KNOWLEDGE REPRESENTATION

João Pavão Martins

Departamento de Engenharia Informática
Instituto Superior Técnico
Universidade Técnica de Lisboa

João Pavão Martins
Grupo de Inteligência Artificial
Departamento de Engenharia Informática
Instituto Superior Técnico
Av. Rovisco Pais
1096 Lisboa CODEX
Portugal
jpm@gia.ist.utl.pt

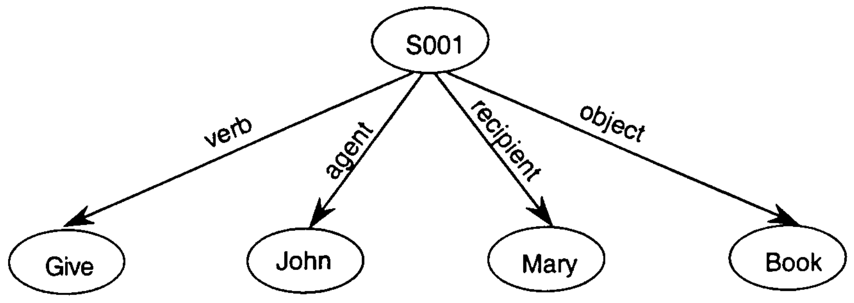

Figure 6.15: Alternative representation for "John gave Mary a book".

from the other arcs:

1. The arcs we had before represent statements. Their presence in the networks corresponds to a proposition about the two nodes that they connect. These are called *assertional* arcs, since they make assertions about the nodes that they relate;

2. New arcs represent "parts" of a proposition. For example, the meaning of S001 is given by *all* arcs that emanate from it, not only by a subset of these arcs. These new arcs are called *structural.* We can argue that these arcs represent the statement that the agent of S001 is John, the recipient of S001 is Mary, and so on. In this representation S001 has no meaning other that what is ascribed by the structural links to other nodes.

One of the important aspects raised by [Woods 75], and later on by [Woods 91], is that we must be careful in making interpretations on the basis of a clear understanding of the semantics of the notation ant not on the basis of the informal semantics implied by the names of the nodes and arcs.

6.3 SNePS: A case study

In this section, we consider, in detail, a propositional semantic network, SNePS ("Semantic Network Processing System"), developed by Stuart C. Shapiro and his research group. The goal of building SNePS was to produce a network that would be able to represent virtually everything that is used in natural language. For this reason, SNePS offers a high expressive power. SNePS has undergone several versions, the most important of which are SNePS 1.0 [Shapiro 79a], [Shapiro and Rapaport 87] and SNePS 2.1 [Shapiro and Martins 90]. A detailed description of the historical evolution of SNePS is presented in [Shapiro and Rapaport 92].

SNePS uses structural relations, as described at the end of the last section. SNePS is called a *propositional* semantic network because each proposition is represented by a node and not by an arc. In SNePS, nodes represent concepts, and arcs represent *non-conceptual* relations[2] between concepts. Arcs are only used to structure concepts. The fact that concepts are represented by nodes means that, in SNePS, we can only talk about the information represented by nodes and we cannot refer to the information represented by arcs. Arcs are considered to be part of the syntactic structure of the node from where they emanate.

SNePS imposes certain restrictions on the representation it uses. It is important to understand these restrictions, because they distinguish SNePS from a general labeled, directed graph, and from many other semantic network formalisms.

6.3.1 Nodes in SNePS

In SNePS, nodes represent concepts. There are two important aspects regarding the concepts represented in SNePS:

1. Nodes represent intensional concepts;
2. Each concept is represented by a unique node and each node represents a unique concept. This is known as the *uniqueness principle* [Maida and Shapiro 82].

The notion of a concept is vague and it corresponds to anything that we can talk about. Concepts, thus, represent, among other things, propositions. The benefit of representing propositions by nodes is that propositions about propositions can be represented with no limits.

[2]That is, information that does not correspond to a concept in the network.

If there is a direct arc[3] from node m to node n, it is said that m *immediately dominates* n. If there is a path of direct arcs from node m to node n, it is said that m *dominates* n. There are no cycles composed only by direct arcs, so no node dominates itself. If there is a direct arc, r, from node m to node n, it is said that n "is the value of" $r(m)$.

According to their position in the network, nodes can be atomic or structured.

1. A node is said to be *atomic*, if it dominates no other node. Atomic nodes can be further divided into constant and variable nodes:

   (a) Atomic constant nodes, also called *base nodes*, represent non-structured concepts, and correspond, in a sense, to the constants of first-order logic. Atomic constant nodes are represented, in our notation,[4] by a rectangle, inside of which there is the name of the node. These nodes are labeled by a string that stands for the name of an individual concept, for the name of a relation, for the name of a property, and so on.

   (b) Atomic non-constant nodes, also called *variable nodes*, represent variables, and correspond, in some sense, to the variables in first-order logic.[5] Variable nodes are represented by a circle, inside of which there is the name of the node, labeled in SNePS by an identifier of the form Vn, where n is a positive integer.

2. A node is said to be *structured* if it dominates other node(s) (structured nodes are represented by a circle, inside of which there is the label of the node). Structured nodes can be further divided into constant and variable nodes:

   (a) Constant structured nodes represent either propositions or structured concepts. These nodes correspond either to well formed formulas in first-order logic without free variables or to complex terms. Constant structured nodes are labeled in SNePS by an identifier of the form Mn, where n is a positive integer. This identifier may be followed by the character "!", meaning that the

[3]Direct arcs will be defined in the next section. For the purpose of the description that we present here, we can think of direct arcs as the arcs that appear in the graphical descriptions of the networks.

[4]This is not a standard notation within SNePS.

[5]These may also represent arbitrary propositions.

corresponding node is believed by SNePS. In this case, we say that the node is *asserted* in the network.

(b) Non-constant structured nodes, also called *pattern nodes*, dominate at least a node corresponding to a free variable. These nodes correspond to well formed formulas in first-order logic with free variables. Non-constant structured nodes are labeled in SNePS by an identifier of the form Pn, where n is a positive integer.

6.3.2 Arcs in SNePS

In SNePS there are two kinds of arcs, direct and inverse arcs. For each relation represented by a direct arc, there is a converse relation represented by an inverse arc. Inverse arcs have the same name as the corresponding direct arcs, followed by a "-", and will not be shown in our diagrams.

The labels of the arcs used in SNePS are defined by the person who is using it to represent knowledge. However, in order to enable the inference engine to interpret the propositions involving logical connectives, there are some pre-defined arcs in SNePS.

6.3.3 Knowledge representation in SNePS

SNePS has no pre-defined arcs, except for those related with the logical connectives (these are described in Section 6.3.4). Therefore, the first step towards representing knowledge in SNePS corresponds to the choice and definition of the arcs that will be used.

Since arcs in SNePS correspond to structural arcs (in the sense of [Woods 75], as described in Section 6.2), to define a piece of knowledge in SNePS we have to use a "case frame" of arcs that emanate from a node and correspond to the relation or proposition that the node represents. Although there are no fixed guidelines for the definition of these case frames (they depend on the intended application), there are some commonly used representations, some of which are described in this section. For a more complete description, refer to [Shapiro and Rapaport 87] and [Shapiro *et. al.* 93].

Two of the basic aspects that appear in most applications are the statements that an individual is member of a given class and that a class is a subclass of another class. The first statement is usually represented by the case frame

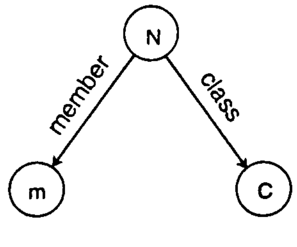

Figure 6.16: Case frame for class membership.

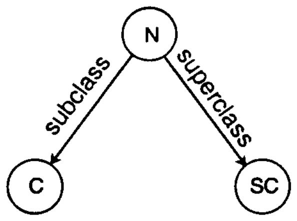

Figure 6.17: Case frame for class containment.

"member/class" (Figure 6.16), where the arc labeled "member" points to the node that corresponds to the concept about which we are asserting membership and the arc labeled "class" points to the node that corresponds to the class. The second statement is usually represented by the case frame "subclass/superclass" (Figure 6.17), where the arc labeled "subclass" points to the node that corresponds to the subclass and the arc labeled "superclass" points to the node that corresponds to the superclass.

