Cables, Paths and "Subconscious" Reasoning in Propositional Semantic Networks*

Stuart C. Shapiro
Department of Computer Science
State University of New York at Buffalo
226 Bell Hall
Buffalo, NY 14260-7022
(716) 636-3182
shapiro@cs.buffalo.edu

1 Introduction

In this paper, I will discuss two aspects of SNePS propositional semantic networks [5, 8, 12, 17] that distinguish them as formalisms for the representation of knowledge—cables and paths. I will also discuss a kind of inference sanctioned by each one—reduction inference and path-based inference, respectively, and the integration of these two kinds of inference into a kind of "subconscious" reasoning.

Informally, a semantic network is a labelled directed acyclic graph in which nodes represent entities and labelled arcs represent binary relations between entities. A *propositional* semantic network is a semantic network in which every proposition represented in the network is represented by a node, rather than by an arc. We will refer to a node that represents a proposition as a *propositional node*. Isolated nodes are not allowed in a semantic network, and since a semantic network is a variety of relational graph, it does not make sense to have two arcs with the same label emanate from the same node and terminate at the same node. However, there is no restriction forbidding several arcs with the same label from emanating from the same node if they terminate in different nodes. Informally, we will call a set of such arcs a *cable*. (We will formalize this below.) A propositional node, therefore, may have a set of cables emanating from it. Each cable represents an argument position of the proposition represented by the propositional node, the label

*This is a preliminary version of S. C. Shapiro, Cables, paths and "subconscious" reasoning in propositional semantic networks, in J. Sowa, Ed. *Principles of Semantic Networks: Explorations in the Representation of Knowledge.* Morgan Kaufmann, San Mateo, CA, 1991, 137–156. All quotes should be from, and all citations should be to that published version.

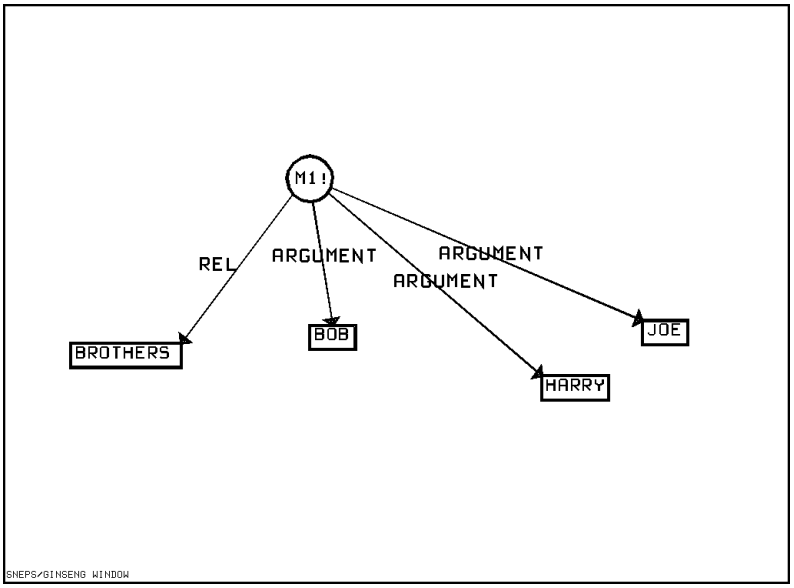

Figure 1: A SNePS network representing the proposition that Bob, Joe, and Harry are brothers. The proposition node `M1!` has two cables emanating from it, one consisting of the `REL` arc and the node it goes to, the other consisting of the `ARGUMENT` arcs and the nodes they go to.

associated with each cable is the keyword that identifies the argument position, and the nodes of each label are the arguments in that position. A proposition with multiple arguments in a single position is not a situation that occurs in the standard syntax of predicate logic. For example, Figure 1 shows a diagram of a SNePS network in which `M1!` is a propositional node from which emates two cables, a cable consisting of one `REL` arc going to the node `BROTHERS` and a cable consisting of three `ARGUMENT` arcs going to the nodes `BOB, JOE,` and `HARRY.` As discussed in [13], `M1!` is intended to represent the proposition that Bob, Joe, and Harry are brothers. In this proposition, the order in which "Bob," "Joe," and "Harry" appear is entirely arbitrary—they are really in the same argument position relative to the relation "are brothers," and this is captured by putting the nodes that represent them in the same cable from the node `M1!.`

Path-Based inference [11, 18] involves the inferring of an arc between two nodes from the existence of a path of arcs between the same two nodes. Since this inference ignores the other arcs emanating from the starting node, it corresponds to an inference rule that ignores the arity of the atomic propositions. This, also, is not a situation that can occur in the standard syntax of predicate logic.

We might consider the labels of semantic network arcs to be binary predicates and the nodes to be individual constants. Then an arc in the network corresponds to a ground atomic formula in which the label-predicate is applied to the two node-individual constants. A cable is then just the set of all such atomic formulas for which the predicate and the first argument are the same. The semantics of a network is then derived by taking all the atomic formulae conjunctively, and path-based inference rules are straight-forward

conditionals. This translation actually gives us a model of the network, rather than another syntax for the same network. The differences include:

- In the atomic predicate version, there is nothing to prevent the situation from occurring in which there are two individual constants, n_1 and n_2 such that $\forall(P,x)[P(n_1,x) \Leftrightarrow P(n_2,x)]$. As we shall see below, this conflicts with the Uniqueness Principle, and cannot occur in a SNePS network the way SNePS networks are defined in this paper.
- In the atomic predicate version, there is nothing to prevent one from adding a new formula $P(n_1,n_2)$ to the database at any time even though there are already formulae whose first arguments are n_1. As we shall see below, this is severly restricted in SNePS.

Nevertheless, the atomic predicate model does suggest that a cable, as a conjunction of formulae, should imply a proper subset of itself. So, for example, node `M1!` of Figure 1 implies that Bob and Harry are brothers, that Bob and Joe are brothers, and that Harry and Joe are brothers[1]. We will adopt a version of this kind of inference, calling it *reduction inference,* in such a way that it does not conflict with the Uniqueness Principle. Reduction inference may involve a kind of arity reduction, and so, is intimately tied up with path-based inference.

