SNePS: A Logic for Natural Language Understanding and Commonsense Reasoning
A Chapter of
Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language
Łucja Iwańska and Stuart C. Shapiro, Editors
to be published by AAAI Press/The MIT Press

Stuart C. Shapiro

May 19, 1999

Abstract

The use of logic for knowledge representation and reasoning systems is controversial. There are, indeed, several ways that standard First Order Predicate Logic is inappropriate for modelling natural language understanding and commonsense reasoning. However, a more appropriate logic can be designed. This chapter presents several aspects of such a logic.

1 Introduction

My colleagues, students, and I have been engaged in a long-term project to build a natural language using intelligent agent. While our approach to natural language understanding (NLU) and commonsense reasoning (CSR) has been logic-based, we have thought that the logics developed for metamathematics, *e.g.* (Kleene, 1950), are not the best ones for our purpose. Instead, we have designed new logics, better suited for NLU and CSR. The current version of these logics constitutes the formal language and inference mechanism of the knowledge representation/reasoning (KRR) system, SNePS 2.4 (Shapiro and The SNePS Implementation Group, 1998). SNePS is a constantly evolving system (*see* (Shapiro and Rapaport, 1992)) that implements our evolving theory of how to build a computational, natural language using, rational agent that does commonsense reasoning.

In this chapter, I will survey several ways in which the SNePS logic has been designed to be more appropriate for NLU and CSR than the standard First Order Predicate Logic (FOPL[1]). In each section, I will present a commensense reasoning problem in English. Then I will discuss the difficulties involved in representing and solving the problem in FOPL, and will show how it is represented and solved in SNePS. For some subtle problems, I will first show the SNePS solution before revealing the difficulties it presents to FOPL solutions. SNePS examples will be in SNePSLOG, [Shapiro *et al.*, 1981, Shapiro and The SNePS Implementation Group, 1998, Chapter 7] an FOPL-like user interface to SNePS.

This chapter is about SNePS logic *per se,* rather than the way we have used it for representing particular entities and relations important to NLU and CSR. For some of these discussions, see: (Peters and Shapiro, 1987; Peters et al., 1988; Peters and Rapaport, 1990) for basic- and superordinate-level categories; (Chun, 1987; Shapiro and Rapaport, 1991) for possessives; (Almeida, 1995) for tense and aspect; (Rapaport et al., 1997) for proper names and beliefs about knowledge and belief.

[1]In this chapter "FOPL" will always refer to the standard, classical, first order predicate logic, using its standard syntax.

2 What is Represented in the KR Formalism

As said above, we view our long-term project as developing a natural language using intelligent agent, who we tend to refer to as Cassie (Shapiro, 1989; Shapiro and Rapaport, 1991). At any point in Cassie's operation, the material represented in SNePS constitutes the contents of Cassie's mind (*see.* (Shapiro, 1993)). We are not interested in representing the "meaning" of words, phrases, clauses, or sentences, rather we are interested in the changes to Cassie's mind that result from her understanding natural language utterances in the context of a conversation or of reading a book or article. In the discussion that follows, I will be interested in the logic of the representation of beliefs that result from understanding utterances in certain ways. It might be that some of the sentences I cite might be understood differently in contexts other than the ones I am considering. That is beside the point. What is to the point, is my claim that there are contexts in which a natural language understander would understand the utterance in the way I suggest, and that in that case SNePS logic is more appropriate for representing that understanding than standard FOPL. Certainly a transducer is needed that can take an English utterance as input and use the entire relevant state of Cassie's mind to modify her mind to register an understanding of that utterance. That transducer, however, is not the subject of this chapter. (Though see (Shapiro, 1982; Shapiro, 1989; Shapiro and Rapaport, 1987; Neal and Shapiro, 1987; Neal and Shapiro, 1991; Neal and Shapiro, 1994; Shapiro and Rapaport, 1995).)

There are four types of expressions in SNePS logic, denoting propositions, rules, acts, and individuals. Propositions are the sorts of entities that agents can believe or disbelieve. We say that a proposition expression representing a proposition that Cassie believes is "asserted". Rule expressions are proposition expressions whose syntax is recognized by the SNePS inference routines. If the proposition `P` is asserted and the rule `P => Q` is asserted, then the SNePS inference routines may cause `Q` to be asserted. Acts are the sort of entities an agent may perform. Act expressions use a syntax recognized by SNeRE, the SNePS Rational Engine (Shapiro and The SNePS Implementation Group, 1998, Chapter 4), but are beyond the scope of this chapter (but see (Kumar, 1996; Kumar and Shapiro, 1994; Shapiro, 1998)). Individuals are everything else. However, all expressions are terms (Shapiro, 1993; Chalupsky and Shapiro, 1994), and may be used as arguments of functions.

3 Set-Oriented Logical Connectives

Consider the following problems:

1. Everything is an animal, a vegetable, or a mineral.
   (a) Squash is a vegetable.
   Is squash an animal? a mineral?
   (b) Marble is neither an animal nor a vegetable.
   Is marble a mineral?
2. For every object, the following statements are equivalent:
   - It is human
   - It is a featherless biped
   - It is a rational animal.

   (a) Socrates is human.
   Is Socrates a featherless biped? A rational animal?
   (b) Snoopy is not a featherless biped.
   Is Snoopy a rational animal? A human?

Consider formalizing problem (1). The FOPL wff

3. $$\forall x[Animal(x) \vee Vegetable(x) \vee Mineral(x)]$$

is wrong because $\vee$ is the inclusive or, and we want to be able to conclude that squash is neither an animal nor a mineral.[2] Neither is

$$\forall x[Animal(x) \oplus Vegetable(x) \oplus Mineral(x)]$$

correct, where $\oplus$ is the exclusive or, because that is satisfied by something that is an animal, a vegetable, *and* a mineral.

In English, utterances of the form "Either P_1 or ... or P_n" are often understood as saying that exactly one of $P_1, \ldots, P_n$ is true, but such an understanding is not easily formulated in FOPL. We have implemented a logical connective for this and similar problems in SNePS [Shapiro, 1979, pp. 189ff., Shapiro and Rapaport, 1992, p. 250, Shapiro and The SNePS Implementation Group, 1998, Section 3.1]. The SNePSLOG wff

$$\texttt{andor(i,j)}\{\texttt{P}_1\texttt{, ..., P}_n\}$$

is true if and only if at least `i` and at most `j` of the wffs in the set $\{\texttt{P}_1\texttt{, ..., P}_n\}$ are true. Using `andor`, problems (1) and (1a) can be solved in SNePSLOG as shown here:

```
: all(x)(andor(1,1) {animal(x), vegetable(x), mineral(x)}).
  all(X)(andor(1,1) {ANIMAL(X),VEGETABLE(X),MINERAL(X)})

: vegetable(squash)!
  VEGETABLE(SQUASH)
  ~ANIMAL(SQUASH)
  ~MINERAL(SQUASH)
```

(The ":" is the SNePSLOG prompt. Input is shown after the prompt in lower, and, occasionally, mixed case. Output is shown in all upper case (except for logical constants such as "`all`" and "`andor`"). Input terminating in a period (".") is stored and echoed, but in the rest of this chapter, the echo will not be shown to conserve space and enhance readability. The terminal "`!`" means store and perform forward inference. The output lines following inference commands report all wffs inferred and stored as a result of the inference. SNePSLOG interactions have been edited only to conserve space and to fit the format of this chapter. The character strings shown, however, are actual input and output.)

