates. For a different formulation of semantic networks in connectionist terms which overcomes these objections, see (Shastn & Feldman, 1984).

A Connectionist Inheritance Model

In E&R, a correspondence was made between the five link types of a semantic network (Strict IS-A and ISNT-A, Default IS-A and ISNT-A, and exception links) and formulae in Default Logic. Since our purpose here is to show that a connectionist network can mimic their inference algorithm, we start with formulae from Default Logic that correspond to inheritance axioms and display the corresponding bits of network. The first step, however, is to choose a representation of the predicates. Following the unit/value principle, we will start with two units for every predicate P, called +P and ~P, representing the two different possible assignments of truth values to those predicates. When computing an extension, a node that is firing (after convergence) represents that it is part of the extension. There is an immediate consistency constraint between these two nodes, i.e., they should not both be on in any stable state. Thus we should make them mutually inhibitory. However, a unit that has evidence should be allowed to

[1]See (Hinton. 1981) for a distributed connectionist approach to semantic networks.

[2]Or, the maximum possible output after decay If we use a thresholded potential for the output, and the activation function employs decay, a unit's maximum output is reduced by the decay factor.

propagate that evidence before being inhibited. This is essential if we are to consider all possibilities in parallel. Thus we introduce a third unit, #P, (to use Touret/kys notation, if not his semantics), which represents "inconsistency". See Figure 2. This node inhibits +P and ~P if *both* of them are firing (by using a conjunctive connection). It outputs the maximum of the two, inhibiting both +P and -P[3]. Thus this introduces a delay in the inhibition between +P and -P.

The important point about this design is that this subnetwork of three units representing a predicate has only three stable states:

(1) They are all off.

(2) + Pis on.

(3) ~P is on.

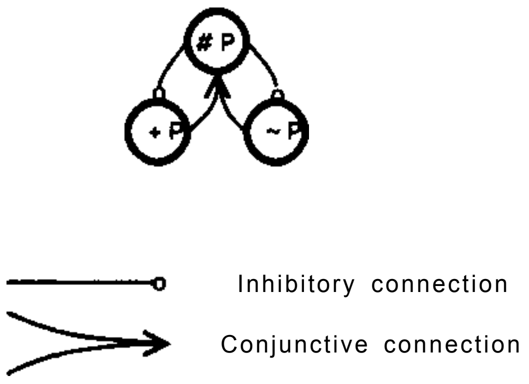
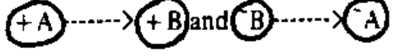


Figure 2. The Spock representation of the predicate P.

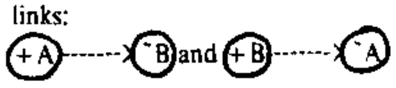
Any other configuration of activation will cause changes in some unit's activation. If only #P is on, it will go off. If only one of the others are on as well, then #P will also go off. If both are on, then #P will inhibit them until one goes off. If #P is off and both of the others are on, #P will come on. Thus if the network is at a fixed point, the predicate P is either assigned the values *true* (+ P is on), *false* (-P is on), or *don't care* (neither is on).

We make the following correspondence between the four formulae defined in E&R, and networks in the connectionist framework. All links shown have weight 1.

(1) (x).A(x) => B(x): This corresponds to two links:

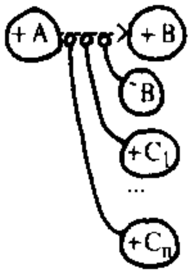


(2) (x).A(x) => ~B(x): This corresponds to two links:

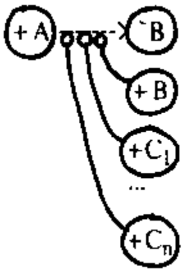


[3]An alternate formulation would have been to use two of these units so that the stronger unit would not receive its own inhibition, but the opposing units. We intend to explore this option in the future.

