1981-5

SNePS Implementation of Possessive Phrases

Soon Ae Chun

Department of Computer Science
State University of New York at Buffalo
May 1987

SNeRG Technical Note #19

1. PROJECT GOAL

The current project concentrates on the issues involved in understanding English possessive phrases. These issues are divided into two subcomponents: one component deals with the issues of representation of possessive phrases and the other deals with the issues of implementing the possessive phrases in SNePS (Shapiro 1979). The current project, however, discusses only the **'s** type (Genitive case) possessive phrases of English, putting aside other means of expressing possession. Before I launch into the discussion of the two subcomponents of the project, I will briefly discuss the linguistic issues concerning possessive phrases.

2. LINGUISTIC ISSUES

This section discusses the syntactic expressions of possession in English and the different types of semantic relations expressed by **'s** phrase.

1. The syntactic devices available to express possession are listed in the following:

(1)

a.	A**'s** B (Genitive case('s))	John's hat
b.	B **of** A	The leg of the table
c.	A **have** B	John has a hat.
d.	B **is A's**	The hat is John's

In the above list, A is called **possessor** and B is called **possessed (object)**. Of all possible devices, this project concentrates on the Genitive case type of possessive phrases.

2. In (2) the genitive case **'s** conveys different relations between possessor and possessed:

(2)

a.	body part relation	John's arm
b.	kinship relation	John's father
c.	location	John's hometown
d.	possession	John's book

This list of semantic relations is not exhaustive by any means. However, the categorization above includes most of the semantic relations. The relations basically depend on the types of possessed. If the possessed is animate, then the relationship between possessor and possessed is mostly kinship/interpersonal relations. If the possessed is a body part nominal, then the relation between possessor and possessed will mostly be body part. But if possessed is inanimate, then the relationship is possession or location. Thus, semantic features like [animate], [body] or [inanimate] will not perfectly predict the semantic relations between possessor and possessed.

It seems that all items in the world can be described as either (1) neutral items or (2) items belonging to some agent. The first case is usually described as non-possessive phrases and used when the agent possessing a particular item is not known or when the expression of the agent possessing an item is not of interest. The second case is expressed with possessive constructions where the possessor is explicitly specified. Consequently, the possessed item is from the world of the possessor rather than from the neutral world. In other words, there seems to be a world (or mental space) where all items are considered as possessed by an agent. When we look at these phenomena this way, all items which have to do with an agent *John* can be optionally expressed with possessive constructions: *John's X*. This view naturally leads to the uniform treatment of the possessive construction, regardless of the semantic diversity conveyed by it.

3. REPRESENTATIONAL ISSUES

Considering different semantic relations between possessor and possessed, we are tempted to represent them differently. However, if the possessed is considered to be an item from the possessor's world (domain), all the semantic diversity seems to be captured in a unique way. Namely, the possessor serves as the domain from which the item which stands in a particular relation, such as *father_of*, *book_of* or *right_arm_of*, is selected. Another way of looking at it is that there is a function which takes the possessor as its domain and returns an item which stands in a particular relationship (such as *father_of*) with the possessor. This functional relation can be denoted with the following notation:

(3)

a. father_of (John) == John's father
b. book_of (John) == John's book

The corresponding case frame of the relation is shown in Figure 1. The possessive relation is represented with a propositional node labeled *m4* in the diagram. *m4* denotes that a possessor *m1* has a relation *m2* with the entity labeled *m3*. For example, *John's father* is represented as in Figure 2. Another example of *Bill read John's favorite book* is shown in Figure 3. Notice that the modifier of *book*, *favorite*, is syntactically represented as *modifier* to the head of noun phrase *book*.