`andor(0,0)` serves as a generalized nor, so that problem (1b) can be solved as follows:

```
: andor(0,0){animal(marble), vegetable(marble)}!
  ~ANIMAL(MARBLE)
  ~VEGETABLE(MARBLE)
  MINERAL(MARBLE)
```

`andor` can also be used to represent the inclusive or. For example,

```
all(x)(andor(1,3){animal(x), vegetable(x), mineral(x)})
```

is the SNePSLOG version of the FOPL wff (3).

Novice logicians would probably try to formalize problem (2) as

$$\forall x[\text{Human}(x) \Leftrightarrow \text{Featherless-Biped}(x) \Leftrightarrow \text{Rational-Animal}(x)]$$

However, this is not correct because, for example, it is satisfied by something that is human, but neither a rational animal nor a featherless biped. The correct way to formalize (2) in FOPL is

[2] Some might argue that the "or" of problem (1) "means" inclusive or, and that we already have background knowledge that the categories of animals, vegetables, and minerals are mutually disjoint. In that case, it is that background knowledge I want to represent, and that is not captured correctly by the proffered FOPL wff.

$$\forall x[(\text{Human}(x) \Rightarrow \text{Featherless-Biped}(x))$$
$$\wedge(\text{Featherless-Biped}(x) \Rightarrow \text{Rational-Animal}(x))$$
$$\wedge(\text{Rational-Animal}(x) \Rightarrow \text{Human}(x))]$$

However, this does not capture the style of the original, which more simply asserted a relation among three propositions.

Problem (2) can be done in SNePSLOG using nested `andor`s as

```
all(x)(andor(1,1){andor(3,3){Human(x),
                             Featherless-Biped(x),
                             Rational-Animal(x)},
                  andor(0,0){Human(x),
                             Featherless-Biped(x),
                             Rational-Animal(x)}})
```

In other words, the three propositions are either all true or all false. However, this also fails to capture the simple relation among the three propositions. Therefore another connective has been included in SNePSLOG. The wff

$$\texttt{thresh(i,j)}\{\texttt{P}_1\texttt{, ..., P}_n\}$$

is true if and only if either fewer than `i` of the wffs in the set $\{\texttt{P}_1\texttt{, ..., P}_n\}$ are true or more than `j` are true. Using `thresh`, problem (2) can be solved in SNePSLOG as shown here:

```
: all(x)(thresh(1,2){human(x), featherless-biped(x), rational-animal(x)}).
: human(Socrates)!
  HUMAN(SOCRATES)
  FEATHERLESS-BIPED(SOCRATES)
  RATIONAL-ANIMAL(SOCRATES)

: ~featherless-biped(Snoopy)!
  ~RATIONAL-ANIMAL(SNOOPY)
  ~FEATHERLESS-BIPED(SNOOPY)
  ~HUMAN(SNOOPY)
```

When, in previous talks, I have suggested that "or" in English usually means exclusive or rather than inclusive or[3], one common rejoinder is that in sentences like "If Hilda is in Boston or Kathy is in Las Vegas, then Eve is in Providence" (Rips, 1983, p. 63) we would certainly not want the inference to be blocked if Hilda were in Boston and Kathy were also in Las Vegas. This is cited as evidence that the "or" in this sentence is the inclusive or. The logical form of the sentence is taken to be $(P \vee R) \Rightarrow Q$, and the steps of reasoning from P to Q are taken to be

1.	$(P \vee R) \Rightarrow Q$	*Hyp.*
2.	P	*Hyp.*
3.	$P \vee R$	$\vee$*Introduction*
4.	Q	$\Rightarrow$ *Elimination*

with the $\vee$ an inclusive or, and the rule of $\vee$ Introduction being truth-functional.

Rips (Rips, 1983), however, studied the reasoning of subjects not trained in formal logic to assess how available certain logical rules of inference were to them. He found that the rule of $\vee$ Introduction was virtually not available at all, but that instead the rule of "Disjunctive Modus Ponens"

$$\frac{P, P \vee R \Rightarrow Q}{Q}$$

[3]Notice that I am no longer making that claim—I am making no claims about the meaning of words. Rather, my claim in the previous section is about the pragmatic understanding of certain utterances.

was among the most available rules. Thus, $(P \vee R)$ is not a subformula of $P \vee R \Rightarrow Q$ whose truth value is assessed. It is as if $\_\_ \vee \_\_ \Rightarrow \_\_$ were a single propositional connective with its own rule of inference.

We have included a generalization of this connective in SNePS, and called it "or-entailment". The SNePSLOG wff

$$\{\mathtt{P}_1,\ \ldots,\ \mathtt{P}_n\}\ \mathtt{v=>}\ \{\mathtt{Q}_1,\ \ldots,\ \mathtt{Q}_m\}$$

is true if and only if $\forall i, j[\mathtt{P}_i\ \mathtt{=>}\ \mathtt{Q}_j]$. The SNePSLOG elimination rule for this connective is the appropriate generalization of Disjunctive Modus Ponens:

```
: {in(Hilda, Boston), in(Kathy, Las_Vegas)} v=> {in(Eve, Providence)}.
: in(Hilda, Boston)!

Since {IN(HILDA,BOSTON),IN(KATHY,LAS_VEGAS)} v=> {IN(EVE,PROVIDENCE)}
and IN(HILDA,BOSTON)
I infer  IN(EVE,PROVIDENCE)

  IN(HILDA,BOSTON)
  IN(EVE,PROVIDENCE)
```

In the above run, I turned the inference trace on, so the reader can see the firing of the generalized Disjunctive Modus Ponens rule.

4 The Unique Variable Binding Rule

When setting up some example in a talk, a philosophy professor said[4]

> "If someone votes for X and someone votes for Y, one of them will be disappointed"

(or something very close to that). Let us formalize our understanding of this sentence in SNePSLOG:

```
(4) all(u,v,x,y)({votesfor(u,x), votesfor(v,y)}
                 &=> {andor(1,1){disappointed(u), disappointed(v)}})
```

(Here I again interpreted "or" to mean exclusive or, and I used the SNePSLOG wff

$$\{A_1, \ldots A_n\}\ \mathtt{\&=>}\ \{C_1, \ldots, C_m\}$$

which means that the conjunction of $\{A_1, \ldots A_n\}$ implies the conjunction of $\{C_1, \ldots, C_m\}$.) To complete this example, we should note that anyone who votes for the winner is not disappointed:

```
all(u,x)({votesfor(u,x), wins(x)} &=> {~disappointed{u}})
```

Now let's try these rules in a specific example:

```
: all(u,v,x,y)({votesfor(u,x), votesfor(v,y)}
               &=> {andor(1,1){disappointed(u), disappointed(v)}}).
: all(u,x)({votesfor(u,x), wins(x)} &=> {~disappointed{u}}).
: votesfor(Hillary, Bill).
: votesfor(Elizabeth,Bob).
: wins(Bill).
: disappointed(?x)?
  DISAPPOINTED(ELIZABETH)
  ~DISAPPOINTED(HILLARY)
```

[4]Deborah Johnson, Department Colloquium, Department of Computer Science, State University of New York at Buffalo, March 17, 1994.