(3) $\dfrac{A(x)\text{: } B(x)\ \&\ \text{~}C_1(x)\ \&...\&\ \text{~}C_n(x)}{B(x)}$ (where n is possibly 0):



(4) $\dfrac{A(x)\text{: } \text{~}B(x)\ \&\ \text{~}C_1(x)\ \&...\&\ \text{~}C_n(x)}{\text{~}B(x)}$ (where n is possibly 0):



Our encoding of an inference rule is thus just to put a positive link from the antecedent to the consequent. This is the first link in the encodings of (1) and (2), and it guarantees that if + A comes on, eventually +B (or ~B, as appropriate) will come on as well. Since these correspond to first order facts, we don't allow any exceptions to these (i.e., modifier links - see below). In an attempt at some semblance of completeness, we encode the contrapositive as well (remember we have no interpreter to deduce these consequences). This is the motivation for the second link in the encodings of (1) and (2). We are explicitly adding the rule -B => ~A (or B => "A, in (2)). In E&R's formulation, extensions are closed under first order inference. This has some odd consequences, which we will see in the Cephalopod example.

In the encodings of (3) and (4), there are no "backwards" links like these, because the contrapositive doesn't hold for default inferences. However, "B in (3) (as well as any of the -t-Cj's) for example, should block the inference of + B if it is on, so we use a modifier link from ~B to the link between +A and +B. This blocks activation from crossing the link if "B has any output at all. Thus the modifier link seems a good choice for encoding the semantics[4].

Now, suppose we have encoded the rules (1) or (3) as above. The inference of B from A corresponds to + A being active, which activates + B. In the case of (3), this

[4]Modifier links have not been implemented in the ISCON (Small et al. 1982) simulator. The actual implementation at least doubles convergence times. See (Cottrcll. 1984) for details.

inference may be retracted later if ^B is inferred by some other (necessarily first order) rule. A default rule will not be able to infer ~B, since this will be blocked by the modifier link from +B. The retraction is accomplished by using activation functions that cause a unit's potential to go to 0 if its evidence goes to 0. In the case of (1), the inference may be retracted if the original inference chain that lead to 4-A included a default inference that was retracted.

An extension of a predicate P is computed by treating +P as an input unit, that is, clamping it on, and then letting the network converge. An extension of P the corresponds to all of the predicate units that are on in a stable state of the network, with the following caveat: Some first order facts may not be relevant to the computation of the extension. These will not be activated at all. Also, all facts derivable from the extension will not be included (e.g., A => A or B). So the actual extension is the logical closure of the union of W and the active predicates in the stable state.

Finally, we specify what the units compute. An evidence function which appears reasonable for our purposes is to take the maximum of all positive input (from subtypes) and add the minimum of all inhibitory inputs. The motivation for this rule is that we will use one "source" for the network's activation, namely the predicate whose extension we are seeking, and so it does not seem appropriate to use more evidence than the maximum from that source. However, there are arguments against this. In a non-demonstration system, one would want an alternate way to combine evidence, for example Dempster-Shafer rules (see Ginsberg, 1984), for some extensions of Dempster-Shafer rules for semantic networks). This is especially important for default rules. If someone is a Republican *and* an NRA member *and* a veteran then we would be more inclined to assume they are not a pacifist, even if they are a Quaker. See Shastri and Feldman (1985) for a formal theory of evidence that incorporates these considerations. Anyway, for the examples we will be discussing, the max and min rule appears sufficient.

The result of the evidence function is passed to the activation function. We have implemented two activation functions in this system. One uses table lookup and looks almost like iterative marker passing (it only uses three values), and the other is more continuous valued. The first function appears in Table 1. The basic idea is to move towards the value of your input. This version of the system we call Spock because it doesn't allow intermediate interpretations. A unit is on, or it isn't. The second uses the activation function in Table 2 (due to McClelland & Rumelhart, 1981). In this function, E is the result of the evidence function, p is the potential and d is a decay constant (we used .2 in the simulations). We call this version Dr. Spock because it is more permissive, allowing intermediate values. If one had no decay, then

Table 1. The Spock Activation Function

Evidence	Current Pot.	New Pot.
0	0	0
0		0
0	-1	0
1	0	I
1		1
1	-1	0
-1	0	-1
-1		0
-1	-1	-1

Table 2: The Dr. Spock Activation function

p <- p + [if E < 0 then E*p else E*(1 - p)] - d*p

starting the system with a 1 on the assumed unit, and only using weights of 1 or -1 on links, then the result is that units only take on the values 0 or 1. The problem with this is that if a unit is at 1, then it will stay there if it is not inhibited, causing false inferences to stay around. The desire to try to have a system with no decay led to the table lookup function above.