2 A Motivating Example

Figure 2 shows a small SNePS network containing the information that Rover is a dog, and that dogs are animals. What each node is supposed to represent is shown in the following table:

`ROVER`	Rover
`DOG`	the class of dogs
`ANIMAL`	the class of animals
`M1!`	the proposition that Rover is a dog
`M2!`	the proposition that dogs are animals

If we want inheritance of classes to be handled by path-based inference, we could give SNePS a path-based inference rule that would sanction the inference of a `CLASS` arc from `M1!` to `ANIMAL`. This past sentence, however, is informal. The attempt to formalize path-based inference raises the following issues and questions:

[1] For the reason that it does not imply that Bob and Bob, Harry and Harry, and Joe and Joe are brothers, see [13].

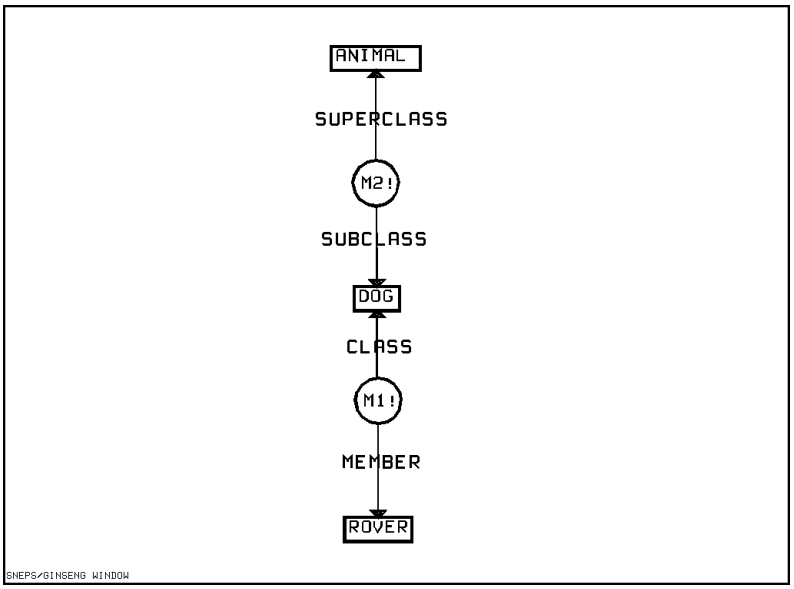

Figure 2: A SNePS network containing the information that Rover is a dog, and that dogs are animals.

- We would want the system whose "knowledge base" is shown in Figure 2 to act as if it had already stored the information that Rover is an animal. *I.e.,* the network shown in Figure 2 already contains that information. In what way is this so?

- SNePS does not allow a user to add a new arc coming out of an existing node, because that would change the entity represented by the node into another entity. Is this not being done by the inference of a new `CLASS` arc emanating from `M1!`?

- One might answer the previous question "No" because that arc (in some sense to be determined by answering the first point above) already exists in the network. But that raises the question of what the structure of node `M1!` actually is, and what the relationship is among the following three nodes[2]:

Node	with `MEMBER` arc to	and with `CLASS` arc to
$\texttt{M1}_1$	`ROVER`	`DOG`
$\texttt{M1}_2$	`ROVER`	`ANIMAL`
$\texttt{M1}_3$	`ROVER`	{`DOG, ANIMAL`}

  By the Uniqueness Principle, they should be different nodes, and should represent different entities. So which one appears in Figure 2, and what is the relationship among them?

In this paper, we will develop the following answers:

[2] The name of a propositional node is of the form `M`n, where n is some integer. A "!" is appended to the name to indicate that the proposition represented by the node is asserted (taken to be true) in the network. However, the "!" does not affect the identity of the node, nor the proposition it represents.

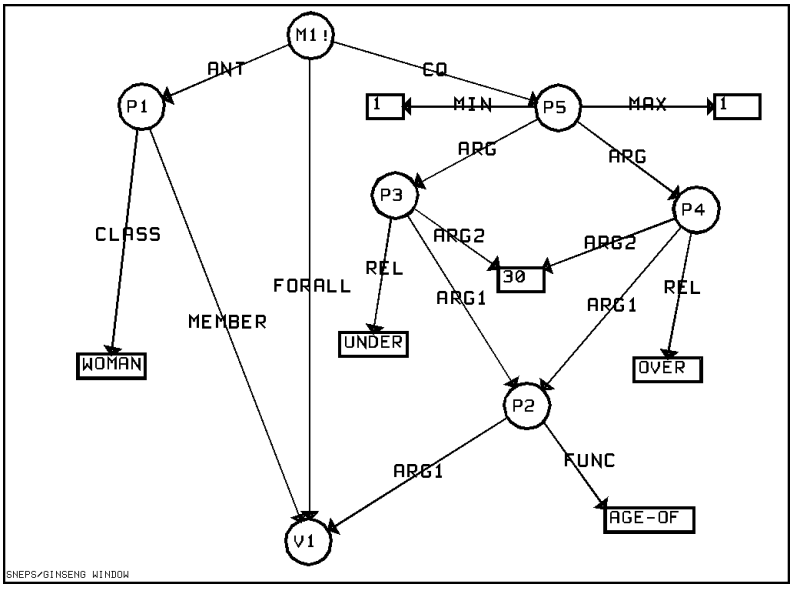

Figure 3: A SNePS network expressing the node-based inference rule that every woman is either under 30 or over 30.

- $\texttt{M1}_1$, $\texttt{M1}_2$ and $\texttt{M1}_3$ are different nodes, and represent different entities.
- $\texttt{M1}_1$ and $\texttt{M1}_2$ are *reductions* of $\texttt{M1}_3$, and are implied by it by reduction inference.
- The network of Figure 2 contains $\texttt{M1}_1$ explicitly, and, given the path-based inference rule, "virtually" contains $\texttt{M1}_2$ and $\texttt{M1}_3$.