(Free variables in queries are indicated by a prefixed "?", which is also used as termination punctuation to start backward inference. The response to a query consists of all positive and negative instances of the query that can be derived.) The conclusion is that Elizabeth is disappointed, but Hillary isn't.

The surprising aspect of this example is that in FOPL, an instance of statement (4) is

```
{votesfor(Hillary,Bill), votesfor(Hillary,Bill)}
&=> {andor(1,1){disappointed(Hillary), disappointed(Hillary)}}
```

From which, given the specific example, `disappointed(Hillary)` follows.

The problem is that in FOPL, one is allowed to replace two universally quantified variables by the same term, but in normal understanding of NL utterances such as the above quote, it is assumed that different noun phrases in one sentence refer to different entities (unless one of the noun phrases is marked as an anaphoric reference to another). An FOPL representation of such an NL utterance usually requires a judicious inclusion of $\neq$ predicates. However, this inclusion is unintuitive, makes the formalized statement more cumbersome, and the transduction process error-prone. For example, in presenting an example of a KLONE definition of an arch, Brachman says that "any example of this type of object has two UPRIGHTs," (Brachman, 1979, p. 37) and goes on to explain the structural description, S2, by saying,

> "S2 specifies that no two UPRIGHTs touch each other" (Brachman, 1979, p. 37)

but in the actual figure being described, the FOPL sentence attached to S2 is

$$\forall X \in \text{UPRIGHT}(\exists Y \in \text{UPRIGHT}. \sim \text{TOUCH}(X, Y))$$

and this can be satisfied by two touching uprights neither of which touches itself.

Our approach to this issue has been to modify the rule of Universal Instantiation so that two variables in one wff cannot be replaced by the same term. This restriction is called the "Unique Variable Binding Rule," or UVBR (Shapiro, 1986). It was UVBR that allowed statement (4) to be the formalization of our understanding of the "disappointed" quote.

5 Set Arguments

Consider the statement, "Mary, Sue, and Sally are sisters." The usual way to formalize this in FOPL would be

$$sisters(Mary, Sue) \wedge sisters(Sue, Sally)$$

along with statements that *sisters* is symmetric

$$\forall(x, y)[sisters(x, y) \Leftrightarrow sisters(y, x)]$$

and almost transitive

$$\forall(x, z)[x \neq z \Rightarrow (\exists(y)[sisters(x, y) \wedge sisters(y, z)] \Rightarrow sisters(x, z))]$$

Because of the cumbersomeness of this formalization compared with the English statement, we have introduced *set arguments* into SNePS (Shapiro, 1986). If P is an m-ary predicate, and τ is a set whose members are of the appropriate type for the i^{th} argument of P, then the following two inference rules are included in SNePS logic.

$$\frac{P(s_1, \ldots, s_{i-1}, \tau, s_{i+1}, \ldots, s_m), \tau' \subset \tau}{P(s_1, \ldots, s_{i-1}, \tau', s_{i+1}, \ldots, s_m)} \qquad \frac{P(s_1, \ldots, s_{i-1}, \tau, s_{i+1}, \ldots, s_m), t \in \tau}{P(s_1, \ldots, s_{i-1}, t, s_{i+1}, \ldots, s_m)}$$

(These are versions of what we have called "reduction inference" [Shapiro 1991, Shapiro and The SNePS Implementation Group, 1998, Chapter 2.5.1].) Thus `sisters({Mary,Sue,Sally})` implies `sisters({Mary,Sue})`, `sisters({Mary,Sally})`, and `sisters({Sue,Sally})` (as well as the admittedly peculiar `sisters(Mary)`, `sisters(Sue)`, and `sisters(Sally)`[5]).

The usefulness of set arguments (combined with UVBR) may be seen in an inference from "Mary, Sue, and Sally are sisters" and "Sisters like each other":

[5] A method of restricting such implications is included in SNePS 3, currently being implemented.

```
: sisters({Mary, Sue, Sally}).
: all(x,y)(sisters({x,y}) => {likes(x,y), likes(y,x)}).
: likes(?x,?y)?
  LIKES(SUE,MARY)
  LIKES(MARY,SUE)
  LIKES(SALLY,SUE)
  LIKES(SUE,SALLY)
  LIKES(MARY,SALLY)
  LIKES(SALLY,MARY)
```

Notice that not only are all six combinations found, but the three instances of `likes(x,x)` are avoided due to UVBR.

6 "Higher-Order" Logic

If a relation, R, is transitive, then whenever any x is in the R relation to some y, and y is also in the R relation to some z, then x is in the R relation to z. That statement is not expressible in FOPL, because it requires quantifying over predicates. Nevertheless, it is useful, so we have allowed users of SNePS to express themselves in higher-order logic (Shapiro et al., 1981):

```
: all(R)(Transitive(R) => all(x,y,z)({R(x,y), R(y,z)} &=> {R(x,z)})).
: Transitive(bigger).
: bigger(elephant,lion).
: bigger(lion,mouse).
: bigger(elephant,mouse)?
  BIGGER(ELEPHANT,MOUSE)
```

It is really only the user language that is higher-order. The representation formalism is only first order. User-language predications such as `bigger(elephant,lion)` are stored using a variety of the "*Holds*" predicate, such as, $Holds(bigger, elephant, lion)$. Thus, the rule about transitive relations is really stored looking more like

$$\forall(R)[Transitive(R) \\ \quad \Rightarrow \forall(x,y,z)[Holds(R,x,y) \wedge Holds(R,y,z)] \Rightarrow Holds(R,x,z)]$$

than like a higher-order rule. Nevertheless, the ability to express rules in a higher-order language is very useful.

Another aspect of higher-order logic is the ability to quantify over formulas. Consider the argument

Anything Bob believes is true.
Bob believes anything Bill believes.
Bill believes whatever Kevin's favorite proposition is.
Kevin's favorite proposition is that John is taller than Mary.
Therefore John is taller than Mary.

