An Introduction to SNePS 3

Stuart C. Shapiro

Department of Computer Science and Engineering
and Center for Cognitive Science
State University of New York at Buffalo
226 Bell Hall
Buffalo, NY 14260-2000
shapiro@cse.buffalo.edu

Abstract. This paper provides an introduction to SNePS 3, the latest entry in the SNePS family of knowledge representation and reasoning (KRR) systems. The emphasis is on SNePS 3 as an example of a logic-based network KRR system.

1 Introduction

SNePS 3 is the latest entry in the SNePS family [1] of knowledge representation and reasoning (KRR) systems.[1] It is based on SNePS 2.5 [2] and ANALOG [3,4], and is currently being implemented in CLOS, the Common Lisp Object System [5]. This paper provides an introduction to SNePS 3 as an example of a logic-based network KRR system. In the rest of this paper, I will use the term "SNePS" when discussing features that are generally true of formalisms in the SNePS family, and "SNePS 3" when discussing features that distinguish SNePS 3 from other family members.

Due to space considerations, I will only be able to provide an informal and incomplete introduction to SNePS 3. Section 6 will be a more formal, but still incomplete, introduction to the syntax of the SNePS 3 language.

Informally, information is represented in SNePS as a network of nodes and labeled directed arcs. An arc label may be used on arbitrarily many different arcs, but each node has a unique identifier. For example, Fig. 1, based on a project in which Cassie, a SNePS-based agent [6], is embodied as a Foveal Extra-Vehicular Activity Helper-Retriever (FEVAHR) robot [7], represents the following information:[2]

> *Cassie is talking to and looking at Stu. Cassie is a FEVAHR. FEVAHRs are robots. Stu and Bill are people. People and robots are agents.*

How the network of Fig. 1 represents this information, *i.e.,* the syntax and semantics of SNePS networks, will be explained below.

[1] "SNePS" originally stood for "Semantic Network Processing System." Now, I would prefer it be thought of as a name in its own right.

[2] Fig. 1 shows only a small part of the FEVAHR knowledge base, chosen to illustrate the issues discussed in this paper.

B. Ganter and G.W. Mineau (Eds.): ICCS 2000, LNAI 1867, pp. 510–524, 2000.

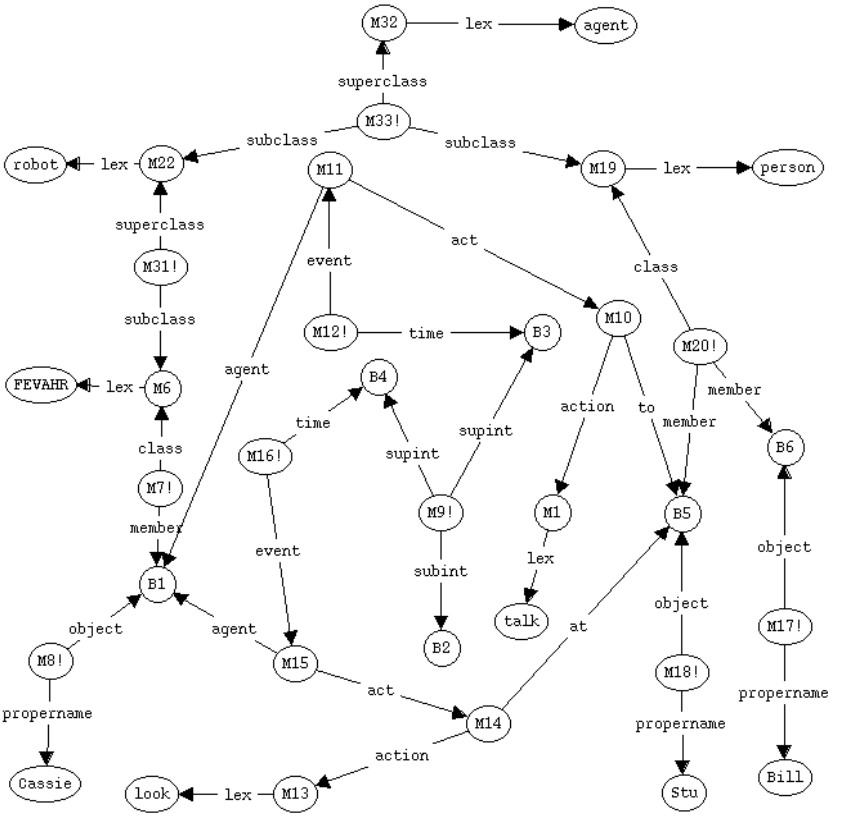

Fig. 1. A SNePS network representing the information that: Cassie is talking to Stu (`M12!`); Cassie is looking at Stu (`M16!`); Cassie is a FEVAHR (`M7!`); FEVAHRs are robots (`M31!`); Stu and Bill are people (`M20!`); people and robots are agents (`M33!`). Tables 1–4 give the intended semantics of each node.