We should state at this point that there is a major difference between F&R's algorithm and the following implementation[5]. We claim here, without proof, that if the network only encodes a consistent set of first order (inheritance) rules, we can allow all first order rules to fire in parallel and the network converges. An interesting question is whether we can allow *all* inferences to proceed in parallel, including the default inferences, while relying on our predicate networks to guarantee consistency. While at this stage we have no proof of convergence or correctness, experimental results with the system support the conjecture. E&Rs algorithm stipulates introducing one default rule at a time, generating all inferences, and then trying another, so we depart from their algorithm in this. The difference is that we don't wait for first order consequences to propagate before we try another default. We use a random update order (simulating asynchrony) to allow one default to run before another, as in E&R's algorithm. If they are "competing" defaults (as in the Nixon example), this ensures one will block the other, so this is basically a tie-breaking strategy. Another method would be to use noise in a synchronous network to break ties.

Simulation Results

We present the results of simulating several of the networks from E&R. In the following, the results are from the Spock version except where noted. This is because in almost all cases, the results from the two systems were similar. As noted above, we use a random update order. That is, units are all equally likely to be

[5]For a sketch of how to mimic their algorithm exactly, see (Cottrell 1984).

updated at any point. An "iteration" consists of doing as many updates as there are units in the network. Note that this doesn't mean that all units have been updated; some may have been updated more than once, others not at all.

We begin with the Cephalopod example from E&R, shown in Figure 3. The source of this example is (Fahlman et al, 1981). In English, it's:

Molluscs are normally shell-bearers.

Cephalopods must be Molluscs but normally are *no!* shell-bearers.

Nautili must be Cephalopods and must be shell-bearers.

The default theory corresponding to this (as given in E&R) is:

$$\{\frac{M(x)\text{: } Sb(x)\ \&\ \tilde{}C(x)}{Sb(x)},\ (x).C(x) \models\!> M(x),\ (x).N(x) \models\!> C(x),$$
$$\frac{C(x)\text{: } \tilde{}Sb(x)\ \&\ \tilde{}N(x)}{\tilde{}Sb(x)},\ (x).N(x) \models\!> Sb(x)\}.$$

The connectionist implementation of these rules is given in Figure 4. Note the "downward" link from *-S~B* to -N, encoding the contrapositive.

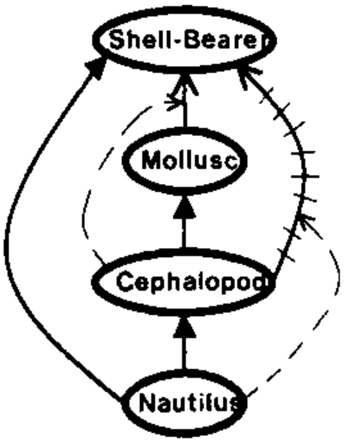


Figure 3. E&Rs network representation of Cephalopod facts.

For this example we spread activation from the + Cephalopod node in order to find the extension of Cephalopod. To shorten the tables, only iterations where one of the units in the table changed are shown. This is a "typical" execution, although of course they are rarely the same. We simulated this network about ten times, and it never took longer than 15 iterations to converge. As the results in Table 3 show, activation spreads from + Cephalopod and activates "Shell-Bearer, which blocks the IS-A link from + Mollusc to + Shell-Bearer.

An interesting result here is that "Nautilus is inferred! Since E&R require that "an extension ... is closed under the Defaults of D as well as first order theoremhood", then they will have to live with this. The inference came about because -Nautilus was consistent, allowing us to infer "Shell-Bearer from +Cephalopod.