SNePS wffs such as `Taller(John, Mary)` are not actually sentences, but functional terms that denote propositions (Shapiro, 1993; Chalupsky and Shapiro, 1994). Therefore, using them as arguments and quantifying over them does not take us out of first order logic. Here is this example in SNePSLOG:

```
: all(p)(Believes(Bob, p) => p).
: all(p)(Believes(Bill, p) => Believes(Bob, p)).
: all(p)(Favorite-proposition(Kevin, p) => Believes(Bill, p)).
: Favorite-proposition(Kevin, Taller(John, Mary)).
: Taller(John, Mary)?
  TALLER(JOHN,MARY)
```

Notice that "higher-order" is in quotes in the heading of this section because, while the SNePSLOG wffs in this section look like higher-order formulas, the underlying SNePS logic is really first order.

7 Intensional Representation

Natural language sentences contain what are known as *opaque contexts,* in which one denoting phrase cannot necessarily be substituted for another even though they denote the same object. An example due to Russell is, "George IV wished to know whether Scott was the author of *Waverly*" (Russell, 1906, p. 108) because *Waverly* was published anonymously. One obviously cannot replace "the author of *Waverly*" by "Scott" in that sentence even though Scott was, in fact the author of *Waverly.* Verbs such as "know" and "believe" put their complements in opaque contexts. The standard terminology is that the denoting phrases "Scott" and "the author of *Waverly*" denote different *intensions,* but the same *extension.* In SNePS, all terms represent intensions only (Maida and Shapiro, 1982; Shapiro and Rapaport, 1987), and the entire SNePS network is considered to be an opaque context. Thus, there is no built-in equality predicate in SNePS because no two terms are taken as denoting the same entity (the *Uniqueness Principle).*

An example from the AI literature is due to McCarthy (McCarthy, 1979):

> the meaning of the phrase *"Mike's telephone number"* in the sentence *"Pat knows Mike's telephone number"* is the concept of Mike's telephone number, whereas its meaning in the sentence *"Pat dialled Mike's telephone number"* is the number itself. Thus if we also have *"Mary's telephone number = Mike's telephone number,"* then *"Pat dialled Mary's telephone number"* follows, but *"Pat knows Mary's telephone number"* does not. (McCarthy, 1979, p. 129–130, italics in the original).

Notice that "knows" creates an opaque context, whereas "dials" does not, so McCarthy is making the same point as above—"Mary's telephone number" cannot replace "Mike's telephone number" in the sentence "Pat knows Mike's telephone number", even though they have the same extension, but it can in the sentence "Pat dialled Mary's telephone number."

Although there is no built-in equality predicate in SNePS, we can introduce one to mean that two entities have the same extension[6], and explicitly specify which contexts are not opaque. A SNePSLOG example of applying this technique to McCarthy's telephone problem is:

```
: all(R)(Transparent(R) => all(a,x,y)({R(a,x), =({x,y})} &=> {R(a,y)})).
: Transparent(Dial).
: =({Telephone(Mike), Telephone(Mary)}).
: Know(Pat, Telephone(Mike)).
: Dial(Pat, Telephone(Mike)).
: ?what(Pat, ?which)?
  DIAL(PAT,TELEPHONE(MARY))
  KNOW(PAT,TELEPHONE(MIKE))
  DIAL(PAT,TELEPHONE(MIKE))
```

(Note the use of set arguments in the `=` predicate, and the use of a second order query.)

8 The Numerical Quantifiers

Consider the following problems:

1. No one has more than one mother.
   Jane is John's mother.
   Is Mary John's mother?

2. The committee members are Chris, Leslie, Pat, and Stevie.
   At least two members of the committee are women.
   Leslie and Stevie are men.
   Is Pat a man or a woman?

[6] This predicate has been called `EQUIV` in previous papers.

To facilitate such "reasoning by the process of elimination," SNePS logic includes the numerical quantifiers (Shapiro, 1979a).

$$\mathtt{nexists(i,j,k)(x)(\{P_1(x),\ldots,P_n(x)\} : \{Q(x)\})}$$

means that there are k individuals that satisfy $\mathtt{P_1(x)} \wedge \cdots \wedge \mathtt{P_n(x)}$, and, of them, at least i and at most j also satisfy Q(x). There are two elimination rules of inference for the numerical quantifiers:

1. If j individuals are known that satisfy $\mathtt{P_1(x)} \wedge \cdots \wedge \mathtt{P_n(x)} \wedge \mathtt{Q(x)}$ then it may be inferred that every other individual that satisfies $\mathtt{P_1(x)} \wedge \cdots \wedge \mathtt{P_n(x)}$ also satisfies ~Q(x).

2. If k-i individuals are known that satisfy $\mathtt{P_1(x)} \wedge \cdots \wedge \mathtt{P_n(x)} \wedge$ ~Q(x) then it may be inferred that every other individual that satisfies $\mathtt{P_1(x)} \wedge \cdots \wedge \mathtt{P_n(x)}$ also satisfies Q(x).

If one expects to use only one of these rules of inference, one may use either abbreviated form nexists(_, j, _) or nexists(i, _, k).

The following interaction shows the above two problems solved in SNePSLOG.

```
: all(x)(Person(x) => nexists(_,1,_)(y)({Person(y)}: {Mother(y,x)})).
: Person({John, Jane, Mary}).
: Mother(Jane, John).
: Mother(Mary, John)?
  ~MOTHER(MARY,JOHN)

: Member({Chris, Leslie, Pat, Stevie}).
: nexists(2,_,4)(x)({Member(x)}: {Woman(x)}).
: all(x)(Member(x) => andor(1,1){Man(x), Woman(x)}).
: Man({Leslie, Stevie}).
: ?What(Pat)?
  WOMAN(PAT)
  MEMBER(PAT)
  ~MAN(PAT)
```

Since the numerical quantifier elimination rules count individuals, we had to decide how to count. The current version of SNePS counts each distinct term as different. We are considering the possibility that in future versions of SNePS, co-extensional terms will be counted only once.

9 Contexts

When doing NLU and CSR, some kind of context mechanism is needed to keep different domains separate. For example, when reading a fictional narrative, it is important to be able to use background knowledge from the real world to understand the narrative, while accepting fictional information even if it contradicts real-world beliefs (*see* (Rapaport and Shapiro, 1995)).

SNePS contains a context mechanism as part of its assumption-based belief revision system (*see* Section 10 below). SNePS contexts are defined both extensionally and intensionally. Extensionally, a context is a set of assumptions or hypotheses—propositions asserted to Cassie by the user. A new hypothesis cannot be added to an existing context, extensionally defined, just as a new element cannot be added to a set, extensionally defined. Intensionally, a context is a named structure that contains a set of assumptions (referred to as "assertions" in the sample run below). A new assumption can be added to a context, intensionally defined.