2 KRR Systems as Logics

Every KRR system is a logic, in the sense that it has, or should have, a well-defined syntax, compositional semantics, and inference procedure. SNePS has been designed specifically as a logic to support natural language (NL) competent agents [3,4,6,8,9]. SNePS constitutes the "language of thought" of a SNePS-based agent. The domain of discourse of that language, the set of entities that are denoted by well-formed SNePS expressions, is the domain of all mental entities conceivable by the agent. Adding a new expression to a SNePS network implements the agent's conceiving of the entity represented by that expression.

The basic SNePS principles are

Propositional Semantic Network: The only well-formed SNePS expressions are nodes.
Term Logic: Every well-formed SNePS expression is a term.
Intensional Representation: Every SNePS term represents (denotes) an intensional (mental) entity.
Uniqueness Principle: No two SNePS terms denote the same entity.
Paraconsistent Logic: A contradiction does not imply anything whatsoever.

The significance of SNePS being a *propositional* semantic network is that only nodes are well-formed expressions having semantics, whereas in many other network-based KRR systems, including many semantic networks, arcs denote propositions—relations that are asserted to hold between entities. (See [10] for a discussion of assertional *vs.* structural information.)

The significance of SNePS being a term logic is that proposition-denoting terms may be arguments of other terms without leaving first-order logic [11].

The significance of SNePS using intensional representation is that cognitively distinct mental entities are denoted by distinct terms even if they are co-extensional—the entire network forms an opaque context in which there is no substitution of equals, because no two terms are fully equal [12,13,14,15]. This principle can also be read as "*Every* SNePS term denotes a mental entity," which means that no terms are created for purely technical reasons, and that even the analogue of logical variables, "variable nodes", have consistent compositional semantics denoting mental entities. Variable nodes did not have such semantics in earlier versions of SNePS [16]. Supplying that was a main motivator of ANALOG [3,4] and SNePS 3.

The Uniqueness Principle supports intensional representation, and imposes a requirement of structure-sharing on the implementation—no two distinct but structurally equal terms exist in a network.

SNePS being a paraconsistent logic means that a contradiction in one area of the knowledge base would not "corrupt" the information in another area of the knowledge base.

3 Levels

To a large extent, SNePS is a logical-level semantic network [17]. Just as a Prolog programmer must choose the predicates to be used in a Prolog program, a SNePS user must choose the arc labels and other representational techniques to use in a SNePS-based application. This person is called a "knowledge engineer", and the job of a knowledge engineer has been referred to as "conceptualization" [18, p. 9]. For example, Figures 2 and 3 show two possible ways of representing *Cassie is a FEVAHR*, and another is shown in Fig. 1.

However, just as the syntax, semantics, and inference mechanism of Horn clauses is built into Prolog, there is a level at which syntax, semantics, and an inference mechanism is built into SNePS. These will be discussed below.

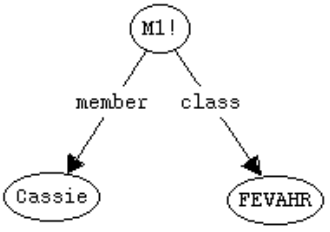

Fig. 2. A possible representation of *Cassie is a FEVAHR.*

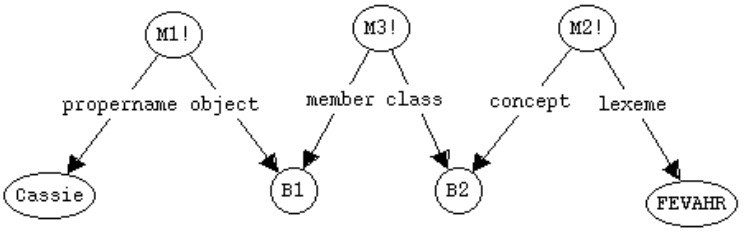

Fig. 3. Another possible representation of *Cassie is a FEVAHR.*

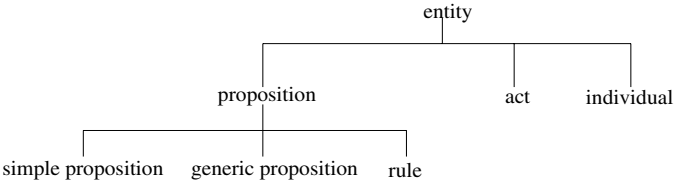

Fig. 4. The initial taxonomy of SNePS semantic classes.

4 Semantics

Every SNePS 3 node must be given a semantic class recognized by the system. The initial semantic taxonomy, which may be extended by the knowledge engineer, is shown in Fig 4.

Proposition nodes have an assertional status that is recognized by the inference mechanism (see Sect. 5 below). Rule nodes are proposition nodes that can be used for node-based inference. An example is shown in Fig. 5. Generic proposition nodes, such as `M89!` in Fig. 6, can be used for subsumption-based inference. Simple proposition nodes are proposition nodes that are neither generic propositions nor rules. Act nodes may be performed by the SNeRE acting executive [19,20], [2, Chap. 4].[3] Individual nodes are nodes that are neither proposition nodes nor act nodes.