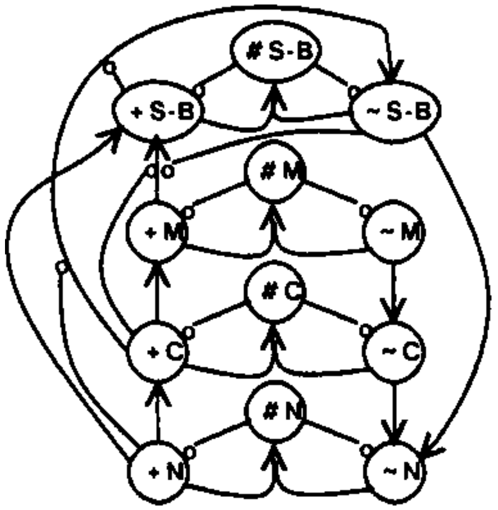


Figure 4. The Spock implementation of the knowledge in Figure 3.

However, since Nautilus => Shell-Bearer, first order, then ""Shell-Bearer => -Nautilus, and we get the result that we prove "Nautilus from assuming it to be consistent. This is not unintuitive, since if Cephalopods are usually not Shell-Bearers, then they are usually not Nautili. This is just not what we expect from an inheritance hierarchy. Not including such "downward" inferences would eliminate completeness, but would also eliminate a problem with the final example.

Table 3. Trace of Unit Outputs from Example 1

	Activating Cephalopod				Activating Nautilus			
Iteration	2	3	4	5	3	6	8	9
+ Nautilus	0	0	0	0	1	1	1	1
"Nautilus	0	0	1	1	0	0	0	0
. + Cephalopod	1	1	1	1	0	0	1	1
+ Mollusc	0	0	0	1	0	0	0	1
+ Shell- Bearer	O	0	0	0	0	0	1	1
- Shell- Bearer	0	1	1	1	0	0	0	0

If we activate + Nautilus instead, activation spreads to + Cephalopod and + Shell-Bearer, and + Nautilus cancels the IS-A from + Cephalopod to "Shell-Bearer, resulting in the correct extension.

If we use the default theory relating to the NETL version of this hierarchy[6] given in E&R (see Figure 5), we get an ambiguity with respect to whether a given Cephalopod has a shell or not, since the default IS-A from + Mollusc to + Shell-Bearer is not cancelled. Also, in this version all inferences are defaults, so "Nautilus is never inferred. Our network shows a marked preference for shortest paths in this case. In 20 runs, we got "Shell-

[6]Note that in NETL (as of 1979), all inferences arc defaults and only exceptions to ISNT-A's are allowed.

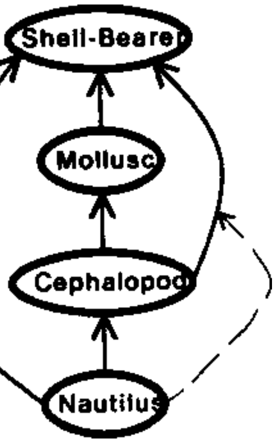


Figure 5. The NETL version of the Cephalopod example. All inferences are defaults.

Bearer 18 times and + Shell-Bearer only twice. This is not surprising, given that the activation is most likely to follow the shorter path first. (In simulations of the Nixon example described earlier, where the paths are of equal length, the the results were 50-50 between the two extensions.)

In example 3 we show that we overcome the fixed radius problems of NETL. The networks in Figure 1 defeat any shortest path algorithm. As Table 4 shows (left hand side), since activation is computed continually, the effects of node + F are felt when it gets activated, and the IS-A from +B to X is cancelled, allowing +C to become active. Note that, because this is all default inferences, +C can't be inferred until C goes off, because -C blocks the inference of +G Table 4 shows that the network of Figure 1(b) works as well. In this case, it is practically impossible for X to come on, since + B is almost always inferred early, blocking the inference of X. If it did the network would still converge as it did for the network of Figure 1(a). In this case, the Dr. Spock version was a little slower, because X took several iterations to decay to 0. The modifier links are strict, so that any output from X stops default inference of +C.