Some SNePSLOG commands take an intensional context as an optional argument. Otherwise, all SNePSLOG commands are taken as referring to the default context, which by default is the one named DEFAULT-DEFAULTCT. The following run demonstrates reasoning in two overlapping contexts, the real-world and the world of mythology.

```
: ;;; Create the real-world context with an empty set of assertions.
set-context real-world ()
: ;;; Make real-world the default context.
set-default-context real-world
: ;;; Animals are partitioned into birds and beasts.
all(x)({Bird(x), Beast(x)} v=> {Animal(x)}).
: all(x)(Animal(x) => andor(1,1){Bird(x), Beast(x)}).
: ;;; Horses are beasts.
all(x)(Horse(x) => Beast(x)).
: ;;; Pegasus is a horse.
Horse(Pegasus).
: ;;; A rider travels by ground or air depending on whether what (s)he is riding has wings.
all(x,y)(Rides(x,y) => andor(1,1){Travelsby(x, air), Travelsby(x, ground)}).
: all(x,y)(Rides(x,y) => thresh(1,1){Winged(y), Travelsby(x, air)}).
: ;;; Bellerophon rides Pegasus.
Rides(Bellerophon, Pegasus).
: ;;; Show the context.  (The set of restrictions is used by belief revision.)
describe-context
((ASSERTIONS (WFF1 WFF2 WFF3 WFF4 WFF5 WFF6 WFF7)) (RESTRICTION NIL)
 (NAMED (REAL-WORLD)))

: ;;; Create the mythology context, initialized to agree with the real-world.
set-context mythology (WFF1 WFF2 WFF3 WFF4 WFF5 WFF6 WFF7)
: ;;; Still in the real world, birds and only birds have wings.
all(x)(Winged(x) <=> Bird(x)).
: ;;; How does Bellerophon travel?
Travelsby(Bellerophon,?what)?
  TRAVELSBY(BELLEROPHON,GROUND)
  ~TRAVELSBY(BELLEROPHON,AIR)

: ;;; Change the default context to be mythology.
set-default-context mythology
: ;;; Here, Pegasus has wings.
Winged(Pegasus).
: ;;; How does Bellerophon travel here?
Travelsby(Bellerophon,?what)?
  ~TRAVELSBY(BELLEROPHON,GROUND)
  TRAVELSBY(BELLEROPHON,AIR)

: ;;; List all beliefs in the mythology context.
list-asserted-wffs mythology
  all(X)({BIRD(X),BEAST(X)} v=> {ANIMAL(X)})
  andor(1,1){TRAVELSBY(BELLEROPHON,GROUND),TRAVELSBY(BELLEROPHON,AIR)}
  WINGED(PEGASUS)
  WINGED(PEGASUS) <=> TRAVELSBY(BELLEROPHON,AIR)
  ANIMAL(PEGASUS)
  BEAST(PEGASUS)
  andor(1,1){BIRD(PEGASUS),BEAST(PEGASUS)}
  ~BIRD(PEGASUS)
  all(X)(ANIMAL(X) => (andor(1,1){BIRD(X),BEAST(X)}))
  ~TRAVELSBY(BELLEROPHON,GROUND)
  all(X)(HORSE(X) => BEAST(X))
  HORSE(PEGASUS)
```

```
  all(X,Y)(RIDES(X,Y) => (andor(1,1){TRAVELSBY(X,AIR),TRAVELSBY(X,GROUND)}))
  all(X,Y)(RIDES(X,Y) => (WINGED(Y) <=> TRAVELSBY(X,AIR)))
  RIDES(BELLEROPHON,PEGASUS)
  TRAVELSBY(BELLEROPHON,AIR)

: ;;; List all beliefs in the real-world context.
list-asserted-wffs real-world
  all(X)({BIRD(X),BEAST(X)} v=> {ANIMAL(X)})
  TRAVELSBY(BELLEROPHON,GROUND)
  andor(1,1){TRAVELSBY(BELLEROPHON,GROUND),TRAVELSBY(BELLEROPHON,AIR)}
  WINGED(PEGASUS) <=> TRAVELSBY(BELLEROPHON,AIR)
  ANIMAL(PEGASUS)
  BEAST(PEGASUS)
  andor(1,1){BIRD(PEGASUS),BEAST(PEGASUS)}
  ~BIRD(PEGASUS)
  ~WINGED(PEGASUS)
  all(X)(ANIMAL(X) => (andor(1,1){BIRD(X),BEAST(X)}))
  ~TRAVELSBY(BELLEROPHON,AIR)
  all(X)(HORSE(X) => BEAST(X))
  HORSE(PEGASUS)
  all(X,Y)(RIDES(X,Y) => (andor(1,1){TRAVELSBY(X,AIR),TRAVELSBY(X,GROUND)}))
  all(X,Y)(RIDES(X,Y) => (WINGED(Y) <=> TRAVELSBY(X,AIR)))
  RIDES(BELLEROPHON,PEGASUS)
  all(X)(WINGED(X) <=> BIRD(X))
```

Notice that Cassie believes that Bellerophon travels through the air in the world of mythology, but on the ground in the real world.

10 Belief Revision

AI systems that get their input from normal people (as opposed to programmers or knowledge engineers) will certainly occasionally get contradictory information. To deal with this, the system needs two facilities:

1. The ability to recognize and trap explicit contradictions so that something can be done about them.
2. The ability to retract stored information inferred from information that is later retracted.

SNePS 2.4 includes SNeBR (Martins and Shapiro, 1988), a Belief Revision system that has these two abilities.

When some proposition is entered or inferred that directly contradicts one that is already stored, SNeBR opens a dialogue with the user:

```
: all(x)(Bird(x) => Flies(x)).
: all(x)(Penguin(x) => Bird(x)).
: all(x)(Penguin(x) => ~Flies(x)).
: Bird(Opus)!
  BIRD(OPUS)
  FLIES(OPUS)

: Penguin(Opus)!
  A contradiction was detected within context DEFAULT-DEFAULTCT.
  The contradiction involves the newly derived proposition:
      ~FLIES(OPUS)
   and the previously existing proposition:
      FLIES(OPUS)
  You have the following options:
```

```
   1. [C]ontinue anyway, knowing that a contradiction is derivable;
   2. [R]e-start the exact same run in a different context which is not inconsistent;
   3. [D]rop the run altogether.
  (please type c, r or d)
=><= ...
```

If the user chooses option (2), the system will help her to identify and remove the proposition(s) that caused the contradiction.