3 Node-Based Inference

It should not be inferred from anything in this paper that the only kinds of inference SNePS supports are reduction inference and path-based inference, although those are the only two that will be discussed here in any detail. SNePS also suppports node-based inference [11]. Figure 3 shows a SNePS network expressing the node-based inference rule that every woman is either under 30 or over 30, taken from the presentation in [6] of the logic puzzle, "The Woman Freeman Will Marry" from [19, p. 6]. What the principal nodes of Figure 3 are intended to represent are shown in the following table:

V1	An arbitrary woman, v_1.
P1	The proposition that v_1 is a woman.
P2	The age of v_1.
P3	The proposition that v_1 is under 30 years old.
P4	The proposition that v_1 is over 30 years old.
P5	The proposition that v_1 is either under 30 or over 30 (not both).
M1!	The rule that every woman is either under 30 or over 30.

The logic that SNePS supports for node-based inference is discussed in [6].

4 The Agent

I will present the semantics of SNePS networks in terms of an "agent." The agent has beliefs and performs actions (see [9]). Such an agent is (a model of) a cognitive agent [17].

Among the actions the agent can perform is the new believing of a previously not believed proposition. Rather than having the agent be logically omniscient [3] (*i.e.,* believe all the logical consequents of its beliefs), at any time, the agent will believe only those logical consequents of its beliefs that it has come to believe by "consciously" performing the act of believing them, or those it has come to believe by thinking of them after already "subconsciously" believing them.

5 The Domain of Interpretation

SNePS nodes are terms of a formal language. The interpretation of a node is an object in the domain of interpretation, $\mathcal{D}$. In this paper, the members of $\mathcal{D}$ will be called "entities": "A being; esp., a thing which has reality and distinctness of being either in fact or for thought" [7, p. 275]. Every SNePS node denotes an entity. If n is a SNePS node, $[\![n]\!]$ will denote the entity represented by n.

In our recent work with SNePS, we distinguish four types of entities: individuals, propositions, acts, and rules. Propositions are characterized by being the kind of entities an agent may or may not believe. Acts (see [15, 16]) are characterized by being the kind of enities an agent may or may not intend to perform. Rules are, in some ways, like both propositions and acts. In order for a rule to "fire," it must be believed, the agent

must intended to apply it, and its (appropriate) antecedents must be believed. When a rule fires, the agent forms the intention of believing its consequents. Intending to apply a rule is what is called "activating" a rule in [14].

Individuals include everything that is neither a proposition nor an act, *i.e.* that is neither the kind of entity that can be believed, nor the kind of entity that an agent could intend to perform. Thus, individuals include not only traditional individuals, but also classes, properties, relations, etc.

SNePS nodes are typed according to the type of entities they represent. Thus, there are four types of nodes—individual nodes, proposition nodes, act nodes, and rule nodes. As research proceeds, there may be a need to distinguish other types of entities, but at this time propositions, acts, rules, and individuals are the only types we have found a need for.

In this paper, I will only discuss individual and proposition nodes. When the need arises, I will refer to SNePS restricted to individual and proposition nodes as SNePS$_P$ (for propositional SNePS).

6 Meta-Predicates

In formalizing SNePS, we need a set of meta-predicates. These will not necessarily be represented in SNePS, although if the agent, itself, were engaged in the appropriate philosophical reflection, it could conceive of them. The meta-predicates we will need include *Conceive*, *Believe*, and =. Others will be introduced subsequently.

Letting n, n_1, and n_2 be meta-variables ranging over nodes, and p be a meta-variable ranging over proposition nodes, the semantics of the meta-predicates listed above are:

Conceive(n) means that the node n is actually constructed in the SNePs network, and that the agent has conceived of, or thought of, or thought about $[\![n]\!]$. *Conceive* is similar to, but different from Fagin and Halpern's awareness functions [2]. They gloss $A_i\phi$ as, "'i is aware of ϕ,' 'i is able to figure out the truth of ϕ,' or even (when reasoning about knowledge bases) 'i is able to compute the truth of ϕ within time T.'" Here, *Conceive*(n) may be true without the agent's being able to figure out the truth of $[\![n]\!]$.

Believe(p) means that the agent believes the proposition $[\![p]\!]$. (In which case, we say that p is an *asserted* node.)

$n_1 = n_2$ means that n_1 and n_2 are the same, identical, node.

Only conceived of entities may be believed. This is captured in the following Axiom:

Axiom 1 $Believe(p) \Rightarrow Conceive(p)$

7 Arcs and Relations

SNePS nodes are connected to each other by labelled, directed arcs. The labels are drawn from the set of *SNePS Relations*, which can be added to by the user of SNePS in the design of a particular agent. Isolated nodes cannot be constructed in SNePS; neither can cycles of arcs.

8 Types of Nodes

Besides the categorization of nodes into individual nodes and proposition nodes, nodes can also be categorized into base nodes and molecular nodes. The two categorizations of nodes are orthogonal, so there are four types of nodes. As a heuristic aid to understanding, base nodes approximately correspond to individual constants in a standard Predicate Logic and molecular nodes to sentences and functional terms. However, remember that all nodes are terms in SNePS.

8.1 Base Nodes

Base nodes have no arcs emanating from them. Each base node represents some entity of the appropriate type. An individual base node represents an individual entity and a propositional base node represents a proposition. No two base nodes represent the same entity. This is the Uniqueness Principle of [5] for base nodes.

Axiom 2 (Contrapositive of Uniqueness Principle) $n_1 \neq n_2 \Rightarrow [\![n_1]\!] \neq [\![n_2]\!]$[3]

[3] Note that "=" is overloaded to represent both identity of nodes and of entities.

8.2 Molecular Nodes

Informally, a molecular node has one or more labelled, directed arcs emanating from it, each labelled by a relation in the set of SNePS Relations, and each going to another node. Two or more arcs may go from one node to one other node, as long as each arc is labelled with a different label.

In this paper, we will formally define molecular nodes using the cableset approach of [8].