[3] Due to space limitations, act nodes and the acting executive will not be discussed in this paper.

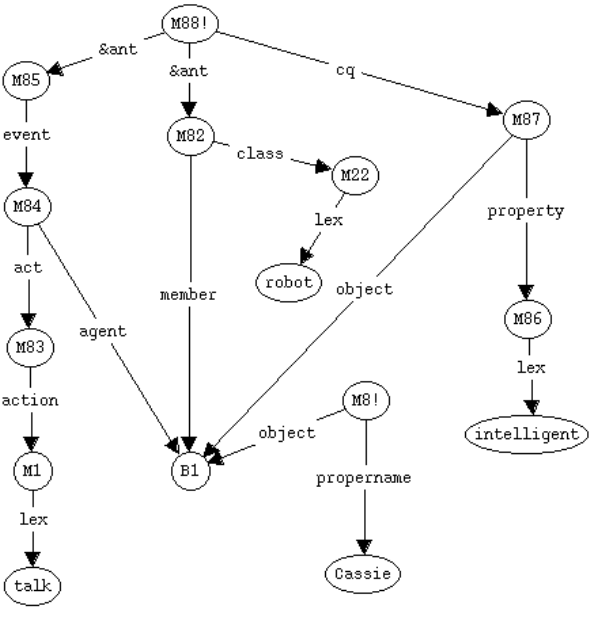

Fig. 5. M88! is a rule node denoting the proposition, *If Cassie is a robot that talks, then Cassie is intelligent.*

The SNePS 3 node classes and the categories of entities represented by those nodes as expressed in the SNePS formalism may be redundant, but this is because we, as SNePS 3 designers, did not want to specify *a priori* how the knowledge engineers could choose to represent categories. For example, node M6 of Fig. 1, node FEVAHR of Fig. 2, and node B2 of Fig. 3 might each represent the category of FEVAHR robots in different conceptualizations. When necessary to distinguish these two levels of semantics, I shall use the terms "SNePS semantics" and "domain semantics."

The intended domain semantics (intended by me, the knowledge engineer of this project) of the nodes in Fig. 1 are shown in Tables 1–4.[4] There, $[\![n]\!]$ is used to refer to the denotation of the SNePS term n. Lexemes, agents, times, categories, actions, acts, events, and propositions are all assumed to be entities in the domain of discourse, which, in this case, is the universe of Cassie's mental entities. Act-denoting terms are nodes in the SNePS act class, proposition-denoting terms are nodes in the SNePS proposition class, and all other terms are nodes in the SNePS individual class.

[4] Table 2 gives the semantics of B2 as "an interval of time." Cassie/FEVAHR contains a variable, NOW, whose value is the term denoting the current time, and which changes as time moves [21]. At the time of the snapshot of Fig. 1, B2 is the value of NOW, which is why Cassie's talking and looking is expressed in the present tense.

Table 1. The semantics of lexeme-denoting terms from Fig. 1

`agent`:	The English lexeme "agent".
`Bill`:	The English lexeme "Bill".
`Cassie`:	The English lexeme "Cassie".
`FEVAHR`:	The English lexeme "FEVAHR".
`look`:	The English lexeme "look".
`person`:	The English lexeme "person".
`robot`:	The English lexeme "robot".
`Stu`:	The English lexeme "Stu".
`talk`:	The English lexeme "talk".

Table 2. The semantics of agent-, time-, and category-denoting terms from Fig. 1

`B1`: Cassie.	`B2`: An interval of time.	`M6`: The category of FEVAHRs.
`B5`: Stu.	`B3`: An interval of time.	`M19`: The category of people.
`B6`: Bill.	`B4`: An interval of time.	`M22`: The category of robots.
		`M32`: The category of agents.

Table 3. The semantics of action-, act-, and event-denoting terms from Fig. 1

`M1`:	The action of talking.
`M13`:	The action of looking.
`M10`:	The act of talking to Stu.
`M14`:	The act of looking at Stu.
`M11`:	The event of Cassie's talking to Stu.
`M15`:	The event of Cassie's looking at Stu.

Table 4. The semantics of proposition-denoting terms from Fig. 1

`M7!`:	The proposition that Cassie is a FEVAHR.
`M8!`:	The proposition that Cassie's name is "Cassie".
`M9!`:	The proposition that ⟦`B2`⟧ is a subinterval of ⟦`B3`⟧ and ⟦`B4`⟧.
`M12!`:	The proposition that Cassie is talking to Stu throughout ⟦`B3`⟧.
`M16!`:	The proposition that Cassie is looking at Stu throughout ⟦`B4`⟧.
`M17!`:	The proposition that Bill's name is "Bill".
`M18!`:	The proposition that Stu's name is "Stu".
`M20!`:	The proposition that Stu and Bill are people.
`M31!`:	The proposition that FEVAHRs are robots.
`M33!`:	The proposition that robots and people are agents.