In all of the previous examples, the results from the two activation functions were essentially the same. The final example shows that this is not always the case. The example is given in Figure 6. It is nearly identical to the example in Figure 1(a), except that the inferences are all first order in the left hand chain. This example looks harmless because it has a unique extension. However, where the Dr. Spock activation function takes only slightly more than 100 iterations to converge on the extension, the Spock function required over 1000 iterations! To see why, recall that first order IS-A's allow downward inference of -P's. (Both A => B and A => "B result in contrapositives with negative consequents). Secondly, the delay in inhibition between a "+" node

Table 4. Trace of Unit Outputs from Example 3

	Figure 1(a)							Figure 1(b)					
Iteration	2	5	7	11	14	16	25	2	4	5	6	8	11
+ A	1	1	1	1	1	1	1	1	1	1	1	1	1
+ B	0	0	1	1	1	1	1	0	1	1	1	1	1
+ C	0	0	0	0	0	0	1	0	0	0	1	1	1
~C	0	0	0	1	1	0	0	0	0	0	0	0	0
+ D	0	1	1	1	1	1	1	0	0	1	1	1	1
+ E	0	0	0	1	1	1	1	0	0	0	0	1	1
+ F	0	0	0	0	1	1	1	0	0	0	0	0	1

and a "*"" node induced by the "#" node allows inference chains to "pass" each other. What happens in this case is that the default inference of X starts a downward chain of "not" inferences. This meets an upward chain of positive inferences. They pass each other, then the consistency constraints begin turning off both 4- and predicate nodes in the middle of the chain. The inference chains then meet rather incorrigible resistance at both ends. +G has a hard time blocking the inference of X because by the time it gets there, its support is falling apart behind it. If it fails, X is inferred again straightaway, since + A is clamped on, and + P follows from that. Then the process starts over again. Dr. Spock appears to be able to break out of this trap because the units are less "all or none". Because +G decays slowly, it is more likely to be on when +P tries to infer X, since any nonzero output from +G blocks the inference. On the other hand, Spock's activity resembles a search for a stable pattern of assignments o\^ 0 or 1 to the units, which takes a lot longer.

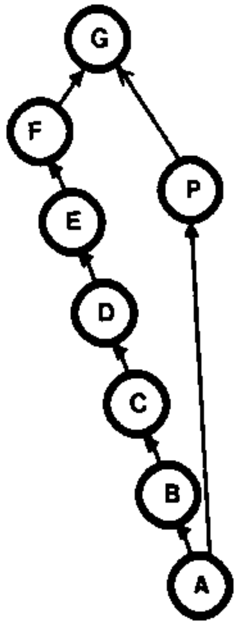


Figure 6. This one causes Spock some trouble (but not the Dr.).

This behavior of the Spock function results from the fact that that the model does not wait for the results of first order knowledge to propagate before applying a default rule, and it took a while getting to the point *of* disagreement. In all of the other examples, the ambiguity was very localized. Local ambiguities are not hard for connectionist networks to handle. It is ones that depend on global properties of the network that are hard to deal with and still maintain a small radius *o(* communication (an unspoken assumption in most connectionist models). There are several ways this problem could be avoided.

A change to the model that may speed convergence on such examples is to make default inferences literally less strong than first order ones. If a default inference link is weighted by .5 instead of 1, then a first order inference could more easily overcome it. In this case, we need an activation function that converges to its evidence, to propagate the .5 value (unfortunately, this is not a property the McClelland & Rumelhart function (called Dr. Spock here), enjoys). The function:

$$p \leftarrow p*(1-d) + E*d$$

achieves this[7]. Experiments with this scheme have been encouraging. The table lookup function could also be altered to reflect this other value. First order inferences which had a default at their source would then reflect that fact in their potential. In this way information that was non-local could be encoded in the signal. This still doesn't avoid the problem altogether. 1 here is no reason why a default inference could not be at the base of each chain.

A second way to avoid this problem in the Spock version which avoids this pitfall is to disallow downward inferences altogether, by going the way of NETL, and assuming that everything is a default inference. (Or by foregoing our attempt at completeness; hence not encoding the contrapositive.) This does avoid the problem of two default inferences being at the bottom of competing chains. The competition at the top is a local one, since either outcome is consistent. Adding weights reflecting belief strengths, as advocated by Rich (1983), might make such a system a possible cognitive model.

Conclusions

We have seen how a connectionist model of inheritance mimics E&R's inference algorithm, avoiding the problems of NETL. So, a massively parallel inheritance scheme apparently *can* work. Two caveats should be mentioned. First, this is within the context of a very simple characterization of semantic networks. Second, the examples are only an informal (engineers) argument for correctness. What is missing from this presentation is a formal proof of correctness. A first cut would be to show that if the network is in a stable state, then the predicate units that are firing along with the original W represent the "seed" *of* an extension (i.e., their closure is the extension), leaving the problem of convergence for another time.