Using these case frames, we show in Figure 6.18, the representation that John is a man (node `M1!`) and that the class of men is a subclass of the class of mammals (node `M2!`). It is important to notice that, from this information alone, SNePS is not able to know that John is a mammal because the "operational semantics" associated with these case frames is not part of the network yet. The "operational semantics" can be specified through the use of the connectives described in Section 6.3.4.

It is common to need to explicitly represent arbitrary relations in SNePS. This can be done using the case frame "rel/arg1/.../argn", where the arc labeled "rel" points to the node that corresponds to the relation and the arc labeled "argi" ($1 \leq i \leq n$) points to the node that corresponds to i-th argument of the relation (Figure 6.19). This case frame can be used to represent

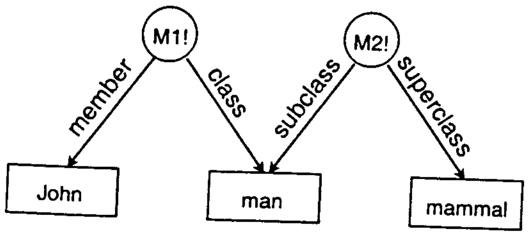

Figure 6.18: John is a man and men are mammals.

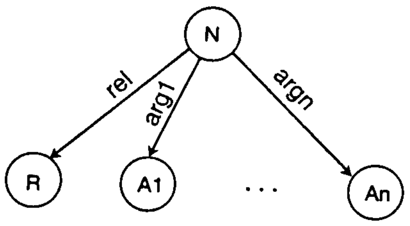

Figure 6.19: Case frame for a n-ary relation.

any relation, namely the membership and subset relations that we discussed before (see Figure 6.20). The representational difference between the use of the case frame of Figure 6.20 and the case frame presented in Figure 6.18 is the fact that using the case frame of Figure 6.20, the membership relation itself becomes a concept in the network, and thus something we can talk about, whereas with the representation of Figure 6.18, the membership relation is implicit in the structure of the arcs and thus cannot be referred by SNePS.

We could also think of the representation shown in Figure 6.21 to represent

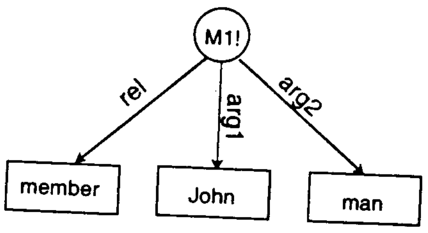

Figure 6.20: Alternative representation for "John is a man".

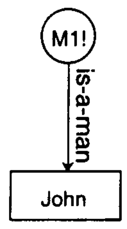

Figure 6.21: Bad representation for "John is a man".

that "John is a man".[6] This representation is a step backwards when compared with the representation of Figure 6.18, since the membership to the class of men is no longer a concept, but rather a relation between two nodes. With the representation of Figure 6.21, we cannot talk about the class of men nor about the relationship that John bears with it.

From this discussion, it may seem that the explicit representation of relations, since it corresponds to a more fine grained representation, should be preferred over the representation of relations through case frames that do not use the explicit concept of relation. You should be aware, however, that a generalized use of the case frame "rel/arg1/.../argn" for the representation of relations has a negative effect on the efficiency of the inference process because the pattern matcher will be overloaded with information (basically, all the nodes that represent relations can match with a generic relation).

The idea behind the "rel/arg1/.../argn" case frame can be used to represent instances of verbs, in the same way as is presented in Figure 6.15. In Figure 6.22, we represent that "John knows that Mary believes that whales are fish" (node `M3!`) and that "Moby Dick is a whale" (node `M4!`). Besides the case frames member/class and subclass/superclass, we use the case frame agent/verb/object to represent instances of verbs. Notice that only nodes M3 and M4 are asserted in the network (their identifier is followed by "!").

The distinction between asserted and non-asserted nodes enables to understand which nodes are believed by SNePS (the asserted nodes) and which nodes are part of the structure of propositions, but not believed by SNePS. The distinction between asserted and non-asserted nodes is made by the user of SNePS and also by the inference system while deducing new information.[7]

[6]I am grateful to António Leitão for suggesting this representation.

[7]In fact, the distinction between asserted and non-asserted nodes is more complex because it resorts to an ATMS-based context shadowing operations.

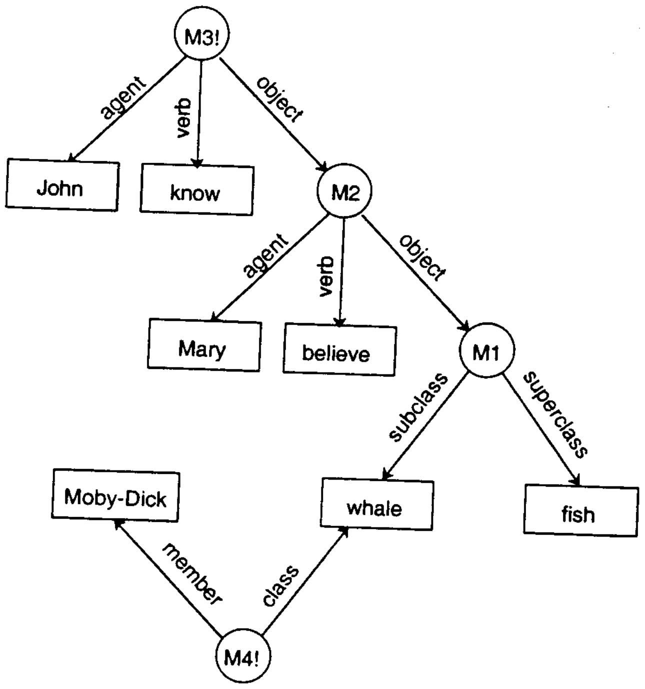

Figure 6.22: Example of representation in SNePS.

With this network, if we ask SNePS whether "whales are fish" it answers "I don't know", but if we ask whether "John knows that Mary believes that whales are fish", it answers "yes".

The meaning of the nodes labeled "John", "Mary", "whale", "fish", "believe", and "Moby Dick" will be defined by the additional network structure that will be connected to them.

6.3.4 Logical connectives in SNePS

An important aspect regarding the research in semantic networks, concerns the study of how to represent relations between propositions. One of the obvious alternatives is to use the traditional logical connectives, which has the advantage of having the guarantee that the inferences produced are solid, well known, and well understood. However, the choice of logical connectives depends on other aspects, such as the ease of representation, the ease of proving properties about the system, the proof of equivalence with other logical systems, and so on. If the number of logical connectives is small, it is easy to prove properties *about* the system and it is difficult to prove properties *in* the system; if the number of logical connectives is large, the opposite holds.

In the choice of the logical connectives to use in SNePS, the main guidelines were an adequate expressive power and the naturalness of the representation. As [Shapiro 79b] argues, "the set of connectives used in traditional first-order logic presents some disadvantages from the network representation point of view. ... the main disadvantage relates to the fact that all the connectives, except negation, are binary and therefore expressing sentences about sets of propositions becomes cumbersome. ... having to choose between $(\alpha \vee \beta) \vee \gamma$ and $\alpha \vee (\beta \vee \gamma)$ is unnecessary and unfairly distinguishes one of the disjuncts".

On the other hand, suppose, for example, that given three propositions, α, β and γ, we wanted to express the fact that exactly one of them is true. Using the traditional logical connectives this would be done as $(((\alpha \wedge \neg\beta) \wedge \neg\gamma) \vee ((\neg\alpha \wedge \beta) \wedge \neg\gamma)) \vee ((\neg\alpha \wedge \neg\beta) \wedge \gamma)$,[8] which is a lengthy and difficult to read wff. Sentences involving more than three propositions (e.g., exactly five out of ten propositions are true) are even more complicated. Since this type of sentence often occurs in some of the intended applications of SNePS,

[8]Again, in each of the conjunctions we had to make the choice of associations because logical connectives are binary.

the need was felt to find a more powerful and simpler way of defining such sentences (for a detailed discussion about the disadvantages of the standard connectives refer to [Shapiro 79b]).