The system keeps track of the hypotheses that underly inferred propositions. (An hypothesis is a proposition that was told to the system, as opposed to one that the system inferred.) So if an hypothesis is retracted, the system retracts every inferred proposition that was derived from it:

```
: all(x)(Bird(x) => Flies(x)).
: all(x)(Flies(x) => Feathered(x)).
: all(x)(Canary(x) => Bird(x)).
: Canary(Tweety)!
  CANARY(TWEETY)
  BIRD(TWEETY)
  FLIES(TWEETY)
  FEATHERED(TWEETY)

: Canary(Clyde)!
  FLIES(CLYDE)
  FEATHERED(CLYDE)
  CANARY(CLYDE)
  BIRD(CLYDE)

: list-asserted-wffs
  all(X)(BIRD(X) => FLIES(X))
  FLIES(CLYDE)
  FEATHERED(CLYDE)
  all(X)(FLIES(X) => FEATHERED(X))
  all(X)(CANARY(X) => BIRD(X))
  CANARY(TWEETY)
  BIRD(TWEETY)
  FLIES(TWEETY)
  FEATHERED(TWEETY)
  CANARY(CLYDE)
  BIRD(CLYDE)

: ~Feathered(Clyde)!
  A contradiction was detected within context DEFAULT-DEFAULTCT.
  The contradiction involves the proposition you want to assert:
      ~FEATHERED(CLYDE)
  and the previously existing proposition:
      FEATHERED(CLYDE)
  You have the following options:
   1. [c] to continue anyway, knowing that a contradiction is derivable;
   2. [r] to revise the inconsistent part of the context
   3. [d] to discard this contradictory new assertion from the context
  (please type c, r or d)
=><= r
```

```
  In order to make the context consistent you must delete at least
  one hypothesis from each of the following sets of hypotheses:
    (WFF1 WFF12 WFF2 WFF3 WFF8)
 In order to make the context consistent you must delete
  at least one hypothesis from the set listed below.
An inconsistent set of hypotheses:
 1 :   all(X)(BIRD(X) => FLIES(X))
          (5 supported propositions: (WFF1 WFF10 WFF11 WFF6 WFF7) )
 2 :   ~FEATHERED(CLYDE)
          (1 supported proposition: (WFF12) )
 3 :   all(X)(FLIES(X) => FEATHERED(X))
          (3 supported propositions: (WFF11 WFF2 WFF7) )
 4 :   all(X)(CANARY(X) => BIRD(X))
          (7 supported propositions: (WFF10 WFF11 WFF3 WFF5 WFF6 WFF7 WFF9) )
 5 :   CANARY(CLYDE)
          (4 supported propositions: (WFF10 WFF11 WFF8 WFF9) )
  Enter the list number of a hypothesis to examine or
  [d] to discard some hypothesis from this list,
  [a] to see ALL the hypotheses in the full context,
  [r] to see what you have already removed,
  [q] to quit revising this set, or
  [i] for instructions
  (please type a number OR d, a, r, q or i)
=><= d
  Enter the list number of a hypothesis to discard,
  [c] to cancel this discard, or [q] to quit revising this set.
=><= 5

The consistent set of hypotheses:
 1 :   all(X)(BIRD(X) => FLIES(X))
          (5 supported propositions: (WFF1 WFF10 WFF11 WFF6 WFF7) )
 2 :   ~FEATHERED(CLYDE)
          (1 supported proposition: (WFF12) )
 3 :   all(X)(FLIES(X) => FEATHERED(X))
          (3 supported propositions: (WFF11 WFF2 WFF7) )
 4 :   all(X)(CANARY(X) => BIRD(X))
          (7 supported propositions: (WFF10 WFF11 WFF3 WFF5 WFF6 WFF7 WFF9) )
  Enter the list number of a hypothesis to examine or
  [d] to discard some hypothesis from this list,
  [a] to see ALL the hypotheses in the full context,
  [r] to see what you have already removed,
  [q] to quit revising this set, or
  [i] for instructions
  (please type a number OR d, a, r, q or i)
=><= q

  The following (not known to be inconsistent) set of
  hypotheses was also part of the context where the
  contradiction was derived:
    (WFF4)
  Do you want to inspect or discard some of them?
=><= no
```

```
  Do you want to add a new hypothesis?
=><= no
  ~FEATHERED(CLYDE)

: list-asserted-wffs
  all(X)(BIRD(X) => FLIES(X))
  ~FEATHERED(CLYDE)
  ~CANARY(CLYDE)
  all(X)(FLIES(X) => FEATHERED(X))
  all(X)(CANARY(X) => BIRD(X))
  CANARY(TWEETY)
  BIRD(TWEETY)
  FLIES(TWEETY)
  FEATHERED(TWEETY)
```

Notice that after retracting `Canary(Clyde)`, the propositions that were inferred from it, `FLIES(CLYDE)`, `BIRD(CLYDE)`, and `FEATHERED(CLYDE)` and were also removed.

SNeBR tells the user how many propositions are supported by each hypothesis in case she wants to remove the hypothesis that makes a minimal change to the knowledge base (Alchourrón et al., 1985). However, since we assume that there is more to be learned, this may not be the definitive criterion, and, in fact, in this example, the removed hypothesis was not the one that made the minimal change.

11 Relevance Logic

In FOPL, a contradiction implies anything whatsoever, but most people would say that just because you believe that Opus does and doesn't fly, that's no reason to believe something totally unrelated to Opus and flying, such as that the Earth is flat. SNePS logic is a version of Relevance Logic (Anderson and Belnap, 1975; Anderson et al., 1992; Shapiro, 1992), a "paraconsistent" logic in which the so-called "paradoxes of implication" such as $(A \wedge \neg A) \Rightarrow B$, are not valid.

```
: all(x)(Flies(x) => Feathered(x)).
: all(x)(~Flies(x) => Swims(x)).
: Flies(Opus).
: ~Flies(Opus).
  A contradiction was detected within context DEFAULT-DEFAULTCT.
  The contradiction involves the proposition you want to assert:
      ~FLIES(OPUS)
  and the previously existing proposition:
      FLIES(OPUS)
  You have the following options:
   1. [c] to continue anyway, knowing that a contradiction is derivable;
   2. [r] to revise the inconsistent part of the context
   3. [d] to discard this contradictory new assertion from the context
  (please type c, r or d)
=><= c
  ~FLIES(OPUS)

: Feathered(Opus)?
  A contradiction was detected within context DEFAULT-DEFAULTCT.
  The contradiction involves the newly derived proposition:
      FLIES(OPUS)
  and the previously existing proposition:
      ~FLIES(OPUS)
```

```
  You have the following options:
   1. [C]ontinue anyway, knowing that a contradiction is derivable;
   2. [R]e-start the exact same run in a different context which is not inconsistent;
   3. [D]rop the run altogether.
  (please type c, r or d)
=><= c
  FEATHERED(OPUS)

: Swims(Opus)?
  A contradiction was detected within context DEFAULT-DEFAULTCT.
  The contradiction involves the newly derived proposition:
      ~FLIES(OPUS)
  and the previously existing proposition:
      FLIES(OPUS)
  You have the following options:
   1. [C]ontinue anyway, knowing that a contradiction is derivable;
   2. [R]e-start the exact same run in a different context which is not inconsistent;
   3. [D]rop the run altogether.
  (please type c, r or d)
=><= c
  SWIMS(OPUS)

: Flat(Earth)?

: list-asserted-wffs
  all(X)(FLIES(X) => FEATHERED(X))
  all(X)((~FLIES(X)) => SWIMS(X))
  FLIES(OPUS)
  ~FLIES(OPUS)
  FEATHERED(OPUS)
  SWIMS(OPUS)
```

(Remember that when a question A`?` is asked, if A can be derived from the stored information, it is printed, and if ~A can be derived, *it* is printed. If neither can be derived, nothing is printed, which is the case here, indicated by nothing being shown between the query, `Flat(Earth)?`, and the next prompt.)