Definition 1 *A* **wire** *is an ordered pair* $\langle r, n\rangle$*, where* r *is a SNePS relation, and* n *is a SNePS node. We will let the meta-variables* $w, w_1, w_2, \ldots$ *range over wires.*

Definition 2 *A* **cable** *is an ordered pair* $\langle r, ns\rangle$*, where* r *is a SNePS relation, and* ns *is a non-empty set of SNePS node. We will let the meta-variables* $c, c_1, c_2, \ldots$ *range over cables.*

Definition 3 *A* **cableset** *is a non-empty set of cables,* $\{\langle r_1, ns_1\rangle, \ldots, \langle r_k, ns_k\rangle\}$*, such that* $r_i = r_j \Leftrightarrow i = j$*. We will let the meta-variables* $cs, cs_1, cs_2, \ldots$ *range over cablesets.*

Definition 4 *Every cableset is a SNePS* **node.** *Every SNePS*$_P$ **node** *is either a base node or a cableset*[4]*.*

Definition 5 *A* **molecular node** *is a cableset.*

Definition 6 *We will overload the* **membership** *relation "*$\in$*" so that* $x \in s$ *holds just under the following conditions:*

- *If* x *is any object and* s *is a set of such objects, then* $\in$ *has its usual meaning. (Note that this situation obtains if* x *is a cable and* s *is a cableset.)*
- *If* x *is a wire,* $\langle r_1, n\rangle$*, and* s *is a cable,* $\langle r_2, ns\rangle$*, then* $x \in s \Leftrightarrow r_1 = r_2 \wedge n \in ns$*.*
- *If* x *is a wire and* s *is a cableset, then* $x \in s \Leftrightarrow \exists(c)[c \in s \wedge x \in c]$*.*
- *If* x *is a wire or a cable and* s *is a base node, then* $x \notin s$*.*

Definition 7 *An* **nrn-path** *from the node* n_1 *to the node* n_{k+1} *is a sequence,* $n_1, r_1, \ldots, n_k, r_k, n_{k+1}$ *where the* n_i *are nodes, the* r_i *are SNePS Relations, and for each* i*,* $\langle r_i, n_{i+1}\rangle$ *is a wire in* n_i*. We say that the nrn-path* $n_1, r_1, \ldots, n_k, r_k, n_{k+1}$ *goes* through $n_i, 1 \leq i \leq k$*.*

[4] Full SNePS also contains variable nodes.

Definition 8 *A node n_1* **dominates** *a node n_2 just in case there is an nrn-path from n_1 to n_2.*

The use of *sets* of cables and *sets* of nodes is significant, *e.g.,*

$$\{\langle r_1, \{n_1, n_2\}\rangle, \langle r_2, \{n_3, n_4\}\rangle\} = \{\langle r_2, \{n_4, n_3\}\rangle, \langle r_1, \{n_2, n_1\}\rangle\}.$$

However, a node and a proper subset of it are different nodes, and if two nodes differ only in that one contains the cable $\langle r, ns_1\rangle$ while the other contains the cable $\langle r, ns_2\rangle$ and the sets ns_1 and ns_2 are different, then the two nodes are non-identical nodes. Notice that this means that it makes no sense to add a new arc emanating from an existing node (*i.e.*, a new wire to a node, while having it remain the same node). Also notice that a node is determined by the arcs emanating from it, not by the arcs pointing into it.

8.2.1 Examples

`M1!` in Figure 1 is the cableset $\{\langle \texttt{REL}, \{\texttt{BROTHERS}\}\rangle, \langle \texttt{ARGUMENT}, \{\texttt{BOB}, \texttt{HARRY}, \texttt{JOE}\}\rangle\}$, one of whose wires is $\langle \texttt{ARGUMENT}, \texttt{HARRY}\rangle$.

In Figure 2, `M1!` is the cableset $\{\langle \texttt{MEMBER}, \{\texttt{ROVER}\}\rangle, \langle \texttt{CLASS}, \{\texttt{DOG}\}\rangle\}$, and `M2!` is the cableset $\{\langle \texttt{SUBCLASS}, \{\texttt{DOG}\}\rangle, \langle \texttt{SUPERCLASS}, \{\texttt{ANIMAL}\}\rangle\}$.

$\texttt{M1}_1$, $\texttt{M1}_2$ and $\texttt{M1}_3$ of Section 2 are the cablesets shown in the following table:

NodeName	*Cableset*
$\texttt{M1}_1$	$\{\langle \texttt{MEMBER}, \{\texttt{ROVER}\}\rangle, \langle \texttt{CLASS}, \{\texttt{DOG}\}\rangle\}$
$\texttt{M1}_2$	$\{\langle \texttt{MEMBER}, \{\texttt{ROVER}\}\rangle, \langle \texttt{CLASS}, \{\texttt{ANIMAL}\}\rangle\}$
$\texttt{M1}_3$	$\{\langle \texttt{MEMBER}, \{\texttt{ROVER}\}\rangle, \langle \texttt{CLASS}, \{\texttt{DOG}, \texttt{ANIMAL}\}\rangle\}$

It is, therefore, clear that they are different nodes, and represent different entities.

8.3 Reduction

The relation between a node and a proper subset of it is captured by the *Reduce* meta-predicate:

Reduce(cs_1, cs_2) means that the set of wires in cs_2 is a subset of the set of wires in cs_1. If *Reduce*(cs_1, cs_2), we will say that cs_2 is a *reduction* of cs_1.

This definition is formalized in:

Axiom 3 $Reduce(cs_1, cs_2) \Leftrightarrow \forall(w)[w \in cs_2 \Rightarrow w \in cs_1]$

Recall that a node, even a cableset, may not actually be built in the SNePS network, so it is possible that there is some node n for which $Conceive(n)$ is false. However, a cableset cannot be in the network unless every node it dominates is in the network.

Axiom 4 $\langle r, n\rangle \in cs \wedge Conceive(cs) \Rightarrow Conceive(n)$

Molecular nodes may represent either individuals or propositions. Which one a given node represents depends on, and is determined by, the set of relations in the node.[5] Propositional molecular nodes roughly correspond to formulae in standard Predicate Logic, while individual molecular nodes (which we sometimes call "structured individuals") roughly correspond to functional terms. (Since propositional molecular nodes are also terms, they both roughly correspond to functional terms.) Like their counterparts, nodes get their semantics from the user—the person who designs a particular SNePS agent. The semantics also depends on the set of relations in the node, which, therefore roughly corresponds to a predicate or function. The Uniqueness Principle for molecular nodes is enforced in virtue of the fact that different nodes *are* different nodes and represent different entities.