Non-standard connectives

The problems that we just outlined, lead Stuart Shapiro to the definition of non-standard connectives [Shapiro 79b]. All logical connectives in SNePS take sets of propositions (represented by nodes) as arguments. The following connectives are defined in SNePS:

1. *And-or*, represented by $\bigvee\!\!\!\!\bigwedge$. The proposition

   $${}_{n}\bigvee\!\!\!\!\bigwedge_{i}^{j} \{\alpha_1, \ldots, \alpha_n\}$$

   asserts that at least i and at most j of its n arguments (propositions) are true. In this way:

   - ${}_{1}\bigvee\!\!\!\!\bigwedge_{1}^{1}\{\alpha\}$ represents the assertion of α;
   - ${}_{1}\bigvee\!\!\!\!\bigwedge_{0}^{0}\{\alpha\}$ represents the negation of α, $\neg\alpha$;
   - ${}_{2}\bigvee\!\!\!\!\bigwedge_{2}^{2}\{\alpha, \beta\}$ represents the conjunction of α and β, $\alpha \wedge \beta$;
   - ${}_{2}\bigvee\!\!\!\!\bigwedge_{1}^{2}\{\alpha, \beta\}$ represents the disjunction of α and β, $\alpha \vee \beta$.

   We have to consider now to represent a proposition whose main connective is an and-or. In order to do so, we have to address two questions: (1) How the individual propositions that compose the and-or are represented in the proposition that corresponds to the and-or; (2) How do we represent the and-or connective. The first question is answered by the introduction of case frames as defined in Section 6.2. A proposition corresponding to an and-or is represented by a structured node that relates the nodes corresponding to the individual propositions. In SNePS, there are pre-defined arcs that emanate from the node that corresponds to the and-or, and whose semantics is known by the inference engine. The second question is answered by the kinds of arcs that emanate from this structured node: arcs labeled `arg` pointing to the nodes corresponding to the propositions that are the arguments of the and-or, and arcs labeled `min` and `max` pointing to the parameters i and j, respectively. In Figure 6.23 we show the representation of and-or in SNePS. In this figure, a triangle below a node represents the network structure of the proposition that is written below it.

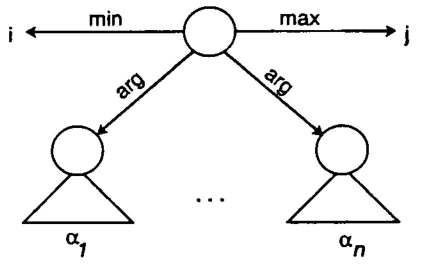

Figure 6.23: Representation of ${}_n\mathbb{X}_i^j\ \{\alpha_1, \ldots, \alpha_n\}$.

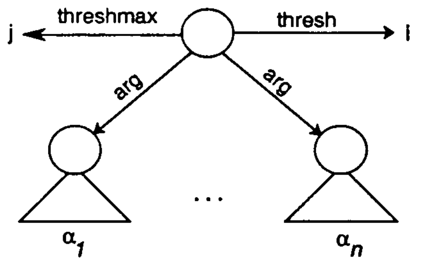

Figure 6.24: Representation of ${}_n\Theta_i\{\alpha_1, \ldots, \alpha_n\}$.

2. *Thresh*, represented by Θ. The proposition

$${}_n\Theta_i^j\{\alpha_1, \ldots, \alpha_n\}$$

   means that either fewer than i or more than j of its n arguments are true. The parameter j may be omitted, in which case, it defaults to $n-1$. If $i = 1$ and j is omitted, ${}_n\Theta_1\{\alpha_1, \ldots, \alpha_n\}$ states that $\alpha_1, \ldots, \alpha_n$ are equivalent.

   Similarly to what we did for and-or, the representation of a proposition whose main connective is thresh is done by a structured node with arcs labeled **arg** pointing to the nodes that correspond to the arguments of the propositio, with an arc labeled **thresh** pointing to the parameter i of the connective, and with an arc labeled **threshmax** pointing to the parameter j of the connective. In Figure 6.24 we show the representation of thresh in SNePS.

3. *And-entailment*, represented by $\wedge\!\Rightarrow$. The proposition

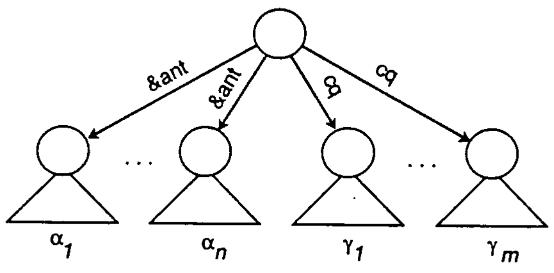

Figure 6.25: Representation of $\{\alpha_1, \alpha_2, \ldots, \alpha_n\} \wedge\!\!\Rightarrow \{\gamma_1, \gamma_2, \ldots, \gamma_m\}$.

$$\{\alpha_1, \alpha_2, \ldots, \alpha_n\} \wedge\!\!\Rightarrow \{\gamma_1, \gamma_2, \ldots, \gamma_m\}$$

means that the conjunction of the antecedents $\alpha_1, \alpha_2, \ldots, \alpha_n$ entails the conjunction of the consequents, $\gamma_1, \gamma_2, \ldots, \gamma_m$.

And-entailment is represented by a node, with arcs labeled **&ant** pointing to the nodes corresponding to the antecedents of the entailment and with arcs labeled cq pointing to the nodes corresponding to the consequents of the entailment (Figure 6.25).

4. *Or-entailment*, represented by $\vee\!\!\Rightarrow$. The proposition

$$\{\alpha_1, \alpha_2, \ldots, \alpha_n\} \vee\!\!\Rightarrow \{\gamma_1, \gamma_2, \ldots, \gamma_m\}$$

states that the disjunction of the antecedents $\alpha_1, \alpha_2, \ldots, \alpha_n$ entails the conjunction of the consequents $\gamma_1, \gamma_2, \ldots, \gamma_m$.

Or-entailment is represented by a node with arcs labeled **ant**, pointing to the nodes corresponding to the antecedents, and arcs labeled cq pointing to the nodes corresponding to the consequents (Figure 6.26).

5. *Numerical entailment*, represented by $i\!\!\Rightarrow$. The proposition

$$\{\alpha_1, \alpha_2, \ldots, \alpha_n\}\ i\!\!\Rightarrow \{\gamma_1, \gamma_2, \ldots, \gamma_m\}$$

states that the conjunction of i antecedents $\alpha_1, \alpha_2, \ldots, \alpha_n$ entails the conjunction of the consequents $\gamma_1, \gamma_2, \ldots, \gamma_m$. Notice that numerical entailment generalizes both and-entailment (if $i = n$) and or-entailment (if $i = 1$), so, in fact, only numerical entailment is needed.

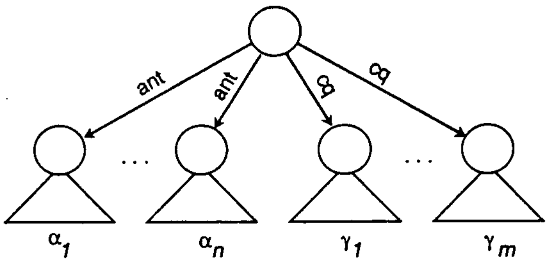

Figure 6.26: Representation of $\{\alpha_1, \alpha_2, \ldots, \alpha_n\} \vee\!\!\Rightarrow \{\gamma_1, \gamma_2, \ldots, \gamma_m\}$.

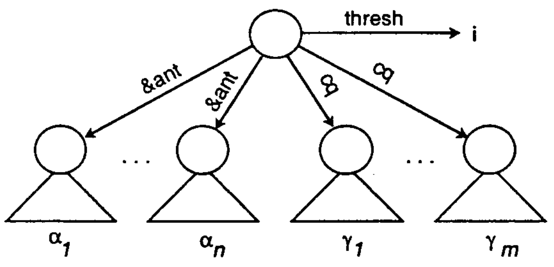

Figure 6.27: Representation of $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}\ i\!\!\Rightarrow \{\gamma_1, \gamma_2, \ldots, \gamma_m\}$.