So the contradiction allows the system to infer related contradictory information, specifically `SWIMS(OPUS)` and `FEATHERED(OPUS)`, but not irrelevant information such as `Flat(Earth)`.

Another paradox of implication is that anything whatsoever implies a truth, $A \Rightarrow (B \Rightarrow A)$. First notice that SNePS can derive implications:

```
: all(x)(Canary(x) => Bird(x)).
: all(x)(Bird(x) => Flies(x)).
: Canary(Tweety) => Flies(Tweety)?
  CANARY(TWEETY) => FLIES(TWEETY)
```

Now, let's try $A \Rightarrow (B \Rightarrow A)$:

```
: Penguin(Opus).
: Canary(Tweety) => Penguin(Opus)?
:
```

The implication is not derived.

12 Circular and Recursive Rules

Above I said that normal people occasionally give contradictory information. They also tend to give circular definitions, which get formalized as recursive rules. The SNePS inference mechanism was designed to work without getting into infinite loops in the face of recursive rules without regard to: the order of entry of rules or ground propositions; the order of predicates within rules; whether recursive rules are left- or right-recursive, or both; what predicates are used in ground propositions. (Shapiro and McKay, 1980; McKay and Shapiro, 1981) An example of using a circular definition is

```
: all(x,y)(thresh(1,1){North-of(x,y), South-of(y,x)}).
: North-of(Seattle, Portland).
: South-of(San_Francisco, Portland).
: North-of(San_Francisco, Los_Angeles).
: South-of(San_Diego, Los_Angeles).
: North-of(?x, ?y)?
  NORTH-OF(SEATTLE,PORTLAND)
  NORTH-OF(SAN_FRANCISCO,LOS_ANGELES)
  NORTH-OF(LOS_ANGELES,SAN_DIEGO)
  NORTH-OF(PORTLAND,SAN_FRANCISCO)
```

A more traditional example of a recursive rule is

```
: all(x,y)(parent(x,y) => ancestor(x,y)).
: all(x,y,z)({ancestor(x,y), ancestor(y,z)} &=> {ancestor(x,z)}).
: parent(John, Mary).
: ancestor(Mary, George).
: ancestor(George, Sally).
: parent(Sally, Jimmy).
: ancestor(John, ?y)?
  ANCESTOR(JOHN,JIMMY)
  ANCESTOR(JOHN,MARY)
  ANCESTOR(JOHN,GEORGE)
  ANCESTOR(JOHN,SALLY)

: ancestor(?x, Jimmy)?
  ANCESTOR(SALLY,JIMMY)
  ANCESTOR(JOHN,JIMMY)
  ANCESTOR(GEORGE,JIMMY)
  ANCESTOR(MARY,JIMMY)
```

Of course, SNePS will infinitely loop if it is asked to forward chain through a rule of the form $\forall(x)[P(x) \Rightarrow P(f(x))]$ or back-chain through one of the form $\forall(x)[P(f(x)) \Rightarrow P(x)]$.

13 Summary

SNePS has been and is being designed to be a KRR system for a computerized natural language using, commonsense reasoning rational agent. SNePS is founded on logic, but on a logic that has been (and is being) designed specifically to support natural language processing and commonsense reasoning. Several aspects of this logic have been summarized in this chapter. We may categorize them as follows.

- Those that differ from FOPL in syntax (with appropriate semantics and inference rules):
  - Set Arguments
  - Set-Oriented Logical Connectives
  - Numerical Quantifiers

  - "Higher-Order" Logic

- Those that differ from FOPL in some inference rule(s) (with appropriate semantics):
  - The Unique Variable Binding Rule
  - Relevance Logic

- Those that differ from FOPL only in semantics:
  - Intensional Representation

- Those that rely on an appropriate inference mechanism:
  - Contexts
  - Belief Revision
  - Use of circular and recursive rules

More recent developments, that have not yet been incorporated with the other features of SNePS 2.4, are "structured variables" (Ali, 1993; Ali and Shapiro, 1993; Ali, 1994) and using simulative reasoning to reason about the beliefs of other agents (Chalupsky and Shapiro, 1994; Chalupsky, 1996; Chalupsky and Shapiro, 1996).

14 Acknowledgments

This chapter is a revised version of Stuart C. Shapiro, Formalizing English, *International Journal of Expert Systems 9,* 1 (1996), 151–171(Shapiro, 1996). Previous versions were presented at the Third Golden West International Conference on Intelligent Systems (GWIC), Las Vegas, NV, June 6–8, 1994, and the AAAI Fall Symposium on Knowledge Representation for Natural Language Processing in Implemented Systems, November, 1994. Developments described in this chapter were carried out in conjunction with William J. Rapaport, Donald P. Mckay, João P. Martins, and other members of the SNePS Research Group of the Department of Computer Science and Engineering, State University of New York at Buffalo (UB). The SNeBR dialogue shown here was programmed by Frances L. Johnson as a modification of a previous version. The author appreciates the help provided by Rohini Srihari and Jean-Pierre Koenig, the comments of Syed Ali and Łucja Iwańska on earlier drafts of this chapter, and the general intellectual support of UB's Center for Cognitive Science and its Research Group on Discourse and Narrative. The work reported here was supported in part by Equipment Grant No. EDUD-US-932022 from SUN Microsystems Computer Corporation, in part by NASA under contract NAS 9-19335, and in part by the U.S. Army CECOM through a contract with CACI Technologies.

References

Alchourrón, C. E., Gärdenfors, P., and Makinson, D. (1985). On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic*, 50(2):510–530.

Ali, S. S. (1993). A structured representation for noun phrases and anaphora. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pages 197–202, Hillsdale, NJ. Lawrence Erlbaum.

Ali, S. S. (1994). *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.

Ali, S. S. and Shapiro, S. C. (1993). Natural language processing using a propositional semantic network with structured variables. *Minds and Machines*, 3(4):421–451.

Almeida, M. J. (1995). Time in narratives. In Duchan, J. F., Bruder, G. A., and Hewitt, L. E., editors, *Deixis in Narrative: A Cognitive Science Perspective*, pages 159–189. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ.

Anderson, A. R. and Belnap, Jr., N. D. (1975). *Entailment*, volume I. Princeton University Press, Princeton.

Anderson, A. R., Belnap, Jr., N. D., and Dunn, M. (1992). *Entailment*, volume II. Princeton University Press, Princeton.

Brachman, R. J. (1979). On the epistemological status of semantic networks. In Findler, N. V., editor, *Associative Networks: The Representation and Use of Knowledge by Computers*, pages 3–50. Academic Press, New York.

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

Chalupsky, H. (1996). *SIMBA: Belief Ascription by Way of Simulative Reasoning*. Ph.D. dissertation, Technical Report 96-18, Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY.

Chalupsky, H. and Shapiro, S. C. (1994). SL: A subjective, intensional logic of belief. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, Hillsdale, NJ. Lawrence Erlbaum.