As examples we will use in the rest of this paper, in SNePS/CASSIE [17], `member, class, subclass,` and `superclass` are SNePS Relations, and the semantics given for SNePS/CASSIE nodes include[6] (paraphrased):

- a node of the form $\{\langle \texttt{member}, \{i_1\}\rangle, \langle \texttt{class}, \{i_2\}\rangle\}$ represents the proposition that the entity $[\![i_1]\!]$ is a member of the class $[\![i_2]\!]$.
- a node of the form $\{\langle \texttt{subclass}, \{i_3\}\rangle, \langle \texttt{superclass}, \{i_4\}\rangle\}$ represents the proposition that the class $[\![i_3]\!]$ is a subclass of the class $[\![i_4]\!]$.

[5] SNePS, as currently implemented, does not actually type nodes as representing individuals or propositions, but a node can be so characterized, as stated, as long as the user supplies a consistent semantics to various sets of relations.

[6] This representation of classification hierarchies is simplistic, but will serve the purposes of the present paper. For a more sophisticated representation of classification hierarchies, see [10].

9 Path-Based Inference and Virtual Belief

Although different nodes represent different entities, an asserted node may give rise to several beliefs depending on the rest of the network it is connected with.

Informally, *path-based* inference [11, 18] is a means of inferring a virtual arc from a node n to a node m when there is a certain path from n to m.

For example, using the relations mentioned above, we may specify the inheritance of class membership with the SNePS User Language (SNePSUL) command,

```
(define-path class (compose class (kstar (compose subclass- ! superclass))))
```

Informally, this says that a virtual `class` arc may be inferred from a node n to a node m whenever a path of arcs consisting of a `class` arc, followed by zero or more occurrences of the path consisting of a `subclass` arc (followed backwards) followed by a `superclass` arc goes from n to m, as long as each `superclass` arc emanates from an asserted node (one representing a believed proposition). There are twelve path formation operators like `compose` and `kstar` in SNePSUL including `converse, kplus, or,` and `and`. Path-based reasoning was described in [11] as being a kind of "subconscious" reasoning. This is captured in the formalization of path-based reasoning which follows.

In the remainder of this section, meta-variables: r will range over SNePS relations; w will range over wires; p will range over paths; $m, m_1, m_2, \ldots$ will range over propositional molecular nodes; $n, n_1, n_2, \ldots$ will range over nodes. Additional meta-predicates we will need are:

$Pbr(r, p)$ means that the path based inference rule `(define-path` r p`)` has been entered into the system.

$HavePath(m, p, n)$ means that the path p is in the network going from m to n, both of which are built in the network. See Appendix A for a formal definition of the syntax of paths and a formal, inductive definition of $HavePath$.

$Vbelieve(m)$ means that the agent acts as if it believes $[\![m]\!]$, although $Conceive(m)$ is not necessarily true. $Vbelieve$ (for *Virtual belief*) is a kind of subconscious belief that captures the notion of the agent's believing a proposition $[\![p]\!]$ even though p is not constructed in the network.[7]

[7] $Vbelieve$ is a kind of implicit belief, but it is not as powerful as Levesque's implicit belief predicate L [4]. $L\alpha$ is true whenever α logically follows from the agent's explicit or implicit beliefs, but as will be seen, $Vbelieve(m)$ is true only when m follows from explicitly believed propositions, conceived of entities, and explicitly entered path-based inference rules.

Definition 9 *We will extend the notion of* $\cup$ *so that for a node cs and a wire w,* $cs \cup w$ *will be the node that contains all wires that cs contains, plus w also.*

The following axioms specify when $Vbelieve(m)$ holds:

First, a proposition that is believed is also subconsciously believed.

Axiom 5 $Believe(m) \Rightarrow Vbelieve(m)$

Second, if a virtual r arc may be inferred as going from a node m, denoting a subconsciously believed proposition, to a node n from a path-based inference rule entered into the system, then the proposition $[\![(m \cup \langle r, n\rangle)]\!]$ is subconsciously believed, even though its node is not necessarily in the network.

Axiom 6 $Vbelieve(m) \wedge Pbr(r,p) \wedge HavePath(m,p,n) \Rightarrow Vbelieve(m \cup \langle r, n\rangle)$

Third, the agent subconsciously believes propositions denoted by reductions of nodes that denote subconsciously believed nodes.

Axiom 7 $VBelieve(m_1) \wedge Reduce(m_1, m_2) \Rightarrow Vbelieve(m_2)$

A subconscious belief in some proposition can lead to a conscious belief in the proposition if the agent conceives of the proposition:

Axiom 8 $Vbelieve(m) \wedge Conceive(m) \Rightarrow Believe(m)$

Let $Pbclosure(n, m)$ mean that n contains all the wires in m and all the virtual wires that can be inferred to be in m by virtue of path-based inference rules:

Axiom 9

$$\begin{aligned} Pbclosure(m_1, m_2) \quad \Leftrightarrow \quad & Reduce(m_1, m_2) \\ & \wedge \forall (r,p)[Pbr(r,p) \wedge HavePath(m_2, p, m_3) \Rightarrow \langle r, m_3\rangle \in m_1] \\ & \wedge \forall (w)[w \in m_1 \Rightarrow w \in m_2 \vee \exists (r,p)[Pbr(r,p) \wedge HavePath(m_2, p, m_3) \wedge w = \langle r, m_3\rangle]\!] \end{aligned}$$

If the agent believes (at least subconsciously) a proposition, it will subconsciously believe the proposition represented by the Pbclosure of the node that represents that proposition.

Lemma 1 $Vbelieve(m_1) \wedge Pbclosure(m_2, m_1) \Rightarrow Vbelieve(m_2)$

Proof: Follows by induction from Axioms 6 and 9.

If the agent conceives of a proposition represented by a reduction of the Pbclosure of an asserted node, the agent will believe that proposition:

Theorem 1 $Believe(m_1) \wedge Pbclosure(m_2, m_1) \wedge Reduce(m_2, m_3) \wedge Conceive(m_3) \Rightarrow Believe(m_3)$

Proof: Follows from Axiom 5, Lemma 1, Axiom 7, and Axiom 8.