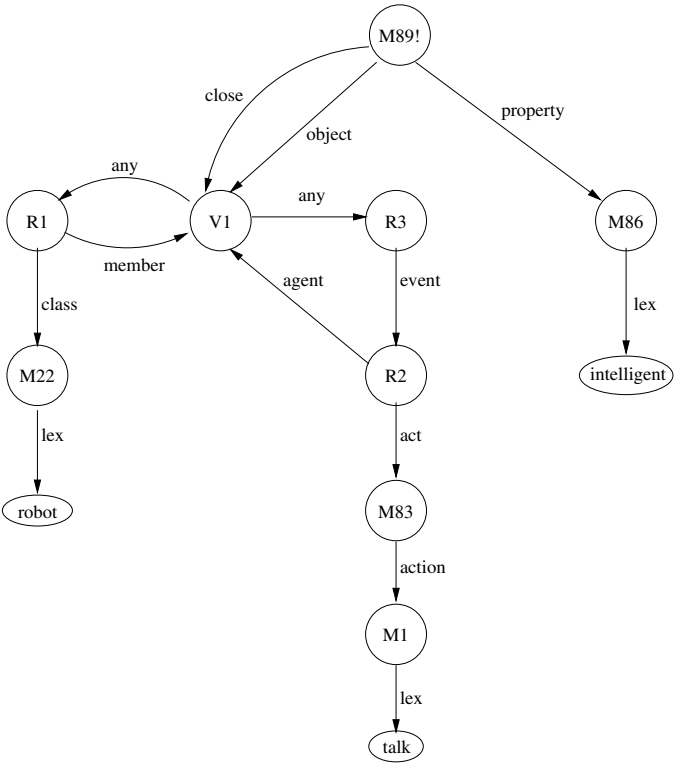

Fig. 6. `M89!` is a generic proposition node denoting the proposition that *Any robot that talks, is intelligent.*

5 Inference Methods

A SNePS-based agent might believe only some of the propositions represented in the network. For example, Cassie might believe that Bill believes that Stu is tall, without herself believing it. We notate believed propositions by appending an exclamation mark (`!`) to the identifiers of nodes that denote believed propositions, and we refer to such nodes as *asserted*. All the proposition nodes of Figures 1–3 are asserted. In Fig. 5, nodes `M82`, `M85`, and `M87` are unasserted proposition nodes, and node `M88!` is an asserted proposition node.

Inference in SNePS is a method of computing newly asserted nodes from previously asserted nodes. Thus, inference causes the agent to have new beliefs. The newly asserted nodes might previously have been in the network, denoting then unbelieved propositions, or they might be created by the inference engine.

There are four inference methods in SNePS, the first two of which distinguish SNePS from logic-based but non-network-based KRR systems. Briefly, and informally, these are:

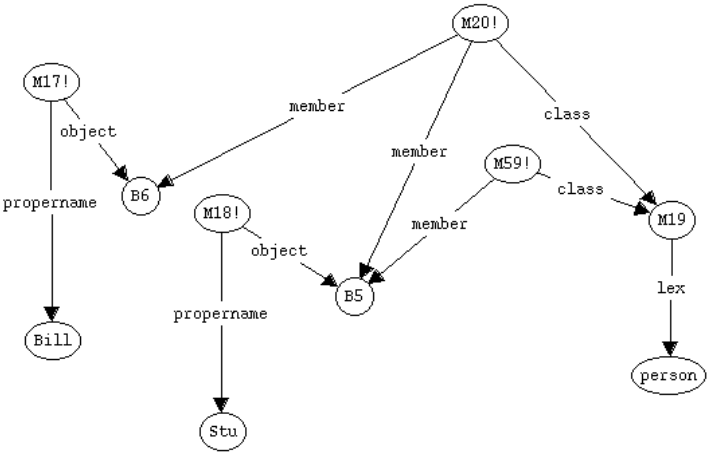

Fig. 7. Node `M20!`, denoting the proposition that Stu and Bill are people, implies `M59!`, denoting the proposition that Stu is a person, by wire-based inference.

Wire-based inference, referred to as "reduction inference" in previous versions of SNePS (notably in [22]), is an inference method whereby a proposition node *m1* implies a proposition node *m2* with a subset or a superset of *m1*'s arcs. (See Secs. 6.1 and 6.2.) For example, node `M20!` in Fig. 7, repeated from Fig. 1, implies node `M59!` by wire-based inference.

Path-based inference [22,23,24] is an inference method whereby a path of arcs from a proposition node *m1* to a node *m2* (of any class) may imply a proposition node *m3* with all of *m1*'s arcs plus an additional one to *m2*. Path-based inference must be sanctioned by path-based inference rules that are given to the SNePS system, but not represented in the SNePS object language. (See [2, Sect. 2.5.2] for the syntax and semantics of paths.) One possible path-based inference rule is

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

Given this rule, node `M60!` of Fig. 8, denoting the proposition that Stu is an agent, may be inferred from `M59!`, denoting the proposition that Stu is a person, and the path of arcs from `M59!` to `M32` by path-based inference followed by wire-based inference so that there is only one `class` arc emanating from `M60!`.