However, since the most commonly used statements involve assertions corresponding to either or-entailment or and-entailment, these are kept.

Numerical entailment is represented by a node with arcs labeled `&ant` pointing to the nodes corresponding to the antecedents of the entailment, an arc labeled `thresh` pointing to the parameter i, and arcs labeled `cq` pointing to the nodes corresponding to the consequents (Figure 6.27).

Quantifiers in SNePS

There are three quantifiers in SNePS, all of them applicable to a set of variables:

1. *Universal quantifier*, represented by $\forall$. The proposition

$$\forall x[\alpha(x)]$$

means that every constant that can replace x makes the proposition $\alpha(x)$ true.

The universal quantifier is represented in SNePS by an arc labeled `forall` that links the node where the quantification is stated to the nodes that correspond to the quantified variables.

2. The *existential quantifier*, is not implemented in the current version of SNePS. In a previous version, it was represented by $\exists$. The proposition

$$\exists x[\alpha(x)]$$

means that there is at least one constant that can replace x and make the proposition $\alpha(x)$ true. The existential quantifier was represented in SNePS by an arc labeled `exists` that links the node where the quantification is stated to the nodes that correspond to the quantified variables.

To represent the effects of the existential quantifier, the current version of SNePS makes use of Skolem functions: whenever an existentially quantified variable y is bound within the scope of a universally quantified variable x, y can be replaced by the skolem function $f(x)$, provide that f is not used anywhere else. In this case, the existential quantifier that binds y can be eliminated.

3. *Numerical quantifier*, represented by ${}_n\exists_i^j$. The proposition

$${}_n\exists_i^j\ x\ [\{\alpha_1(x), \ldots, \alpha_k(x) : \beta(x)\}]$$

means that there are n constants that when replaced by x make $\alpha_1(x) \wedge \ldots \wedge \alpha_k(x)$ true. However, at least i and at most j of them also satisfy $\beta(x)$.

Numerical quantifier is represented by an arc labeled `nexists` that links the node where the quantification is made to the nodes that correspond to the quantified variables. From the node where the quantification is made, there are also arcs labeled `etot`, `emax` and `emin`, pointing, respectively, to n, i and j. There are also arcs labeled `&ant` pointing to the nodes that correspond to the antecedents ($\alpha_1(x)$, …, $\alpha_k(x)$) and an arc labeled `cq` that points to the consequent ($\beta(x)$).

Numerical quantification enables to represent propositions such as "there are exactly two numbers such that $x^2 + 4 = 4x$"; "there are at least two numbers such that $z + 2 < 6$".

Another advantage of the numerical quantifier is to allow reasoning by exclusion. For example, knowing that every person has exactly one mother and that Betty is John's mother, we may conclude that Mary is not John's mother.

A node that represents a proposition and dominates one or more pattern nodes, may have arcs labeled `forall` or `nexists` pointing to one or more nodes. However, there is the restriction that only one of these types of arcs may emanate from a single node (so that an ordering may be imposed in the use of quantification). Therefore, if more that one quantifier is to be used node embedding is required.

6.3.5 Putting it all together

In this section we discuss the representation in SNePS involving both user-defined case frames and the pre-defined case frames used to represent logical connectives and quantifiers. Every node in SNePS that corresponds to a connective or that contains a quantifier is called a *rule node* or just a *rule*. We will use the case frames described in Section 6.3.3.

We will start by defining rules that specify some of the properties of the membership and subclass relations. The first statement that we represent asserts that if an element is member of a class, then it is also member of a superclass of that class. Using the non-standard connectives, this statement can be expressed as follows:

$$\forall(m, C_1, C_2)\{member(m, C_1), subclass(C_1, C_2)\} \wedge \Rightarrow \{member(m, C_2)\}.$$

The representation of this statement in SNePS is shown in Figure 6.28. This figure makes use of both user-defined case frames (member/class and subclass/superclass) and system-defined case fames to represent the rule. Node `M4!` states that for all `V1`, `V2`, and `V3`, if `V1` is a member of class `V2` *and* `V2` is a subclass of `V3`, *then* `V1` is a member of class `V3`.

If we intend to use the class-subclass relation we should also state that if a class is a subclass of another class, which, in turn, is a subclass of a third class, then the first class is a subclass of the third class. This is a statement of the transitivity relationship that holds between the class-subclass relation. This statement can be expressed as:

$$\forall(C_1, C_2, C_3)\{subclass(C_1, C_2), subclass(C_2, C_3)\} \wedge \Rightarrow \{subclass(C_1, C_3)\}.$$

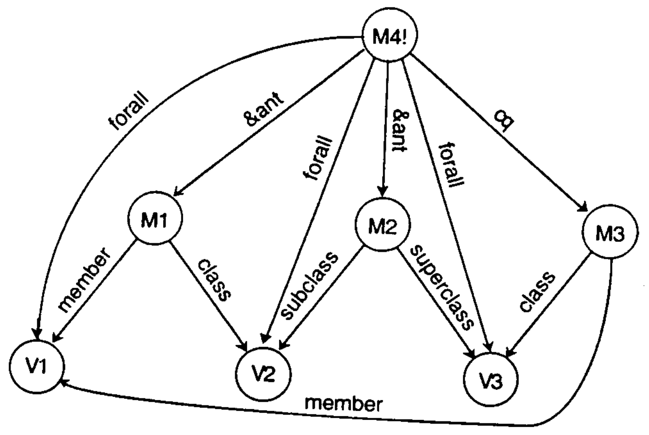

Figure 6.28: Transitivity of membership.

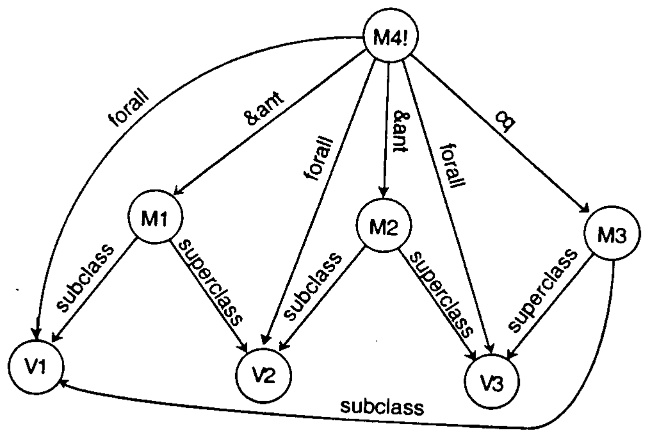

Figure 6.29: Transitivity of subclass.

The representation of this statement is SNePS is shown in Figure 6.29.

With these two rules and the network shown in Figure 6.18, SNePS is now capable of knowing that John is a mammal.

If we adopt the previous representation for asserting the transitivity of a relation, we would have to write similar rules for *all* the relations (case frames) represented in the network that satisfy the transitivity relation. An alternative way of representing the transitivity of the subclass relation is to explicitly state that the relation is transitive, and to have a rule that states what it means for a relation to be transitive (this representation is shown in Figure 6.30):

$$\begin{array}{ll}\forall(R) & \{Transitive(R)\} \\ & \vee\Rightarrow \\ & \{\forall(C_1, C_2, C_3)[\{R(C_1, C_2), R(C_2, C_3)\}\wedge \Rightarrow \{(R(C_1, C_3)\}]\}\end{array}$$

Stating that a relation is transitive, requires the possibility of talking *about* the relation. Since the case frames member/class and subclass/superclass do not allow to explicitly talk about the relation involved, the approach used in Figure 6.30 requires the use of the case frame rel/arg1/.../argn.

Notice that, due to the fact that, in SNePS nodes represent propositions (among other things), we are able to write propositions *about* propositions. In this way, the logic underlying the reasoning of SNePS is a higher-order logic. The advantage of this second approach of representation is that it can be applied to *any* transitive relation. Thus, a SNePS's user, after defining the network structure of Figure 6.30, just has to state that a certain relation is transitive to obtain the desired behavior. On the other hand, this second approach requires additional work from the inference system.

6.3.6 Inference in SNePS

There are two types of inference in SNePS, node-based inference and path-based inference.