This theorem captures the notion of subconscious reasoning in SNePS. Propositions that are derived on the basis of reduction and path-based inference are essentially "already" represented in the network "embedded" in the nodes that have been explicitly built and asserted. This subconscious reasoning contrasts with the "conscious" reasoning performed on the basis of node-based inference rules[11]. For an up-to-date presentation of node-based inference in SNePS, see [6].

As an example of subconscious reasoning, assume again the SNePS Relations `member, class, subclass,` and `superclass`, and the path-based inference rule shown above. Then,

$$\{\langle \texttt{member}, \{\texttt{rover}, \texttt{snoopy}\}\rangle, \langle \texttt{class}, \{\texttt{dog}, \texttt{male}\}\rangle\}$$

represents the proposition that $[\![\texttt{rover}]\!]$ and $[\![\texttt{snoopy}]\!]$ are $[\![\texttt{dog}]\!]$s and $[\![\texttt{male}]\!]$s, and $\{\langle \texttt{subclass}, \{\texttt{dog}\}\rangle, \langle \texttt{superclass}, \{\texttt{animal}\}\rangle\}$ represents the proposition that $[\![\texttt{dog}]\!]$s are $[\![\texttt{animal}]\!]$s. In that case, belief in the two propositions:

$$[\![\{\langle \texttt{member}, \{\texttt{rover}, \texttt{snoopy}\}\rangle, \langle \texttt{class}, \{\texttt{dog}, \texttt{male}\}\rangle\}]\!]$$
$$[\![\{\langle \texttt{subclass}, \{\texttt{dog}\}\rangle, \langle \texttt{superclass}, \{\texttt{animal}\}\rangle\}]\!]$$

entails belief in any of the following (different) propositions that the agent conceives of:

$$[\![\{\langle \texttt{member}, \{\texttt{rover}\}\rangle, \langle \texttt{class}, \{\texttt{dog}\}\rangle\}]\!],$$
$$[\![\{\langle \texttt{member}, \{\texttt{rover}\}\rangle, \langle \texttt{class}, \{\texttt{male}\}\rangle\}]\!],$$
$$[\![\{\langle \texttt{member}, \{\texttt{rover}\}\rangle, \langle \texttt{class}, \{\texttt{animal}\}\rangle\}]\!],$$
$$[\![\{\langle \texttt{member}, \{\texttt{snoopy}\}\rangle, \langle \texttt{class}, \{\texttt{dog}\}\rangle\}]\!],$$
$$[\![\{\langle \texttt{member}, \{\texttt{snoopy}\}\rangle, \langle \texttt{class}, \{\texttt{male}\}\rangle\}]\!],$$

$$[\![\{\langle \texttt{member}, \{\texttt{snoopy}\}\rangle, \langle \texttt{class}, \{\texttt{animal}\}\rangle\}]\!].$$

Example Run

The following is the output of an interaction with SNePS 2.0, edited only to eliminate extra blank lines and the list of nodes returned by the `describe` command, and to add comments (in italics). The SNePSUL prompt is "`*`". `build` is the command to construct a node in the network, and thereby to make the agent conceive of the entity represented by the built node. `assert` builds a node and makes it asserted, thereby causing the agent to believe the proposition represented by the node. `describe` is a command to print a Lisp-like description of a node, so the reader can see its cableset. Symbols of the form `M`*n*, where *n* is an integer, are the names of the nodes. SNePS prints the names of asserted nodes with "`!`" appended, and does not append "`!`" to the names of unasserted nodes. The fact that a previously unbuilt node is asserted as soon as it is built shows that it was already Vbelieved before it was Conceived of.

```
*(define member class subclass superclass) declare the relations to be used.
(MEMBER CLASS SUBCLASS SUPERCLASS)
CPU time :  0.25

*(define-path class (compose class (kstar (compose subclass- !  superclass))))
CLASS implied by the path (COMPOSE CLASS (KSTAR (COMPOSE SUBCLASS- !  SUPERCLASS)))
CLASS- implied by the path (COMPOSE (KSTAR (COMPOSE SUPERCLASS- !  SUBCLASS)) CLASS-)
CPU time :  0.30

*(describe (build subclass dog superclass animal))
(M1 (SUBCLASS DOG) (SUPERCLASS ANIMAL)) ; M1 is built, but not asserted.
CPU time :  0.10

*(describe (assert superclass animal subclass dog)) ; order of cables doesn't matter.
(M1!  (SUBCLASS DOG) (SUPERCLASS ANIMAL)) ; This is M1 again, now asserted.
CPU time :  0.10

*(describe (assert member (rover snoopy) class (dog male)))
(M2!  (CLASS DOG MALE) (MEMBER ROVER SNOOPY)) ; built and asserted.
CPU time :  0.12

*(describe (build member rover class dog))
(M3!  (CLASS DOG) (MEMBER ROVER)) ; A restriction of M2!, therefore asserted
CPU time :  0.05

*(describe (build member rover class male))
(M4!  (CLASS MALE) (MEMBER ROVER)) ; A restriction of M2!, therefore asserted
CPU time :  0.07

*(describe (build member rover class animal))
(M5!  (CLASS ANIMAL) (MEMBER ROVER)) ; restriction of Pbclosure of M2!, therefore asserted
CPU time :  0.07
```

```
*(describe (build member snoopy class dog))
(M6!  (CLASS DOG) (MEMBER SNOOPY))
CPU time :  0.10

*(describe (build member snoopy class male))
(M7!  (CLASS MALE) (MEMBER SNOOPY))
CPU time :  0.08

*(describe (build member snoopy class animal))
(M8!  (CLASS ANIMAL) (MEMBER SNOOPY))
CPU time :  0.08
```

10 Concluding Remarks

The definition of SNePS molecular nodes as cablesets captures the notion that a new arc (wire) cannot be added as emanating from an already existing node. It makes it clear that that would amount to changing the denotation of the node. Instead, a new wire joined to an old node makes a new node that is related to the old one by the reduction relation. Similarly, if one contemplates a node without one or more of its wires, one is contemplating a new node that is a reduction of the old one. The propositions denoted by a node and a reduction of it are related by reduction inference, which is one kind of "subconscious" inference supported by SNePS. Path-based inference is another kind of "subconscious" inference that justifies belief in a proposition when a reduction is already believed and the "extra" wires can be inferred from path-based inference rules and paths in the network. The set of propositions subconsciously believed by the SNePS agent is the set denoted by the set of nodes that could be gotten by path-based-closure of asserted nodes followed by reduction. These nodes are "virtually" or "implicitly" in the net, and need be made explicit only when there is a specific reason (such as the user asks about one, or explicitly builds one).