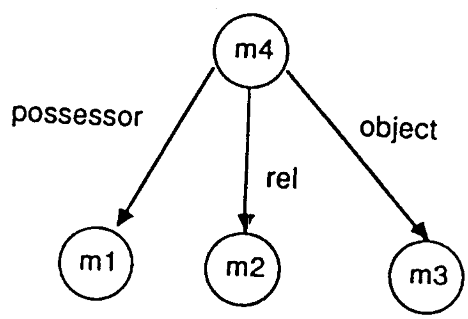

Figure 1. Case Frame for Possessive Relations

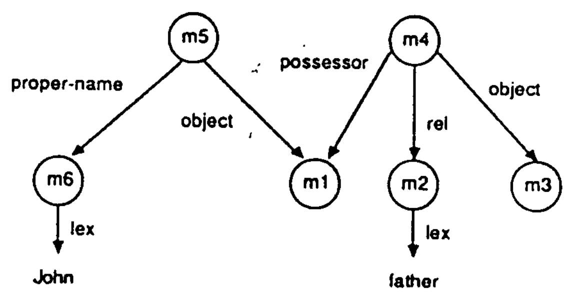

Figure 2. Representation of *John's father*

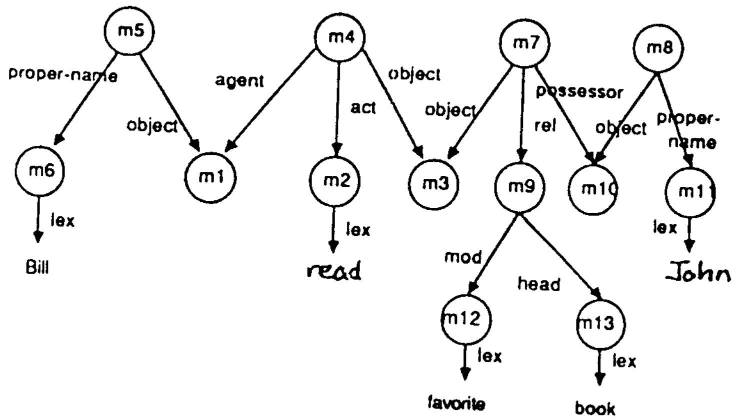

Figure 3. Representation of *Bill read John's favorite book.*

4. COMPUTATIONAL ISSUES

In this section, I describe the approach of implementing the parsing and the generation of possessive phrases. These are basically the descriptions of changes that I have made in the existing ATN grammar.

1. The major change for parsing is done in the states **npp, npdet,** and **npa,** where noun phrases are parsed. A Noun phrase, before my change, is parsed by these states as shown in Figure 4. For example, consider a noun phrase *a yellow dog*. The **det** arc at state **npp** is taken

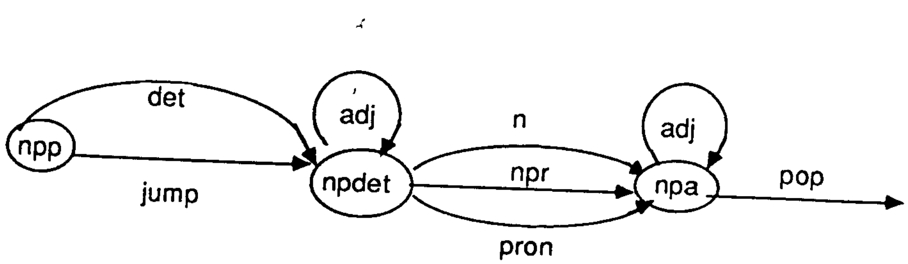

Figure 4. Transition Network for Parsing NP (before change)

with the word *a*. At state **npdet** the **adj** arc is taken with the word *yellow*. Here the adjective *yellow* is stored in register **adj** for later use at state **npa.** Next the parser takes the **n** arc at state **npdet** with the noun *dog*, and then it stores it in the **nh** register. Finally, at state **npa** the parser finishes the noun phrase parsing and builds an *object-property* case frame with the adjective stored in **adj** register, namely, *yellow*, as *property* and the noun stored in **nh** register, namely *dog*, as *object*.

To parse the possessive phrase, I added a state **nppos** and **nprel** as in Figure 5. Let us consider a phrase *Lucy's little brother* to see how a possessive phrase is parsed. First, the parser takes the **jump** arc at state **npp** with the word *Lucy*. At state **npdet,** the arc **npr** is taken with *Lucy* and goes to state **nppos.** At state **nppos** the **wrd '** arc is matched with the current word ' but the flag **pos** is not set. Thus the parser jumps to state **npa.** (The wrd arc ' at state **nppos** is only taken when the parser parses the second or third possessor in multiple possessor phrases, such as A's B's C.) At state **npa,** the parser sets the flag **pos** to be true and stores *Lucy* in register **possor**. Then the parser goes back to state **npdet.** At state **npdet** the parser has *s*