Node-based inference uses nodes that are the analogue of non-atomic formulas in first-order predicate logic to represent domain rules that SNePS can use to infer newly asserted proposition nodes from previ-

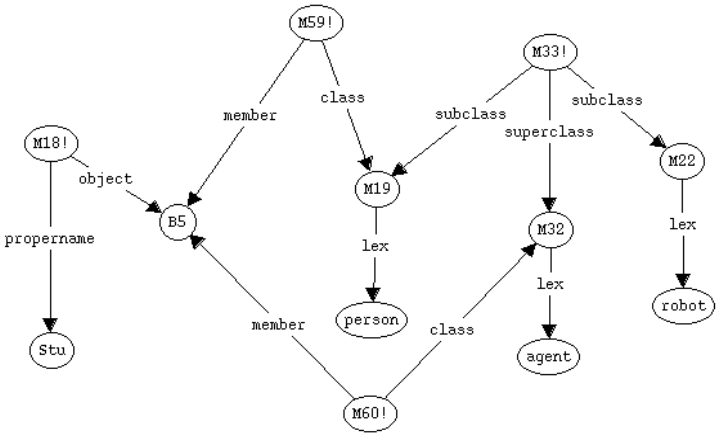

Fig. 8. Node M60!, denoting the proposition that Stu is an agent, may be inferred from M59!, denoting the proposition that Stu is a person, and the path of arcs from M59! to M32 by path-based inference followed by wire-based inference.

ously asserted proposition nodes. For example, in Fig. 5, node M88! is an "and-entailment"—a kind of entailment with M82 and M85 as conjoined antecedents and M87 as a consequent. M88! denotes the domain rule, *If Cassie is a robot that talks, then Cassie is intelligent.* Since M82 follows from Fig. 1's M7! by path-based inference, and M85 follows from Fig. 1's M12! by wire-based inference, M87 may be inferred by node-based inference.

Subsumption inference is an inference method in SNePS 3 that is not available in earlier versions of SNePS, and uses the ANALOG [3,4] technique of representing variables. For example, Fig. 6 shows a SNePS 3 representation of the generic proposition, *Any robot that talks, is intelligent.* V1 is a structured variable that denotes the arbitrary talking robot [25], R1 and R3 are restrictions on V1, and M89! denotes the asserted generic proposition. Since R1 subsumes Fig. 5's M82 and R3 subsumes Fig. 5's M85 and those nodes follow from Fig. 1 as explained above, therefore V1 subsumes Fig. 1's (and Fig. 5's) B1, M89! subsumes Fig. 5's M87, and M87 follows from M89! by subsumption inference.

6 Syntax

The syntax of the SNePS 3 language is defined in terms of *nodes*, *relations*, *wires*, *cables*, and *cablesets* (see also [3,4,22]). Informally, a node is what I have been calling a node heretofore, and a relation is an arc label. A wire is a pair, $\langle r, n \rangle$,

of a relation r and a node n. A cable is a pair, $\langle r, ns\rangle$ of a relation r and a set of nodes (or "nodeset") ns. A cableset is a set of cables, $\{\langle r_1, ns_1\rangle, \ldots, \langle r_k, ns_k\rangle\}$, such that no two r_i are the same. A set of wires all of which have the same relation forms a cable. A set of wires with different relations forms a cableset. Nodes that I have been informally presenting as labeled circles or ovals with arcs emanating from them formally *are* cablesets.

With the previous paragraph as introductory, and the foregoing sections as motivational, I can now give a more formal presentation of SNePS 3 syntax.

6.1 Relations

A SNePS 3 *relation* is a four-tuple, $\langle name, type, adjust, limit\rangle$, where the four constituents are:

***name*:** A symbolic name of the relation. No two relations may have the same name. Relation names were used in Figures 1–8 as labels on directed arcs.

***type*:** The SNePS semantic class of the nodes in the range of this relation, *i.e.,* of the nodes pointed to by arcs labeled with this relation.

***adjust*:** Either `expand`, `reduce`, or `none`. This specifies how wire-based inference treats instances of this relation. For example, Fig. 7 illustrates the reducibility of the relation named `member`. However, the relation named `&ant`, shown in Fig. 5 is expandable. A value of `none` means that this relation is not subject to wire-based inference.

***limit*:** The minimal size of a nodeset that can be paired with this relation in a cable. If *adjust* is `reduce`, then *limit* is the minimal allowed reduction. For example, to prevent Fig. 7's node `M20!` from implying a node with only a `class` arc and no `member` arcs, the *limit* of the relation named `member` is 1. If *adjust* is `expand`, then *limit* is the minimal nodeset-size of a cable that allows more wires to be added. For example, it does not make sense to add `&ant` arcs to a node that is not a kind of entailment.