Although analogues of reduction inference and path-based inference could be defined on knowledge representation formalisms other than propositional semantic networks, they most naturally arose from, and are most easily understood in terms of these networks.

11 Acknowledgements

The development of the theory and implementation of SNePS has been carried out over the years with the help of the members of SNeRG, the SNePS Research Group of SUNY at Buffalo. Their collaboration is gratefully acknowledged. I am also grateful for comments on earlier versions of this paper made by

SNeRG members, and by the other participants of the Workshop on Formal Aspects of Semantic Networks, particularly Robert Wilensky.

This work was supported in part by the National Science Foundation under Grant IRI-8610517, and in part by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, New York 13441-5700, and the Air Force Office of Scientific Research, Bolling AFB DC 20332 under Contract No. F30602-85-C-0008, which supported the Northeast Artificial Intelligence Consortium (NAIC).

References

[1] R. J. Brachman and H. J. Levesque, editors. *Readings in Knowledge Representation*. Morgan Kaufmann, San Mateo, CA, 1985.

[2] R. Fagin and J. Y. Halpern. Belief, awareness, and limited reasoning. *Artificial Intelligence*, 34(1):39–76, December 1987.

[3] J. Hintikka. Impossible possible worlds vindicated. *Journal of Philosophical Logic*, 4:475–484, 1975.

[4] H. J. Levesque. A logic of implicit and explicit belief. In *Proceedings of the National Conference on Artificial Intelligence*, pages 198–202. Morgan Kaufmann, San Mateo, CA, 1984.

[5] A. S. Maida and S. C. Shapiro. Intensional concepts in propositional semantic networks. *Cognitive Science*, 6(4):291–330, October–December 1982. Reprinted in [1, pp. 170–189].

[6] J. P. Martins and S. C. Shapiro. A model for belief revision. *Artificial Intelligence*, 35:25–79, 1988.

[7] Merriam-Webster. *Webster's New Collegiate Dictionary*. G. & C. Merriam Co., Springfield, MA, 1961.

[8] E. J. M. Morgado. *Semantic Networks as Abstract Data Types*. PhD thesis, Technical Report 86–19, Department of Computer Science, SUNY at Buffalo, Buffalo, NY, 1986.

[9] E. J. M. Morgado and S. C. Shapiro. Believing and acting: A study of meta-knowledge and meta-reasoning. In *Proceedings of EPIA-85 "Encontro Portugues de Inteligencia Artificial"*, pages 138–154, Oporto, Portugal, 1985.

[10] S. L. Peters and S. C. Shapiro. A representation for natural category systems. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 140–146. Morgan Kaufmann, San Mateo, CA, 1987.

[11] S. C. Shapiro. Path-based and node-based inference in semantic networks. In D. L. Waltz, editor, *Tinlap-2: Theoretical Issues in Natural Languages Processing*, pages 219–225. ACM, New York, 1978.

[12] S. C. Shapiro. The SNePS semantic network processing system. In N. V. Findler, editor, *Associative Networks: The Representation and Use of Knowledge by Computers*, pages 179–203. Academic Press, New York, 1979.

[13] S. C. Shapiro. Symmetric relations, intensional individuals, and variable binding. *Proceedings of the IEEE*, 74(10):1354–1363, October 1986.

[14] S. C. Shapiro. Processing, bottom-up and top-down. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 779–785. John Wiley & Sons, New York, 1987. Reprinted in Second Edition, 1992, pages 1229–1234.

[15] S. C. Shapiro. Representing plans and acts. In *Proceedings of the Third Annual Workshop on Conceptual Graphs*, pages 3.2.7–1–3.2.7–6, Menlo Park, CA, 1988. AAAI.

[16] S. C. Shapiro, D. Kumar, and S. Ali. A propositional network approach to plans and plan recognition. In A. Maier, editor, *Proceedings of the 1988 Workshop on Plan Recognition*. Morgan Kaufmann, San Mateo, CA, 1990.

[17] S. C. Shapiro and W. J. Rapaport. SNePS considered as a fully intensional propositional semantic network. In N. Cercone and G. McCalla, editors, *The Knowledge Frontier*, pages 263–315. Springer-Verlag, New York, 1987.

[18] R. Srihari. Combining path-based and node-based reasoning in SNePS. Technical Report 183, Department of Computer Science, SUNY at Buffalo, Buffalo, NY, 1981.

[19] G. Summers. *Test Your Logic*. Dover, New York, 1972.

A Formal Syntax of Paths

Below is a formal syntax of all the paths available in SNePS, and a formal, inductive definition of the $HavePathL$ meta-predicate. In the axioms for $HavePathL$, meta-variables: r will range over SNePS relations; w will range over wires; $p, p_1, p_2, \ldots$ will range over paths; $m, m_1, m_2, \ldots$ will range over cablesets; $n, n_1, n_2, \ldots$ will range over nodes; $i, i_1, i_2, \ldots$ will range over non-negative integers. $HavePathL$ is related to $HavePath$ as shown by the following axiom:

Axiom 10 $HavePath(m, p, n) \Leftrightarrow \exists(i) HavePathL(m, p, n, i)$

The syntax of paths with the axioms for $HavePathL$ are:

unitpath ::= *relation*

If the wire $\langle r, n \rangle$ is in the cableset m, then r is a unitpath from m to n.

unitpath ::= *relation*-

If the wire $\langle r, n \rangle$ is in the cableset m, then $r-$ is a unitpath from n to m.