Node-based inference

Node-based inference allows a node or a structure of nodes to be inferred from the existence of instances of patterns of nodes. Node-based inference

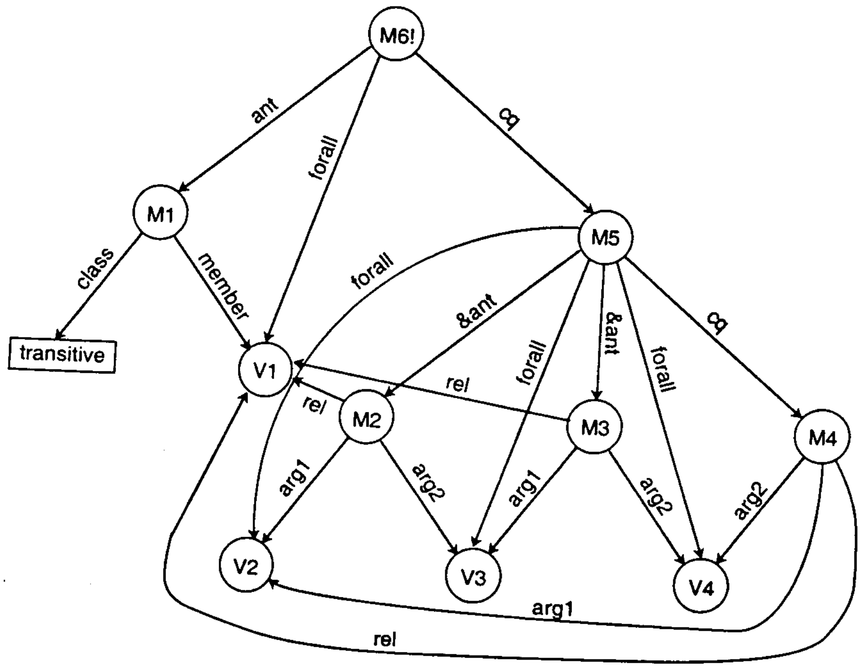

Figure 6.30: Meaning of a transitive relation.

relies on a pattern matching operation that takes a node (either a constant node or a pattern node) and locates the nodes in the network that match with it.

The SNePS inference system (in what concerns node-based inference) has the following characteristics: it allows both backward and forward inference to be performed; every rule in the network may be either used in backward or forward inference, or both; when a rule is used it is activated and remains active until explicitly de-activated by the user; the activated rules are assembled into a set of processes, called an *active connection graph* (acg) [McKay and Shapiro 81], which carry out the inferences; the acg also stores all the results generated by the activated rules; if during some deduction, the inference system needs some of the rules activated during a previous deduction, it uses their results directly instead of re-deriving them.

There are two main concepts involved in the implementation of the inference package, pattern-matching and the use of procedural (or active) versions of rules:

- The *pattern-matching process* is given a piece of the network (either to be deduced in backward inference or to be added in forward inference) and locates relevant rules in the network [Shapiro 77];
- The rules located by the pattern matcher are then *compiled* into a set of processes which are given to a multi-processing system for execution. The *multi-processing system* used by SNePS, called MULTI [McKay and Shapiro 80] is a LISP-based system mainly consisting of a simple evaluator, a scheduler, and system primitives.

In node-based inference SNePS's inference engine looks for nodes that correspond to propositions, using the non-standard connectives, and matches nodes against components of these propositions.

For example, from the network of Figure 6.31, Node `M1!` matches node `M3` with the substitution {`V1/John`, `V2/man`} and node `M2!` matches node `M4` with the substitution {`V2/man`, `V3/mammal`} (the nodes that match are linked, in Figure 6.31, by a dashed line). This allows the inference of an instance of the consequent of proposition represented by node `M6!` (`M5`) with the substitution {`V1/John`, `V3/mammal`}, this new node, `M7!` is represented as shadowed in Figure 6.31.

Node-based inference may either work in *backward inference* mode, trying

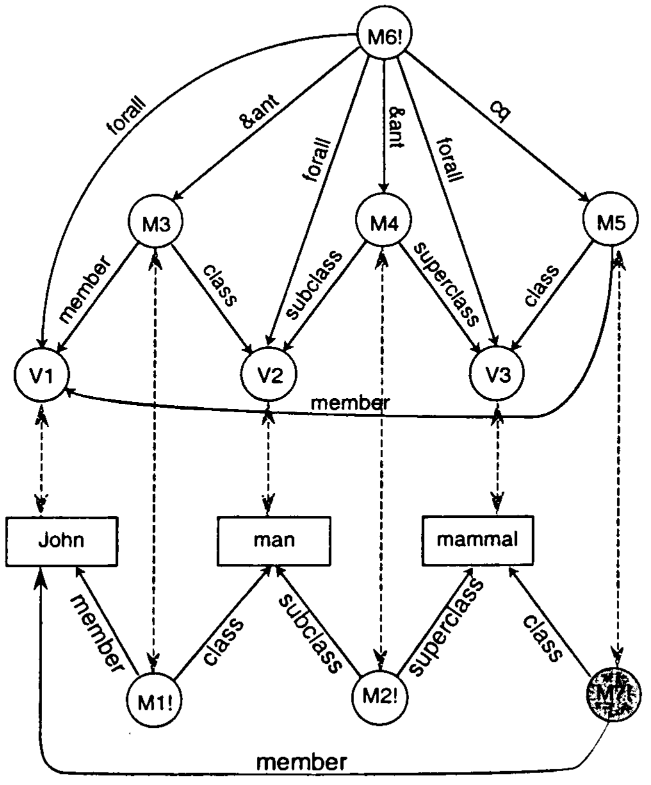

Figure 6.31: Network used in node-based inference.

to prove instances of consequents, or in *forward inference* mode, trying to find consequences of antecedents.

Path-based inference

Path-based inference allows an arc, or a path of arcs, between to nodes to be inferred, from the existence of a specified path of arcs between the same two nodes.

The result obtained in the previous section could have been obtained in a rather different way: Suppose that we have nodes `M1!` and `M2!` of Figure 6.31 that we tell SNePS that whenever the sequence of arcs

```
member-/class/subclass-/superclass
```

is found from a node `x` to a node `y` then it can infer the existence of the sequence of arcs `member-/class` from node `x` to node `y`; under these circumstances SNePS could obtain the same results as before.

However, the way this result is obtained is very different from the previous derivation: rather than relying on the existence of instances of patterns of nodes we rely on the existence of a sequence of arcs (which we call a *path*) from a node to another, this is *path-based inference* [Shapiro 78], [Srihari 81].

6.3.7 Foundations of the reasoning in SNePS

The reasoning of SNePS is ruled by a logic called SWM*[9] that was developed to support an ATMS-like system. The interesting aspect of supporting an ATMS-like system in SWM* is that the dependencies among propositions are computed by the system itself rather than having to force the user (or an outside system) to do this, as in many existing systems.

SWM* is loosely based on relevance logic [Anderson and Belnap 75]. The main features of relevance logic used in SWM* are the association of each wff with (1) a set containing all hypotheses (non-derived propositions) that

[9]After *S*hapiro, *W*and, and *M*artins. The SWM* system is a successor to the SWM system [Martins and Shapiro 83, 88], which, in turn, is a successor to the system of [Shapiro and Wand 76].

were *really* used in its derivation (the origin set) and (2) the statement of the rules of inference taking origin sets into account, specifying what should be the origin set of the resulting wff.

Another important issue in systems capable of revising their beliefs consists in the recording of the conditions under which contradictions may occur. This is important, because once the BRS discovers that a given set is inconsistent, it may not want to consider it again, and even if it wants to consider it, it wants to keep in mind that it is dealing with an inconsistent set.

Knowledge states

A *knowledge state*, written $[[KB, KIS]]$, is a pair that contains a knowledge base (KB) and a set of known inconsistent sets (KIS). The *knowledge base* is a set of supported wffs (propositions, written as wffs, together with an indication of dependencies between that particular wff and other wffs in the knowledge base); the *known inconsistent sets* is a set containing those sets of wffs in the KB that are known to be inconsistent.[10]

A knowledge state is intended to represent the knowledge that we have at a particular moment: KB contains all the propositions that were received from the outside world up to that moment and the subset of their consequences that was derived so far; and KIS contains information about all the sets that have been discovered to be inconsistent. The knowledge base does not necessarily contain all the consequences that can be drawn from the propositions it contains. It may even happen that the knowledge base is inconsistent but that the inconsistency has not been discovered. Whenever new inconsistencies are detected, they are recorded in the known inconsistent sets.