Assuming that the knowledge engineer has declared the classes `category`, `event`, and `time` to be subclasses of the SNePS class of individual nodes, some example relations would be:

$\langle$`member`, `entity`, `reduce`, `1`$\rangle$ $\langle$`class`, `category`, `reduce`, `1`$\rangle$
$\langle$`event`, `event`, `reduce`, `1`$\rangle$ $\langle$`time`, `time`, `reduce`, `0`$\rangle$
$\langle$`&ant`, `proposition`, `expand`, `1`$\rangle$ $\langle$`cq`, `proposition`, `reduce`, `1`$\rangle$

6.2 Case Frames

A *case frame* is a pair, $\langle c, \rho\rangle$, where c is a SNePS semantic class and ρ is set of relations. Informally, c is the class of all nodes that have the given set of arcs emanating from them. A more formal explanation is given below.

Case frames are partially ordered by the relation *adjustable* ($\sqsupseteq_{adj}$). Case frame $\langle c_1, \rho_1\rangle$ is adjustable to case frame $\langle c_2, \rho_2\rangle$ ($\langle c_1, \rho_1\rangle \sqsupseteq_{adj} \langle c_2, \rho_2\rangle$) iff

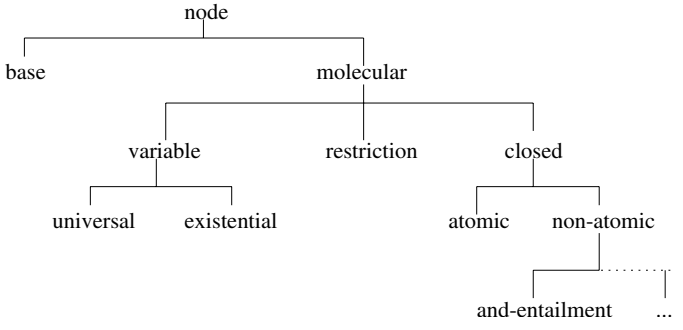

Fig. 9. Part of the hierarchy of the syntactic classes of SNePS nodes.

1. $c_1 = c_2$;
2. every relation in $\rho_1 - \rho_2$ is reducible and has *limit*=0;
3. every relation in $\rho_2 - \rho_1$ is expandable and has *limit*=0.

For example, the fact that Fig. 5's `M85` follows from Fig. 1's `M12!` by wire-based inference is partially justified by the fact that
$\langle$`proposition`, $\{\langle$`event`, `event`, `reduce`, `1`$\rangle$, $\langle$`time`, `time`, `reduce`, `0`$\rangle\}\rangle$
$\sqsupseteq_{adj}$
$\langle$`proposition`, $\{\langle$`event`, `event`, `reduce`, `1`$\rangle\}\rangle$

6.3 Nodes

Before entering nodes into a SNePS 3 network, the knowledge engineer must declare the case frames to be used. The system will check that relations are being used consistently. The knowledge engineer may also add additional semantic classes to the SNePS 3 class hierarchy, and use these new classes in the case frames.

A valid SNePS 3 node must be in one of the syntactic classes in the SNePS 3 syntactic hierarchy, part of which is shown in Fig. 9.

A base node consists only of an identifier. A base node identifier may be created by a SNePS 3 user. Some examples of these in Fig. 1 are `Cassie`, `robot`, and `look`. Other base nodes may be created and named by the system. Examples of these in Fig. 1 are `B1` and `B2`. When a base node is created, it must be given a SNePS 3 semantic class.

An atomic closed molecular node is a cableset $\{\langle r_1, ns_1\rangle, \ldots, \langle r_k, ns_k\rangle\}$ such that:

1. the set $\{r_1, \ldots r_k\}$ is the relation set of some declared case frame $\langle c, \rho\rangle$;
2. for each i, each node in ns_i is of the semantic class which is the *type* of r_i;
3. for each i, the cardinality of ns_i is at least the *limit* of r_i.

If a cableset satisfies these three conditions, it is considered to be an *instance* of the case frame $\langle c, \rho\rangle$, and is entered into the network as a node in the semantic class c. Closed molecular nodes are given identifiers of the form `M`n by the system.

We can refer to a cableset either by using the cableset notation, using just its identifier, or using both. For example, node `M20!` of Figures 1 and 7 may be referred to as $\texttt{M20!} : \{\langle \texttt{member}, \{\texttt{B5}, \texttt{B6}\}\rangle, \langle \texttt{class}, \{\texttt{M19} : \{\langle \texttt{lex}, \{\texttt{person}\}\rangle\}\}\rangle\}$.

SNePS 3 knowledge engineers may create any case frames they desire to form atomic closed molecular nodes. The syntax of non-atomic, variable, and restriction nodes, however, are fixed.

Non-atomic molecular nodes are used to represent rules used for node-based inference. For example, node `M88!` of Figure 5 is an and-entailment, the case frame for which is

$$\langle \texttt{rule}, \{\langle \&\texttt{ant}, \texttt{proposition}, \texttt{expand}, 1\rangle, \langle \texttt{cq}, \texttt{proposition}, \texttt{reduce}, 1\rangle\}\rangle$$

See [2, Chap. 3] for the syntax of the other non-atomic nodes.

