Formalizing English*

Stuart C. Shapiro
Department of Computer Science
and Center for Cognitive Science
State University of New York at Buffalo
226 Bell Hall
Buffalo, NY 14260-2000
U.S.A.
phone: 716-645-3180 Ext. 125
fax: 716-645-3464
e-mail: shapiro@cs.buffalo.edu

May 3, 1996

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 paper presents several aspects of such a logic.

Keywords: Knowledge representation; reasoning; natural language; commonsense reasoning; 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 are not, necessarily, 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 2.3 (Shapiro and The SNePS

*This is a preliminary version of Stuart C. Shapiro, Formalizing English. *International Journal of Expert Systems 9,* 1996. All quotes should be from, and all citations should be to the published version.

Implementation Group, 1995). 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 paper, 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, 1995, Chapter 7] an FOPL-like user interface to SNePS.

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 (*cf.* (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 paper. (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).)

3 Set-Oriented Logical Connectives

Consider the following problems:

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

1. Everything is an animal, a vegetable, or a mineral.
   Squash is a vegetable.
   Is squash an animal? a mineral?

2. For every object, the following statements are equivalent:

   - It is human
   - It is a featherless biped
   - It is a rational animal.

   Socrates is human.
   Is Socrates a featherless biped? A rational animal?

Consider formalizing (1). The FOPL wff

$$\forall x[\text{Animal}(x) \vee \text{Vegetable}(x) \vee \text{Mineral}(x)]$$

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

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

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 interpreted to mean 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, 1995, 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`, problem (1) 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)
```

[2]Some might argue that the "or" of (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.

(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. 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 paper. The character strings shown, however, are actual input and output.)

Notice that `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 cited above. It is not that SNePS logic does not contain inclusive or, but that the generalized exclusive or, "exactly 1 of ...," is also available.

Novice logicians would probably try to formalize (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

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

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)}).
  all(X)(HUMAN(X) <=> FEATHERLESS-BIPED(X)
                  <=> RATIONAL-ANIMAL(X))

: human(Socrates)!
  HUMAN(SOCRATES)
  FEATHERLESS-BIPED(SOCRATES)
  RATIONAL-ANIMAL(SOCRATES)
```

When 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"

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

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),IN(KATHY,LAS_VEGAS)} v=> {IN(EVE,PROVIDENCE)}
```

[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.

```
: 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:

```
(3)
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

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

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(X,Y,U,V)({VOTESFOR(U,X),VOTESFOR(V,Y)}
               &=> {andor(1,1){DISAPPOINTED(U),
                               DISAPPOINTED(V)}})
```

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

```
: all(u,x)({votesfor(u,x), wins(x)} &=> {~disappointed{u}}).
  all(U,X)({VOTESFOR(U,X),WINS(X)} &=> {~DISAPPOINTED(U)})

: votesfor(Hillary, Bill).
  VOTESFOR(HILLARY,BILL)

: votesfor(Barbara,George).
  VOTESFOR(BARBARA,GEORGE)

: wins(Bill).
  WINS(BILL)

: disappointed(?x)?
  DISAPPOINTED(BARBARA)
  ~DISAPPOINTED(HILLARY)
```

(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 Barbara is disappointed, but Hillary isn't.

The surprising aspect of this example is that in FOPL, an instance of (3) 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 explains 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

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

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 (3) 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

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

Because of the cumbersomeness of this formalization compared with the English statement, we have introduced *set arguments* into SNePS (Shapiro, 1986). Any predicate of the form

$$P(s_1, \ldots, s_i, \{t_1, \ldots t_n\}, s_{i+1}, \ldots, s_m)$$

implies every predicate of the form

$$P(s_1, \ldots, s_i, \tau, s_{i+1}, \ldots, s_m)$$

where $\tau \subset \{t_1, \ldots t_n\}$, and also every predicate of the form

$$P(s_1, \ldots, s_i, t_j, s_{i+1}, \ldots, s_m), 1 \leq j \leq n$$

(This is a version of what we have called "reduction inference" [Shapiro 1990, Shapiro and The SNePS Implementation Group, 1995, 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":

```
: sisters({Mary, Sue, Sally}).
  SISTERS({MARY,SALLY,SUE})

: all(x,y)(sisters({x,y}) => {likes(x,y), likes(y,x)})
  all(X,Y)({SISTERS({X,Y})} v=> {LIKES(X,Y),LIKES(Y,X)})
```