The *knowledge base* is a set of supported wffs. Supported wffs are of the form $<A, \tau, \alpha>$, where A is a wff with origin tag τ and origin set α. The pair (τ, α) is called the *support* of the supported wff <A, τ, α>. The support of a supported wff is associated with a *particular derivation of its wff*. There are standard formation rules for wffs, and the language that they define is denoted by $\mathcal{L}$. The *origin tag* indicates how the supported wff was placed

[10]It is important to distinguish between a set *being* inconsistent and a set *being known to be* inconsistent. An inconsistent set is one from which a contradiction *can be* derived; a set known to be inconsistent is an inconsistent set from which a contradiction *has been* derived.

in the knowledge base (i.e., whether it was supplied by an outside system or was generated during deduction). It is an element of the set $\{hyp, der, ext\}$: *hyp* identifies hypotheses, *der* identifies normally derived wffs within SWM*, and *ext* identifies special wffs whose origin set was extended. A supported wff with *ext* origin tag has to be treated specially in order to avoid the introduction of irrelevancies (for a discussion on this issue see [Martins and Shapiro 88]). The *origin set* indicates the dependencies of the supported wff on other wffs in the knowledge base. It is a set of hypotheses. The rules of inference of SWM* guarantee that the origin set of a supported wff contains *all and only* the hypotheses that were used in its derivation.

Some inference rules

The rules of inference of SWM* are grouped into two sets, pure logic rules and computational rules. *Pure logic rules* allow the introduction of new supported wffs into the knowledge base; *computational rules* update the information about sets known to be inconsistent.

The rules of inference make use of a function (Λ) to compute the origin tag of a supported wff resulting from the application of the rules of inference. The origin tag of a derived proposition will be *ext* if the origin tag of any of the propositions used in its derivation is *ext*, and will be *der* otherwise. The function Λ is defined as:

$$\Lambda(\tau_1, \tau_2) = \begin{cases} ext & \text{if } \tau_1 = ext \text{ or } \tau_2 = ext \\ der & \text{otherwise} \end{cases}$$

1. *Pure logic rules.* These rules correspond to traditional inference rules. They have the effect of adding new supported wffs to the KB. The addition of new supported wffs to the KB may be done in two different ways: a new supported wff is introduced from the outside (this is called a hypothesis) or a supported wff is derived from other supported wffs in the KB. Pure logic rules transform $[[KB, KIS]]$ into $[[KB', KIS]]$ where $KB \subset KB'$. The following are some of the pure logic rules:

   (a) *Hypothesis* (Hyp). From $[[KB, KIS]]$ and any wff $A \in \mathcal{L}$, we may infer $[[KB \cup \{<A, hyp, \{A\}>\}, KIS]]$.
   The wff A in $<A, hyp, \{A\}>$ is called a *hypothesis.*

   (b) *Negation Introduction* ($\neg$I). This rule states that from the hypotheses underlying a contradiction we can conclude that the conjunction of any number of them must be false under the assumption of the others.

From $[[KB, KIS]]$, in which $<A \wedge \neg A, \tau, \alpha \cup \{H_1, \ldots, H_n\}> \in KB$, we may infer $[[KB \cup \{<\neg(H_1 \wedge \ldots \wedge H_n), \Lambda(\tau,\tau), \alpha>\}, KIS]]$.

(c) *Implication Introduction* ($\rightarrow$I). From $[[KB, KIS]]$, in which $<C$, der, $\alpha \cup \{H\}> \in KB$, we may infer $[[KB \cup \{<H \rightarrow C$, der, $\alpha>\}$, $KIS]]$.

(d) *Implication Elimination* ($\rightarrow$E). From $[[KB, KIS]]$, in which $<A$, τ_1, $\alpha_1> \in KB$ and $<A \rightarrow B, \tau_2, \alpha_2> \in KB$, we may infer $[[KB \cup \{<B, \Lambda(\tau_1, \tau_2), \alpha_1 \cup \alpha_2>\}, KIS]]$.

(e) *And Introduction* ($\wedge$I). From $[[KB, KIS]]$ in which $<A, \tau_1, \alpha> \in KB$ and $<B, \tau_2, \alpha> \in KB$, we may infer $[[KB \cup \{<A \wedge B, \Lambda(\tau_1, \tau_2), \alpha>\}, KIS]]$; From $[[KB, KIS]]$ in which $<A, \tau_1, \alpha_1> \in KB$, $<B, \tau_2, \alpha_2> \in KB$, and $\alpha_1 \neq \alpha_2$, we may infer $[[KB \cup \{<A \wedge B, ext, \alpha_1 \cup \alpha_2>\}, KIS]]$.

(f) *And Elimination* ($\wedge$E). From $[[KB, KIS]]$ in which $<A \wedge B, \tau, \alpha> \in KB$, and $\tau \neq ext$, we may infer either $[[KB \cup \{<A, der, \alpha>\}, KIS]]$ or $[[KB \cup \{<B, der, \alpha>\}, KIS]]$.

2. *Computational rules.* These rules are triggered upon the discovery of inconsistent sets. They are obligatorily applied whenever a new inconsistent set is discovered. When this happens a new set is added to KIS. These rules transform $[[KB, KIS]]$ into $[[KB, KIS']]$ in which $KIS \neq KIS'$.

   (a) *Updating of Inconsistent Sets* (UIS). This rule is obligatorily applied whenever a new contradiction is detected.

   From $[[KB, KIS]]$ in which $<A \wedge \neg A, \tau, \alpha> \in KB$, and $(\forall s \in KIS)[s \not\subset \alpha]$, – this condition guarantees that the set α had not been discovered to be inconsistent – we *must* generate the knowledge state $[[KB, Min(KIS \cup \{\alpha\})]]$, where Min is a function that removes maximal sets.

   (b) *Derived Hypothesis* (DH). This rule is obligatorily applied when a supported wff is derived such that there is already a hypothesis in the KB with the same wff and that hypothesis belongs to a known inconsistent set.

   Given $[[KB, KIS]]$ in which $<H, hyp, \{H\}> \in KB$, $<H, \tau, \alpha> \in KB$, $\tau \neq hyp$, and $(\exists s \in KIS)[(H \in s)$ *and* $(\forall r \in KIS)$ $[r \not\subset ((s - \{H\}) \cup \alpha)]]$, – this means that the set $(s - \{H\}) \cup \alpha$ had not been discovered to be inconsistent – then we *must* generate the knowledge state $[[KB,\ Min(KIS \cup \{\sigma\ :\ (\exists s \in KIS)[(H \in s\ and\ \sigma = (s - \{H\}) \cup \alpha)\ and\ (\forall r \in KIS)[r \not\subset \sigma]]\})]]$.

Derivability

We define derivability within SWM* ($\vdash_{SWM*}$) as follows: Given $[[KB, KIS]]$, we write

$$[[KB, KIS]]\ \vdash_{SWM*}\ [[KB', KIS']]$$

if and only if there is a non-empty sequence of rules of inference of SWM* that transforms the knowledge state $[[KB, KIS]]$ in the knowledge state $[[KB', KIS']]$.