A universal variable node is an instance of one of the case-frame schemata

$$\langle c, \{\langle \texttt{any}, \textit{restriction}, \texttt{expand}, 1\rangle\}\rangle$$

and an existential variable node is an instance of one of the case frame schemata

$$\langle c, \{\langle \texttt{some}, \textit{restriction}, \texttt{none}, 1\rangle, \langle \texttt{depends}, \textit{universal}, \texttt{none}, 0\rangle\}\rangle$$

where c is any semantic class, *restriction* is any restriction node, and *universal* is any universal variable node. Variable nodes are given identifiers of the form `Vn` by the system.

A restriction node is an instance of any case frame declared by the knowledge engineer, except that every restriction node must dominate at least one variable node. Thus, variable nodes and restriction nodes form cycles in the SNePS 3 network. Two such cycles are illustrated in Fig. 6: `V1-R1-V1` and `V1-R3-R2-V1`. Restriction nodes are given identifiers of the form `Rn` by the system.

One more relation is needed for the SNePS 3 syntax of generic propositions: $\langle \texttt{close}, \textit{universal}, \texttt{none}, 0\rangle$. Nodes from which `close` arcs emanate are taken to form the scope of universal variable nodes that the `close` arcs point to, as well as all existential variable nodes dependent on them. An example of the need for this is to distinguish *Any talking robot is not intelligent.* from *It is not the case that any talking robot is intelligent.* A `close` arc is shown in Fig. 6.

7 Summary

SNePS 3, the latest version of the SNePS family of propositional semantic networks, is a logic- and network- based knowledge representation, reasoning, and acting system, based on a paraconsistent, first-order term logic, with compositional intensional semantics.

SNePS 3 differs from earlier versions of SNePS by having the following features:

1. formal SNePS semantic classes of nodes
2. formal definition of relations

3. formal definition of case frames
4. structured variables
5. wire-based inference, including expansion as well as reduction, and limits on adjustment
6. subsumption inference

8 Availability and Use

SNePS 3 is currently being implemented. SNePS 2.5 has been implemented in ANSI Common Lisp and runs on any platform where that language is installed. SNePS 2.5 is useable for research and experimentation, and can currently handle knowledge bases on the order of about 1,000 SNePS nodes. It and other versions of SNePS are currently in use at various sites around the world, including the U.S., Portugal, Italy, and Japan.

The SNePS 2.5 source code and manual may be freely downloaded from the SNePS Research Group web pages at `http://www.cse.buffalo.edu/sneps/`, along with a tutorial, sample demonstration runs, and a bibliography.

9 Benchmarks and Comparisons

SNePS comes with a suite of demonstration problems and applications that can be used to familiarize oneself with how to use it, and may be used for comparison with other systems. Demonstrations that were taken from other sources include The Jobs Puzzle from [26, Chapter 3.2], Schubert's steamroller problem (see [27]), and a database management system example from [28].

Schubert's steamroller problem was run on a 1993 version of SNePS [29], and the results compared with those reported in [27]. The SNePS version produced fewer unifications and was faster than most unsorted logic solutions, but was outperformed by sorted logic solutions. The current version of SNePS is much faster on this problem than the 1993 version, partially due to improvements in SNePS, and partially due to faster, bigger computers.

The SNePS representation of the Jobs Puzzle is much simpler and closer to the English version of the puzzle than the clause form representation presented in [26, p. 58ff].

Acknowledgments

The design and implementation of SNePS 3 is a joint effort of the author and Syed S. Ali, Debra T. Burhans, Alistair E. Campbell, Alan M. Hunt, Haythem O. Ismail, Geoffrey D. Koplas, and other members of the SNePS Research Group, Department of Computer Science and Engineering, University at Buffalo. The author is grateful an anonymous reviewer, and to William J. Rapaport, Haythem O. Ismail, and other members of the SNePS Research Group for their comments on earlier versions of this paper.