[5] A method of restricting such implications is planned for a future version of SNePS.

```
: 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)})).
  all(R)(TRANSITIVE(R)
         => (all(X,Y,Z)({R(X,Y),R(Y,Z)} &=> {R(X,Z)})))

: Transitive(bigger).
  TRANSITIVE(BIGGER)

: bigger(elephant,lion).
  BIGGER(ELEPHANT,LION)

: bigger(lion,mouse).
  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

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

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. Actually, according to a recent interpretation of SNePS logic, wffs such as `bigger(elephant,mouse)` are not 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 an example of this:

```
: ;;; Bob believes anything Bill believes.
all(p)(Believes(Bill, p) => Believes(Bob, p)).
  all(P)(BELIEVES(BILL,P) => BELIEVES(BOB,P))

: ;;; Bill believes whatever Kevin's favorite proposition is.
all(p)(Favorite-proposition(Kevin, p) => Believes(Bill, p)).
  all(P)(FAVORITE-PROPOSITION(KEVIN,P) => BELIEVES(BILL,P))

: ;;; Kevin's favorite proposition
  ;;; is that John is taller than Mary.
Favorite-proposition(Kevin, Taller(John, Mary)).
  FAVORITE-PROPOSITION(KEVIN,TALLER(JOHN,MARY))

: ;;; What does Bob believe?
Believes(Bob, ?what)?
  BELIEVES(BOB,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 didn't know that Sir Walter Scott was the author of Waverly" because Waverly was published anonymously. One obviously cannot replace "the author of Waverly" by "Sir Walter 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 "Sir Walter Scott" and "the author of Waverly" denote different *intensions,* but the same *extension.* In SNePS, all nodes 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 nodes 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)(Extensional(R)
        => all(a,x,y)({R(a,x), =({x,y})} &=> {R(a,y)})).
  all(R)(EXTENSIONAL(R)
        => (all(A,X,Y)({R(A,X),=({X,Y})} &=> {R(A,Y)})))

: Extensional(Dial).
  EXTENSIONAL(DIAL)

: =({Telephone(Mike), Telephone(Mary)}).
  =({TELEPHONE(MIKE),TELEPHONE(MARY)})

: Know(Pat, Telephone(Mike)).
  KNOW(PAT,TELEPHONE(MIKE))

: Dial(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.)

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

8 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 2.3 includes SNeBR (Martins and Shapiro, 1988), a Belief Revision system that has these two abilities.

When some wff 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)(BIRD(X) => FLIES(X))

: all(x)(Penguin(x) => Bird(x)).
  all(X)(PENGUIN(X) => BIRD(X))

: all(x)(Penguin(x) => ~Flies(x)).
  all(X)(PENGUIN(X) => (~FLIES(X)))

: Bird(Opus)!
  FLIES(OPUS)
  BIRD(OPUS)

: Penguin(Opus)!
  A contradiction was detected within context DEFAULT-DEFAULTCT.
  The contradiction involves the newly derived node:
      ~FLIES(OPUS)
  and the previously existing node:
      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 wff(s) that caused the contradiction.