The rules of inference of SWM* guarantee the following results involving the derivability relation:

1. Every supported wff in the KB can be derived from a subset of other supported wffs in the KB. Given $[[KB, KIS]]$, $(\forall \mathcal{F} \in KB)$ $(\exists \Delta \subset KB)$ such that:

$$[[\Delta, \{\}]] \vdash_{SWM*} [[KB' \cup \{\mathcal{F}\}, KIS']].$$

2. The hypotheses in every known inconsistent set produce a contradiction. Given $[[KB, KIS]]$, $\forall s \in KIS$:[11]

$$[[\{< H, hyp, \{H\} >\ : H \in s\}, \{\}]] \vdash_{SWM*} [[KB' \cup \{\bot\}, KIS']])].$$

Given a set of wffs ($\Delta \subset \mathcal{L}$) and a single wff ($C \in \mathcal{L}$), we say that the single wff is derivable from the set of wffs, written $\Delta \vdash C$, if and only if,

$$[[\{< H, hyp, \{H\} >\ :\ H \in \Delta\}, \{\}]]\ \vdash_{SWM*}\ [[KB \cup \{< C, \tau, \alpha >\}, KIS]].$$

Contexts and Belief Spaces

As we said at the outset, among the propositions in the knowledge base, there are some in which the system believes and there may be some others in which the system does not believe. Inputs from the outside world or reasoning carried out by the system may lead to the detection of contradictions,

[11]The symbol $\bot$ denotes a contradiction.

in which case the system has to revise its beliefs in order to get rid of the contradiction and to accommodate the new information.

Up to now, we have been concerned with the definition of the rules of reasoning of a system. In this section, we address the issues of defining the beliefs of a system based on SWM*. The system we define is called MBR, for Multiple Belief Reasoner.

We define a *context* to be a set of hypotheses. A context determines a *belief space*, which is the set of all hypotheses defining the context and all the wffs in KB that were derived exclusively from them. A belief space is represented by $\ll [[KB, KIS]], C \gg$, where $(\forall H \in C)[<H, hyp, \{H\}> \in KB]$.

The belief space determined by a context is the subset of all the wffs existing in the KB that were derived (according to the rules of inference of SWM*) from the supported wffs corresponding to the hypotheses in the context. It contains those wffs that *have been derived* in the KB among all possible *derivable* wffs. The wffs in a belief space are characterized by the existence of a supported wff in KB with an origin set that is contained in the context, i.e.,

$$\ll [[KB, KIS]], C \gg = \{F \ : \ (\exists < F, \tau, \alpha > \in KB) \ and \ (\alpha \subset C)\}.$$

Any operation performed by MBR (query, addition, etc.) is associated with a context. We refer to the context under consideration, i.e., the context associated with the operation currently being performed, as the *current context*. While the operation is being carried out, the only propositions that will be considered are the propositions in the belief space defined by the current context. This belief space will be called the *current belief space*. A proposition is said to be *believed* if it belongs to the current belief space.

MBR and SNePS

The concepts that we described about SWM* and MBR have been implemented in SNePS. Every SNePS node that corresponds to a proposition is associated with a support in SWM*'s sense. The whole SNePS network corresponds to the KB in SWM*'s sense and a context is any set of hypotheses (nodes that were directly introduced by the user).

The interaction between a user and SNePS can be reduced to three main kinds of operations:

1. *Assert hypothesis H.* This operation generates the creation of a node in the network corresponding to the hypothesis H, $<H, hyp, \{H\}>$ and also adds the hypothesis H to the current context.

   The effect of this operation can be expressed as:

   $$\begin{aligned}&Assert(\ll [[KB, KIS]], C \gg,\ H) = \\ &\qquad\qquad \ll [[KB \cup \{<H, hyp, \{H\}>\}, KIS]],\ C \cup \{H\} \gg .\end{aligned}$$

2. *Deduce F.* This operation originates backward inference in the network (KB) trying to deduce the node corresponding to the wff F. The effect of this operation depends on the contents of KB and on the wff F, but the following things can happen:

   (a) New nodes (corresponding to supported wffs) are added to the knowledge base. KB is transformed into KB', where $KB \subset KB' \subset Cn(KB)$, where $Cn(KB)$ is the set of all logical consequences of KB, it is defined as $Cn(KB) = \{\mathcal{F} : (\exists [[KB^*, KIS^*]])$ $[\mathcal{F} \in KB^*\ and\ [[KB, KIS]] \vdash_{SWM*} [[KB^*, KIS^*]]]\}$.
   It may happen that $<F, \tau, \alpha> \in KB'$ and this is the desired result. This will happen whenever there are relevant rules in the network that allow the derivation of F.

   (b) During the inference originated by this operation new contradictions may be discovered, and, in this case, KIS is updated.

   (c) It may be discovered that the current context is inconsistent and that it should be revised (see next section).

   The effect of this operation can be expressed as:

   $$Deduce(\ll [[KB, KIS]], C \gg,\ F)\ =\ \ll [[KB', KIS']], C' \gg,$$

   where $KB \subset KB'$, $KIS \subset KIS'$, and $C' \subset C$.

3. *Add hypothesis H.* This operation originates forward inference in the network (KB) trying to deduce consequences of the node corresponding to the hypothesis H. The effect of this operation depends on the contents of KB and on the hypothesis H, but the following things may happen:

   (a) New supported wffs are added to the knowledge base. KB is transformed into KB', where $KB \subset KB' \subset Cn(KB \cup <H, hyp, \{H\}>)$. In this case $<H, hyp, \{H\}> \in KB'$.

(b) During the inference originated by this operation new contradictions may be discovered, and, in this case, KIS is updated.

(c) It may be discovered that the current context is inconsistent and that it should be revised (see next section).

The effect of this operation can be expressed as:

$$Add(\ll [[KB, KIS]], C \gg,\ H) = \\ \ll [[KB' \cup \{<H, hyp, \{H\}>\}, KIS']], C' \cup \{H\} \gg,$$

where $KB \subset KB'$, $KIS \subset KIS'$, and $C' \subset C$.

This operation is completed when no more information can be deduced from H or its consequences.

Detection of contradictions

When a contradiction is detected within SNePS (and MBR) one of two things can happen, depending on whether the contradiction belongs to the current belief space or belongs to a belief space strictly containing the current belief space. The main difference between them is that the former may require changes in the current context and allows the deduction of new supported wffs, while the latter leaves this context unchanged and does not allow the addition of new wffs to the knowledge base.

Suppose that the current belief space is $\ll [[KB, KIS]], C \gg$ and that KB contains the supported wff $<A \wedge \neg A, \tau, \alpha>$. Suppose, furthermore, that α does not contain any member of KIS ($(\forall s \in KIS)[s \not\subset \alpha]$), meaning that this contradiction had not been discovered yet. In this case, one of two things will happen:

1. *The contradictory wff does not belong to the current belief space.* This means that $\alpha \not\subset C$. In this case, the contradiction is recorded (through the application of UIS), but nothing more happens. The effect of doing so is to record that the set of hypotheses α is now known to be inconsistent.

2. *The contradictory wff belongs to the current belief space.* This means that $\alpha \subset C$. In this case, UIS is applied, resulting in the updating of the sets known to be inconsistent. The rule of $\neg$I can be applied (generating new supported wffs in the knowledge base) and a revision

of beliefs should be performed if we want to work within a consistent belief space.

This revision of beliefs is accomplished by removing hypothesis from the current context. Since MBR only considers wffs in the current belief space, the removal of hypothesis from the current context entails the removal of wffs from the current belief space. The resolution of a contradiction in the current belief space entails a *contraction* in Gärdenfors and Makinson's sense [Gärdenfors and Makinson 1988]. This contraction is performed through a family of functions R_H^-, indexed by the wff, H, to be removed:

$$R_H^-(\ll [[KB, KIS]], C \gg) = \ll [[KB, KIS]], C - \{H\} \gg .$$

From SWM*'s standpoint, after the discovery of the inconsistent set α ($\alpha \subset C$), the removal of *any one* of the hypotheses in α is *guaranteed* to remove this contradiction from the current belief space and restore unknown inconsistency to the current context if it was not known to be inconsistent before discovery of this contradiction.

Examples

In this section we show two examples of the resasoning followed by SNePS, using an interface called SNePSLOG [Shapiro, McKay, Martins, and Morgado 81], [Matos and Martins 91]. SNePSLOG allows the use of logic for the interaction with SNePS (all the input and output is done trough logical formulas) and SNePSLOG translates the logical formulas to and from SNePS nodes.

Example 1: Flying horses[12]

Suppose that we begin with the knowledge state $[[\{\}, \{\}]]$, and that we assert the following hypotheses.[13]

$swff1:\quad < White(Pegasus), hyp, \{wff1\} >$
$swff2:\quad < Horse(Tornado), hyp, \{wff2\} >$

[12]I am gratefull for Maria R. Cravo for contributing this example.