A second point is that the choice of activation function can make a big difference in the speed of convergence of the network. The current results appear to favor smoother activation functions over the relatively discrete Spock version. Also, an interesting avenue for exploration now is using weights on the links to encode default strengths. This could also have a speed-up effect on convergence, and could possibly be an interesting cognitive model. Finally, incorporating an evidence function that would better reflect the contribution *of* multiple sources of evidence is left for future research.

[7]It turns out (unbeknownst to us when we derived it) that this is the function used by McClelland in his (1979) Cascade model.

Acknowledgements

I would like to thank James Allen and Jerry Feldman for many helpful discussions on this paper, and Mark Fanty for prompt addition of the asynchronous facility to the C version of ISCON. This document greatly increased in clarity through the comments of one Patrick J. Hayes, who nevertheless may be dismayed at this association with "connectionism". This work was supported by grants IST-8208571 and MCS-8203920 from the National Science Foundation.

Bibliography

Ballard, D.H. "Parameter networks." *Artificial Intelligence, 22,* 235-267, 1984.

Cottrell, G.W. "Re: On inheritance hierarchies with exceptions", in *Proceedings of the Workshop on Non-Monotonic Reasoning,* New Paltz, N.Y., October 1984.

Etherington, D. "Formalizing non-monotonic reasoning systems". TR 83-1, Department of Computer Science. University of British Columbia, 1983.

Etherington D. and R. Reiter "On inheritance hierarchies with exceptions", in *Proceedings of the National Conference on Artificial Intelligence,* Washington, D.C., August 1983.

Fahlman, S. E. "NETL: A System for Representing and Using Real-World Knowledge". MIT Press, Cambridge, Mass, 1979.

Fahlman, S.E., Touretzky, D.S., and W. van Roggen. "Cancellation in a parallel semantic network.'Tn *Proceedings of the Seventh International Joint Conference on Artificial Intelligence,* Vancouver, B.C., 1981.

Feldman, Jerome A. and Dana Ballard. Connectionist Models and their Properties, *Cognitive Science,* 1982, *6* 205-254.

Ginsberg, Matthew W. "Non-Monotonic Reasoning Using Dempster's Rule." In *Proceedings of the National Conference on Artificial Intelligence,* Austin, Texas, August, 1984.

Hayes, P. J. In defense of logic. In Proceedings of the Fifth Annual International Joint Conference on Artificial Intelligence, Cambridge, Mass., 1977.

Hinton, G.E. Implementing semantic networks in parallel hardware. In G. E. Hinton, & J. A. Anderson (Eds.), Parallel models of associative memory, Hillsdale, N.J.: Lawrence Erlbaum Associates, 1981.

Hinton, G.E. and T. Sejnowski. Optimal perceptual inference. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Washington, D.C 1983.

McClelland, J.L. On the time relations of mental processes: An examination of systems of processes in cascade. Psych. Review. 86,

McClelland, J.L. and D.E. Rumelhart. An interactive activation model of the effect of context in perception: Part I, An account of basic findings. Psych. Review, 88,

Reiter, Ray "A Logic for Default Reasoning", Artificial Intelligence 13, 1980, pp 81-132.

Rich, E. "Default reasoning as likelihood reasoning." In Proceedings of the National Conference on Artificial Intelligence, Washington, D.C, August 1983.

Shastri, L. and Feldman, J.A. Semantic networks and neural nets. T.R. 131, Dept. of Computer Science, University of Rochester, May 1984.

Shastri L. and J.A. Feldman. Evidential reasoning in semantic networks: A formal theory. In this volume. 1985.

Small, S. L., Shastri L., Brucks M., Kaufman S., Addanki, S. and Cottrell, G.W. ISCON: An Interactive Simulator For Connectionist Networks, Technical Report 109, Department of Computer Science, University of Rochester, Dec. 1982.

Touretzky, David S. "The Mathematics of Inheritance Systems." Ph.D. Thesis, Carnegie Mellon University 1984. Available as T.R. CMU-CS-84-136.