Chalupsky, H. and Shapiro, S. C. (1996). Reasoning about incomplete agents. In *Proc., 5th Int'l. Conf. on User Modelling*.

Chun, S. A. (1987). SNePS implementation of possessive phrases. SNeRG Technical Note 19, Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY.

Feigl, H. and Sellars, W., editors (1949). *Readings in Philosophical Analysis*. Appleton-Century-Crofts, New York.

Kleene, S. C. (1950). *Introduction to Metamathematics*. D. Van Nostrand, Princeton, NJ.

Kumar, D. (1996). The SNePS BDI architecture. *Decision Support Systems*, 16:3.

Kumar, D. and Shapiro, S. C. (1994). Acting in service of inference (and *vice versa*). In Dankel II, D. D., editor, *Proceedings of The Seventh Florida AI Research Symposium (FLAIRS 94)*, pages 207–211. The Florida AI Research Society.

Lehmann, F., editor (1992). *Semantic Networks in Artificial Intelligence*. Pergamon Press, Oxford.

Maida, A. S. and Shapiro, S. C. (1982). Intensional concepts in propositional semantic networks. *Cognitive Science*, 6(4):291–330. Reprinted in (Brachman and Levesque, 1985, pp. 170–189).

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

McCarthy, J. (1979). First order theories of individual concepts and propositions. In Hayes, J. E., Michie, D., and Mikulich, L. I., editors, *Machine Intelligence 9*, pages 129–147. Ellis Horwood Limited, Chichester, England. Reprinted in (Brachman and Levesque, 1985, pp. 524–533).

McKay, D. P. and Shapiro, S. C. (1981). Using active connection graphs for reasoning with recursive rules. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 368–374. Morgan Kaufmann, San Mateo, CA.

Neal, J. G. and Shapiro, S. C. (1987). Knowledge-based parsing. In Bolc, L., editor, *Natural Language Parsing Systems*, pages 49–92. Springer-Verlag, Berlin.

Neal, J. G. and Shapiro, S. C. (1991). Intelligent multi-media interface technology. In Sullivan, J. W. and Tyler, S. W., editors, *Intelligent User Interfaces*, pages 11–43. Addison-Wesley, Reading, MA.

Neal, J. G. and Shapiro, S. C. (1994). Knowledge-based multimedia systems. In Buford, J. F. K., editor, *Multimedia Systems*, pages 403–438. ACM Press/Addison Wesley, Reading, MA.

Peters, S. L. and Rapaport, W. J. (1990). Superordinate and basic level categories in discourse: Memory and context. In *Proceedings of the Twelvth Annual Conference of the Cognitive Science Society*. Lawrence Erlbaum Associates, Hillsdale, NJ.

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

Peters, S. L., Shapiro, S. C., and Rapaport, W. J. (1988). Flexible natural language processing and Roschian category theory. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, pages 125–131. Lawrence Erlbaum Associates, Hillsdale, NJ.

Rapaport, W. J. and Shapiro, S. C. (1995). Cognition and fiction. In Duchan, J. F., Bruder, G. A., and Hewitt, L. E., editors, *Deixis in Narrative: A Cognitive Science Perspective*, pages 107–128. Lawrence Erlbaum Associates, Hillsdale, NJ.

Rapaport, W. J., Shapiro, S. C., and Wiebe, J. M. (1997). Quasi-indexicals and knowledge reports. *Cognitive Science*, 21:63–107.

Rips, L. J. (1983). Cognitive processes in propositional reasoning. *Psychological Review*, 90(1):38–71.

Russell, B. (1906). On denoting. *Mind*, 14. Reprinted in (Feigl and Sellars, 1949).

Shapiro, S. C. (1979a). Numerical quantifiers and their use in reasoning with negative information. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, pages 791–796. Morgan Kaufmann, San Mateo, CA.

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

Shapiro, S. C. (1982). Generalized augmented transition network grammars for generation from semantic networks. *American Journal of Computational Linguistics*, 8(1):12–25.

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

Shapiro, S. C. (1989). The CASSIE projects: An approach to natural language competence. In Martins, J. P. and Morgado, E. M., editors, *EPIA 89: 4th Portugese Conference on Artificial Intelligence Proceedings, Lecture Notes in Artificial Intelligence 390*, pages 362–380. Springer-Verlag, Berlin.

Shapiro, S. C. (1991). Cables, paths and "subconscious" reasoning in propositional semantic networks. In Sowa, J., editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 137–156. Morgan Kaufmann, Los Altos, CA.

Shapiro, S. C. (1992). Relevance logic in computer science. In Anderson, A. R., Belnap, Jr., N. D., and Dunn, M., editors, *Entailment*, volume II, pages 553–563. Princeton University Press, Princeton.

Shapiro, S. C. (1993). Belief spaces as sets of propositions. *Journal of Experimental and Theoretical Artificial Intelligence (JETAI)*, 5(2&3):225–235.

Shapiro, S. C. (1996). Formalizing English. *International Journal of Expert Systems*, 9(1):151–171.

Shapiro, S. C. (1998). 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.

Shapiro, S. C. and McKay, D. P. (1980). Inference with recursive rules. In *Proceedings of the First Annual National Conference on Artificial Intelligence*, pages 151–153. Morgan Kaufmann, San Mateo, CA.

Shapiro, S. C., McKay, D. P., Martins, J., and Morgado, E. (1981). SNePSLOG: A "higher order" logic programming language. SNeRG Technical Note 8, Department of Computer Science, SUNY at Buffalo. Presented at the Workshop on Logic Programming for Intelligent Systems, R.M.S. Queen Mary, Long Beach, CA, 1981.

Shapiro, S. C. and Rapaport, W. J. (1987). SNePS considered as a fully intensional propositional semantic network. In Cercone, N. and McCalla, G., editors, *The Knowledge Frontier: Essays in the Representation of Knowledge*, pages 262–315. Springer-Verlag, New York.

Shapiro, S. C. and Rapaport, W. J. (1991). Models and minds: Knowledge representation for natural-language competence. In Cummins, R. and Pollock, J., editors, *Philosophy and AI: Essays at the Interface*, pages 215–259. MIT Press, Cambridge, MA.

Shapiro, S. C. and Rapaport, W. J. (1992). The SNePS family. *Computers and Mathematics with Applications*, 23(2–5):243–275. Reprinted in (Lehmann, 1992, pp. 243–275).

Shapiro, S. C. and Rapaport, W. J. (1995). An introduction to a computational reader of narrative. In Duchan, J. F., Bruder, G. A., and Hewitt, L. E., editors, *Deixis in Narrative: A Cognitive Science Perspective*, pages 79–105. Lawrence Erlbaum Associates, Hillsdale, NJ.

Shapiro, S. C. and The SNePS Implementation Group (1998). *SNePS 2.4 User's Manual*. Department of Computer Science, SUNY at Buffalo, Buffalo, NY.