[13]We use the notation $swff1$: $<White(Pegasus), hyp, \{wff1\}>$ to denote that $<White(Pegasus), hyp, \{White(Pegasus)\}>$ is a supported wff called $swff1$ and that the wff $White(Pegasus)$ is represented by $wff1$.

$swff3: \quad < \forall(x)[Horse(x) \rightarrow \neg Flies(x)], hyp, \{wff3\} >$
$swff4: \quad < \forall(x)[WingedHorse(x) \rightarrow Flies(x)], hyp, \{wff4\} >$
$swff5: \quad < \forall(x)[WingedHorse(x) \rightarrow Horse(x)], hyp, \{wff5\} >$
$swff6: \quad < \forall(x)[WingedHorse(x) \rightarrow HasWings(x)], hyp, \{wff6\} >$

At this point, our knowledge state is:

$$[[\{swff1, swff2, swff3, swff4, swff5, swff6\}, \{\}]].$$

And the current belief space is:

$$\ll [[\{swff1, swff2, swff3, swff4, swff5, swff6\}, \{\}]], \\ \{wff1, wff2, wff3, wff4, wff5, wff6\} \gg.$$

Suppose, that we add the following hypothesis (this originates forward inference, trying to deduce the consequences of $WingedHorse(Pegasus)$):

$$swff7: \quad < WingedHorse(Pegasus), hyp, \{wff7\} >$$

In the current belief space, we can derive $swff8$, $swff9$, and $swff10$:

$swff8: \quad < \neg Flies(Pegasus), der, \{wff3, wff5, wff7\} >$
$swff9: \quad < Flies(Pegasus), der, \{wff4, wff7\} >$
$swff10: \quad < Flies(Pegasus) \wedge \neg Flies(Pegasus), ext,$
$\{wff3, wff4, wff5, wff7\} >$

We thus discover that the set $\{wff3, wff4, wff5, wff7\}$ is inconsistent. When this happens, the rule of UIS is applied, resulting in the following updated knowledge state:

$$[[\{swff1, swff2, swff3, swff4, swff5, swff6, swff7, swff8, swff9, \\ swff10\}, \ \{\{wff3, wff4, wff5, wff7\}\}]].$$

We should revise the system's beliefs by removing from the current context at least one hypothesis from the set $\{wff3, wff4, wff5, wff7\}$. The most natural decision would be to drop the hypothesis that $\forall(x)[WingedHorse(x) \rightarrow Horse(x)]$, and to consider the belief space:

$$\ll [[\{swff1, swff2, swff3, swff4, swff5, swff6, swff7, swff8, swff9, swff10\}, \{\{wff3, wff4, wff5, wff7\}\}]], \{wff1, wff2, wff3, wff4, wff6, wff7\} \gg .$$

Example 2: The Russell set[14]

Suppose we have a knowledge state containing only one hypothesis, which states that there is a set, S, that contains all the sets that are not members of themselves:

$$[[\{< \forall(x)[(\neg(x \in x) \rightarrow x \in S) \wedge (x \in S \rightarrow \neg(x \in x))], hyp, \{wff1\} >, \{\}]].$$

If we add the following hypothesis to the current belief space

$$swff2: \quad < S \in S, hyp, \{wff2\} >$$

we can generate the supported wff $swff3$, using the rule of universal elimination:[15]

$$swff3: \quad < (\neg(S \in S) \rightarrow S \in S) \wedge (S \in S \rightarrow \neg(S \in S)), der, \{wff1\} >$$

Now, we can derive the following:

$$\begin{array}{ll} swff4: & < S \in S \rightarrow \neg(S \in S), der, \{wff1\} > \\ swff5: & < \neg(S \in S), der, \{wff1, wff2\} > \\ swff6: & < (S \in S) \wedge \neg(S \in S), ext, \{wff1, wff2\} > \end{array}$$

At this point, a contradiction is detected ($swff6$), triggering the application of UIS, which produces the knowledge state:

$$[[\{swff1, swff2, swff3, swff4, swff5, swff6\}, \{\{wff1, wff2\}\}]]$$

We can apply the rule of $\neg$I to $swff6$ to infer $swff7$:

$$swff7: \quad < \neg(S \in S), ext, \{wff1\} >$$

We now revise the system's beliefs by applying the following contraction:

[14]I am gratefull for Stuart C. Shapiro for contributing this example.

[15]This rule was not discussed in this paper, but its use is obvious (see [Martins and Shapiro 1988]).

$$R^{-}_{wff2}(\ll [[\{swff1, swff2, swff3, swff4, swff5, swff6, swff7\}, \{\{wff1, wff2\}\}]], \{wff1, wff2\} \gg) = \ll [[\{swff1, swff2, swff3, swff4, swff5, swff6, swff7\}, \{\{wff1, wff2\}\}]], \{wff1\} \gg$$

We can perform further reasoning, generating $swff8$ by $\wedge$E applied to $swff3$; $swff9$ by MP applied to $swff7$ and $swff8$; and $swff10$ by $\wedge$I applied to $swff7$ and $swff9$:

$swff8: \; < \neg(S \in S) \rightarrow S \in S, der, \{wff1\} >$
$swff9: \; < S \in S, ext, \{wff1\} >$
$swff10: \; < S \in S \wedge \neg(S \in S), ext, \{wff1\} >$

Again, UIS is applied to $swff10$, resulting in the knowledge state:

$$[[\{swff1, swff2, swff3, swff4, swff5, swff6, swff7, swff8, swff9, swff10\}, \{\{wff1\}\}]].$$

If further reasoning is to be performed in a consistent belief space, then $wff1$ (which is itself inconsistent) must be removed from the current context.

6.3.8 Conclusions

$\boxed{\text{To be added.}}$

6.4 Other kinds of semantic networks

6.4.1 Semantic primitives

Associated with each representational formalism there is the problem of deciding the vocabulary to be used in the formalism. For example, if we use logic, we have to decide the names of the predicates, functions, and constants to use; if we use SNePS we have to decide the case frames to adopt for our representation.

The research in semantic primitives addresses this problem. We can define a *semantic primitive* as a representational symbol that is used without defini-

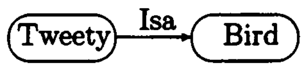

Figure 6.38: Impossible representation in SNePS.

```
  I understand
* A nostril is a part of a nose
  I understand
* A professor is a teacher
  I understand
* A teacher is a person
  I understand
* Is a nostril part of a professor
  Yes
* A person is a living-creature
  I understand
* Is a nostril part of a living-creature
  Sometimes
```

3. Woods introduces the distinction between assertional and structural arcs. Explain the meaning of each of these types of arcs and provide networks that use each one of them.

4. Consider the alternative representations for class membership shown in Figures 6.16 and 6.18.

   (a) Discuss the advantages and drawbacks of each of them in what respects the expressive power of the representation.

   (b) Discuss the advantages and drawbacks of each of them regarding the possibility of defining rules that define the properties of the membership relation.

5. SNePS does not allow the representation shown in Figure 6.38 for the statement "Tweety is a bird". Explain the reason for this decision.

6. Explain the difference between the representations of figures 6.21 and 6.38. Why is the representation of Figure 6.21 legal?

7. What does ${}_2\mathbb{X}_1^1\{\alpha, \beta\}$ represent?

8. What does ${}_n\mathbb{X}_0^0\ \{\alpha_1, \ldots, \alpha_n\}$ represent?

9. Using SNePS, represent, in a graphical form, the following statements: Anna knows that John bought a dog. The dog is a German Shepherd. John paid 300 dollars for the dog. Anna also knows that Stu sold his dog. Anna believes that John bought Stu's dog.

10. Using SNePS, represent, in a graphical form, the following statements: John and Mary are lawyers. Charles believes that all the lawyers are rich. Charles knows John's address but he does not know that he and Mary live together.

11. Using SNePS, represent, in a graphical form, the following statements: John is a professor and earns 50,000 dollars a year. His friends believe that he earns much more. They do not know that he earns as much as a police officer.

12. Represent in SNePS the concept of a reflexive relation.

13. Represent in SNePS the concept of a symmetric relation.

14. Represent in SNePS the concept of an equivalence relation.