path ::= *unitpath*

$\langle r, n \rangle \in m \wedge Conceive(m) \Leftrightarrow HavePathL(m, r, n, 1)$

$\langle r, n \rangle \in m \wedge Conceive(m) \Leftrightarrow HavePathL(n, r-, m, 1)$

path ::= (`converse` *path*)

$HavePathL(m, p, n, i) \Leftrightarrow HavePathL(n, (\texttt{converse}\, p), m, i)$

$path ::= \texttt{(compose}\ path\ [!]\ path^*\texttt{)}$

$\exists(m_2)[HavePathL(m_1, p_1, m_2, i_1) \wedge HavePathL(m_2, p_2, m_3, i_2)]$

$\Leftrightarrow HavePathL(m_1, (\texttt{compose}\, p_1\, p_2), m_3, i_1 + i_2)$

$\exists(m_2)[HavePathL(m_1, p_1, m_2, i_1) \wedge Believe(m_2) \wedge HavePathL(m_2, p_2, m_3, i_2)]$

$\Leftrightarrow HavePathL(m_1, (\texttt{compose}\, p_1\, !\, p_2), m_3, i_1 + i_2)$

$\exists(m_2)[HavePathL(m_1, p_1, m_2, i_1) \wedge HavePathL(m_2, (\texttt{compose}\, p_2 \ldots p_k), m_3, i_2)]$

$\Leftrightarrow HavePathL(m_1, (\texttt{compose}\, p_1\, p_2 \ldots p_k), m_3, i_1 + i_2)$

$\exists(m_2)[HavePathL(m_1, p_1, m_2, i_1) \wedge Believe(m_2) \wedge HavePathL(m_2, (\texttt{compose}\, p_2 \ldots p_k), m_3, i_2)]$

$\Leftrightarrow HavePathL(m_1, (\texttt{compose}\, p_1\, !\, p_2 \ldots p_k), m_3, i_1 + i_2)$

$path ::= \texttt{(kstar}\ path\texttt{)}$

$HavePathL(m_1, (\texttt{kstar}\, p), m_1, 0)$

$\exists(m_2)[HavePathL(m_1, p, m_2, i_1) \wedge HavePathL(m_2, (\texttt{kstar}\, p), m_3, i_2)]$

$\Leftrightarrow HavePathL(m_1, (\texttt{kstar}\, p), m_3, i_1 + i_2)$

$path ::= \texttt{(kplus}\ path\texttt{)}$

$HavePathL(m_1, p, m_2, i) \Leftrightarrow HavePathL(m_1, (\texttt{kplus}\, p), m_2, i)$

$\exists(m_2)[HavePathL(m_1, p, m_2, i_1) \wedge HavePathL(m_2, (\texttt{kplus}\, p), m_3, i_2)]$

$\Leftrightarrow HavePathL(m_1, (\texttt{kplus}\, p), m_3, i_1 + i_2)$

$path ::= \texttt{(or}\ \{path\}^*\texttt{)}$

$HavePathL(m_1, p_1, m_2, i_1) \vee HavePathL(m_1, p_2, m_2, i_2) \Leftrightarrow HavePathL(m_1, (\texttt{or}\, p_1\, p_2), m_2, \min(i_1, i_2))$

$HavePathL(m_1, p_1, m_2, i_1) \vee HavePathL(m_1, (\texttt{or}\, p_2 \ldots p_k), m_2, i_2)]$

$\Leftrightarrow HavePathL(m_1, (\texttt{or}\, p_1\, p_2 \ldots p_k), m_2, \min(i_1, i_2))$

$path ::= \texttt{(and}\ \{path\}^*\texttt{)}$

$HavePathL(m_1, p_1, m_2, i_1) \wedge HavePathL(m_1, p_2, m_2, i_2) \Leftrightarrow HavePathL(m_1, (\texttt{and}\, p_1\, p_2), m_2, \max(i_1, i_2))$

$HavePathL(m_1, p_1, m_2, i_1) \wedge HavePathL(m_1, (\texttt{and}\, p_2 \ldots p_k), m_2, i_2)]$

$\Leftrightarrow HavePathL(m_1, (\texttt{and}\, p_1\, p_2 \ldots p_k), m_2, \max(i_1, i_2))$

$path ::= \texttt{(not}\ path\texttt{)}$

$\neg HavePath(m_1, p, m_2) \Leftrightarrow HavePathL(m_1, (\texttt{not}\, p), m_2, 0)$

$path$ ::= (**relative-complement** $path$ $path$)

$HavePathL(m_1, p_1, m_2, i) \wedge \neg HavePath(m_1, p_2, m_2)$

$\Leftrightarrow HavePathL(m_1, (\texttt{relative} - \texttt{complement}\, p_1\, p_2), m_2, i)$

$path$ ::= (**irreflexive-restrict** $path$)

$HavePathL(m_1, p, m_2, i) \wedge m_1 \neq m_2 \Leftrightarrow HavePathL(m_1, (\texttt{irreflexive} - \texttt{restrict}\, p), m_2, i)$

$path$ ::= (**domain-restrict** ($path$ $node$) $path$)

$HavePath(m_1, p_1, m_2) \wedge HavePathL(m_1, p_2, m_3, i)$

$\Leftrightarrow HavePathL(m_1, (\texttt{domain} - \texttt{restrict}(p_1\ m_2)p_2), m_3, i)$

$path$ ::= (**range-restrict** $path$ ($path$ $node$))

$HavePathL(m_1, p_1, m_2, i) \wedge HavePath(m_2, p_2, m_3)$

$\Leftrightarrow HavePathL(m_1, (\texttt{range} - \texttt{restrict}\, p_1\ (p_2\ m_3)), m_2, i)$

$path$ ::= (**exception** $path$ $path$)

$HavePathL(m_1, p_1, m_2, i_1) \wedge \neg\exists(i_2)[i_2 \leq i_1 \wedge HavePathL(m_1, p_2, m_2, i_2)]$

$\Leftrightarrow HavePathL(m_1, (\texttt{exception}\, p_1\, p_2), m_2, i_1)$