References

1. Stuart C. Shapiro and William J. Rapaport. The SNePS family. *Computers & Mathematics with Applications*, 23(2–5):243–275, January–March 1992. Reprinted in [30, pp. 243–275].
2. Stuart C. Shapiro and The SNePS Implementation Group. *SNePS 2.5 User's Manual*. Department of Computer Science and Engineering, State University of New York at Buffalo, Buffalo, NY, 1999.
3. Syed S. Ali and Stuart C. Shapiro. Natural language processing using a propositional semantic network with structured variables. *Minds and Machines*, 3(4):421–451, November 1993.
4. Syed S. Ali. *A "Natural Logic" for Natural Language Processing and Knowledge Representation*. PhD thesis, Technical Report 94-01, Department of Computer Science, State University of New York at Buffalo, Buffalo, NY, January 1994.
5. Guy L. Steele, Jr. *Common Lisp the Language*. Digital Press, second edition, 1990.
6. Stuart C. Shapiro. The CASSIE projects: An approach to natural language competence. In J. P. Martins and E. M. Morgado, editors, *EPIA 89: 4th Portugese Conference on Artificial Intelligence Proceedings, Lecture Notes in Artificial Intelligence 390*, pages 362–380. Springer-Verlag, Berlin, 1989.
7. Stuart C. Shapiro. Embodied Cassie. In *Cognitive Robotics: Papers from the 1998 AAAI Fall Symposium, Technical Report FS-98-02*, pages 136–143. AAAI Press, Menlo Park, California, October 1998.
8. Stuart C. Shapiro and William J. Rapaport. Models and minds: Knowledge representation for natural-language competence. In Robert Cummins and John Pollock, editors, *Philosophy and AI: Essays at the Interface*, pages 215–259. MIT Press, Cambridge, MA, 1991.
9. Stuart C. Shapiro. SNePS: A logic for natural language understanding and commonsense reasoning. In Łucja Iwańska and Stuart C. Shapiro, editors, *Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language*. AAAI Press/The MIT Press, Menlo Park, CA, 2000.
10. William A. Woods. What's in a link. In Daniel G. Bobrow and Allan Collins, editors, *Representation and Understanding*, pages 35–82. Academic Press, New York, 1975. Reprinted in [31, pp. 218–241].
11. Stuart C. Shapiro. Belief spaces as sets of propositions. *Journal of Experimental and Theoretical Artificial Intelligence (JETAI)*, 5(2&3):225–235, April–September 1993.
12. Anthony S. Maida and Stuart C. Shapiro. Intensional concepts in propositional semantic networks. *Cognitive Science*, 6(4):291–330, October–December 1982. Reprinted in [31, pp. 170–189].
13. Stuart C. Shapiro. Symmetric relations, intensional individuals, and variable binding. *Proceedings of the IEEE*, 74(10):1354–1363, October 1986.
14. Stuart C. Shapiro and William J. Rapaport. SNePS considered as a fully intensional propositional semantic network. In Nick Cercone and Gordon McCalla, editors, *The Knowledge Frontier*, pages 263–315. Springer-Verlag, New York, 1987.
15. William J. Rapaport, Stuart C. Shapiro, and Janyce M. Wiebe. Quasi-indexicals and knowledge reports. *Cognitive Science*, 21(1):63–107, January–March 1997. Reprinted in [32, pp. 235–294].
16. David J. Israel. Interpreting network formalisms. In N. Cercone, editor, *Computational Linguistics*, pages 1–13. Pergammon Press, Oxford, 1983.

17. Ronald J. Brachman. On the epistemological status of semantic networks. In Nicholas V. Findler, editor, *Associative Networks: The Representation and Use of Knowledge by Computers*, pages 3–50. Academic Press, New York, 1979.
18. Michael R. Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, Los Altos, CA, 1987.
19. Deepak Kumar. The SNePS BDI architecture. *Decision Support Systems*, 16(1):3–19, January 1996.
20. Deepak Kumar and Stuart C. Shapiro. Acting in service of inference (and *vice versa*). In Douglas D. Dankel II, editor, *Proceedings of The Seventh Florida AI Research Symposium (FLAIRS 94)*, pages 207–211. The Florida AI Research Society, May 1994.
21. Haythem O. Ismail and Stuart C. Shapiro. Two problems with reasoning and acting in time. In Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR 2000)*, pages 355–365, San Francisco, 2000. Morgan Kaufmann.
22. Stuart C. Shapiro. Cables, paths and "subconscious" reasoning in propositional semantic networks. In John Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 137–156. Morgan Kaufmann, Los Altos, CA, 1991.
23. Stuart C. Shapiro. Path-based and node-based inference in semantic networks. In David L. Waltz, editor, *Tinlap-2: Theoretical Issues in Natural Languages Processing*, pages 219–225. ACM, New York, 1978.
24. Rohini Srihari. Combining path-based and node-based reasoning in SNePS. Technical Report 183, Department of Computer Science, SUNY at Buffalo, Buffalo, NY, 1981.
25. Kit Fine. A defence of arbitrary objects. *Proceedings of the Aristotelian Society*, Supp. Vol. 58:55–77, 1983.
26. L. Wos, R. Overbeek, E. Lusk, and J. Boyle. *Automated Reasoning: Introduction and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1984.
27. Mark E. Stickel. Automated deduction by theory resolution. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI'85)*, volume 2, pages 1181–1186, Los Altos, CA, 1985. Morgan Kaufmann.
28. C. J. Date. *An Introduction to Database Systems*, volume 1. Addison-Wesley, Reading, MA, 3 edition, 1981.
29. Joongmin Choi. *Experience-Based Learning in Deductive Reasoning Systems*. Technical report 93-20, Department of Computer Science, State University of New York at Buffalo, Buffalo, NY, 1993.
30. Fritz Lehmann, editor. *Semantic Networks in Artificial Intelligence*. Pergamon Press, Oxford, 1992.
31. Ronald J. Brachman and Hector J. Levesque, editors. *Readings in Knowledge Representation*. Morgan Kaufmann, San Mateo, CA, 1985.
32. Francesco Orilia and William J. Rapaport, editors. *Thought, Language, and Ontology: Essays in Memory of Hector-Neri Castañeda*. Kluwer Academic Publishers, Dordrecht, 1998.