The system keeps track of the hypotheses that underly inferred wffs. (An hypothesis is a wff 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 wff that was derived from it:

```
: all(x)(Bird(x) => Flies(x)).
  all(X)(BIRD(X) => FLIES(X))

: all(x)(Flies(x) => Feathered(x)).
  all(X)(FLIES(X) => FEATHERED(X))

: all(x)(Canary(x) => Bird(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)

: ~Canary(Clyde).
  A contradiction was detected within context DEFAULT-DEFAULTCT.
  The contradiction involves the node you want to assert:
      ~CANARY(CLYDE)
  and the previously existing node:
      CANARY(CLYDE)
  You have the following options:
```

```
   1. [c]ontinue anyway,
      knowing that a contradiction is derivable;
   2. remove [a]nother assertion from this context
   3. [r]emove the new assertion from this context
  (please type c, a or r)
=><= a
  In order to make the context consistent
 you must delete some hypotheses from the set (WFF8)
...
Do you want to take a look at hypothesis M8!?
=><= y
  CANARY(CLYDE)
...
  What do you want to do with hypothesis M8!?
  [d]iscard from the context, [k]eep in the context,
  [u]ndecided, [q]uit this package
  (please type d, k, u or q)
=><= d
  ~CANARY(CLYDE)

: list-asserted-wffs
  all(X)(BIRD(X) => FLIES(X))
  ~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 wffs that were inferred from it, `FLIES(CLYDE)`, `FEATHERED(CLYDE)` and `BIRD(CLYDE)` were also removed.

9 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 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) => FEATHERED(X))
```

```
: all(x)(~Flies(x) => Swims(x)).
  all(X)((~FLIES(X)) => SWIMS(X))

: Flies(Opus).
  FLIES(OPUS)

: ~Flies(Opus).
  A contradiction was detected within context DEFAULT-DEFAULTCT.
  The contradiction involves the node you want to assert:
      ~FLIES(OPUS)
  and the previously existing node:
      FLIES(OPUS)
  You have the following options:
   1. [c]ontinue anyway,
      knowing that a contradiction is derivable;
   2. remove [a]nother assertion from this context
   3. [r]emove the new assertion from this context
  (please type c, a or r)
=><= c
  ~FLIES(OPUS)

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

  and the previously existing node:
      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 and the next prompt.)

So the contradiction allows the system to infer related contradictory information, specifically `FEATHERED(OPUS)` and `SWIMS(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)(CANARY(X) => BIRD(X))

: all(x)(Bird(x) => Flies(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).
  PENGUIN(OPUS)

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

The implication is not derived.

10 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){North-of(x,y), South-of(y,x)}).
  all(X,Y)(NORTH-OF(X,Y) <=> SOUTH-OF(Y,X))

: North-of(Seattle, Portland).
  NORTH-OF(SEATTLE,PORTLAND)

: South-of(San_Francisco, Portland).
  SOUTH-OF(SAN_FRANCISCO,PORTLAND)

: North-of(San_Francisco, Los_Angeles).
  NORTH-OF(SAN_FRANCISCO,LOS_ANGELES)

: South-of(San_Diego, 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)(PARENT(X,Y) => ANCESTOR(X,Y))

: all(x,y,z)({ancestor(x,y), ancestor(y,z)} &=> {ancestor(x,z)}).
  all(X,Y,Z)({ANCESTOR(X,Y),ANCESTOR(Y,Z)} &=> {ANCESTOR(X,Z)})

: parent(John, Mary).
  PARENT(JOHN,MARY)

: ancestor(Mary, George).
  ANCESTOR(MARY,GEORGE)
```

```
: ancestor(George, Sally).
  ANCESTOR(GEORGE,SALLY)

: parent(Sally, Jimmy).
  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 $\forall(x)[P(f(x)) \Rightarrow P(x)]$.

11 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 paper. We may subcategorize them as follows.

- Those that differ from FOPL in syntax (with appropriate semantics):
  - Set-Oriented Logical Connectives
  - Set Arguments
  - "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:
  - Belief Revision
  - Use of circular and recursive rules

A more recent development, that has not yet been incorporated with the other features of SNePS 2.3, is "structured variables" (Ali, 1993; Ali and Shapiro, 1993; Ali, 1994).

12 Acknowledgments

Previous versions of this paper 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 paper 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, State University of New York at Buffalo (UB). 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 paper, 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, and in part by NASA under contract NAS 9-19335.

References

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.

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. 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.

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.

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

Shapiro, S. C. (1979). The SNePS semantic network processing system. In Findler, N. V., editor, *Associative Networks: The 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. *The 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. 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*, pages 263–315. 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 & 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 narratives. 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, Inc., Hillsdale, NJ.

Shapiro, S. C. and The SNePS Implementation Group (1995). *SNePS 2.3 User's Manual*. Department of Computer Science, SUNY at Buffalo, Buffalo, NY.