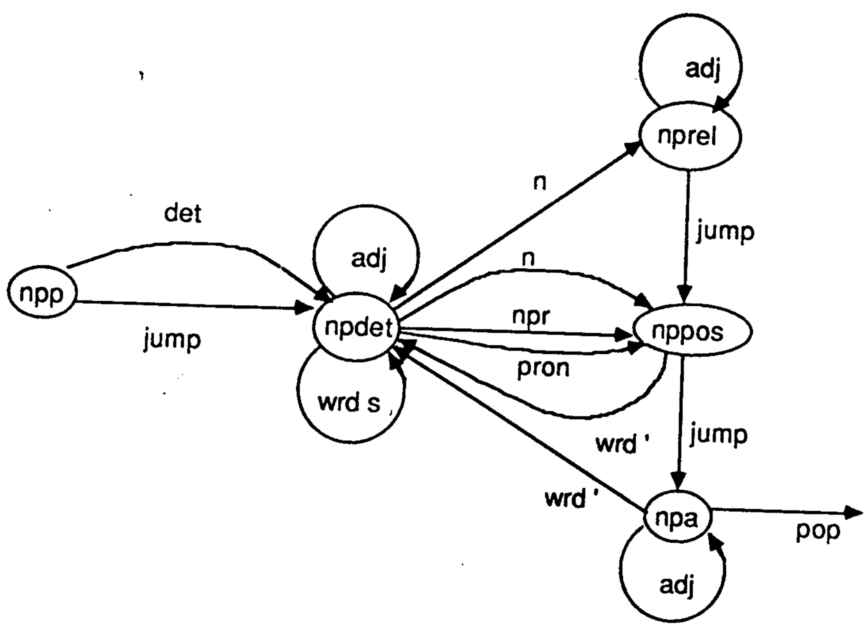

Figure 5. Transition Network for Parsing NP (after change)

as its current word and it matches with the word arc.

After parsing up to *Lucy's*, the parser at state **npdet** follows the arc **adj** to itself with the word *little*. Then, with the word *brother*, the parser takes arc **n** leading to state **nprel** since the possessive flag **pos** is set when the parser parsed *Lucy's*. At state **nprel**, the parser builds (or finds, if it already exits) the *mod-head* case frame with the adjective *little* and the head noun *brother*. Then the parser jumps to state **npppos** to find or build the *possessor-rel-object* case frame with possessor *Lucy*, rel *little brother* and object a base node. Finally, the parser jumps to state **npa** to pop from the np parsing, with a node built (or found) (labeled m8 in Figure 6) which corresponds to the proposition of *Lucy's little brother* represented by the *possessor-rel-object* case frame. (See Figure 6 for the resulting network representation in CASSIE's mind.)

2. In order to generate possessive phrases, I added states **possessive, rel, mod** and **head.** (See Shapiro 1982 for the details on generating from the semantic networks.) To illustrate the process of generating a possessive phrase, consider the network built in Figure 6 on the example of *Lucy's little brother*.

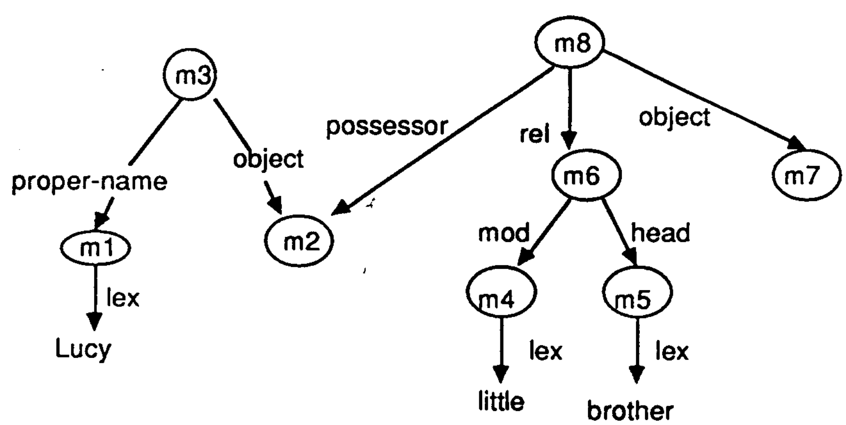

Figure 6. Representation of *Lucy's little brother*.

Given node **m8** as the current node at state gs, the generator decides the node headed by the **object** arc, namely **m7**, is going to be the subject of the sentence and the node **m8** is to be the object of the sentence. In other words, it prepares to generate a sentence *m7 is Lucy's little brother*. (Notice here we do not know what is m7 in the current network. It can be a simple proper noun like *John* or it can be any description of *Lucy's little brother*.) Assume that m7, the subject of the sentence as well as the verb *is* are generated and the generator is about to generate the "object" of the sentence with node **m8**, namely *Lucy's little brother*.

The generator separates the node **m2** for generating the possessor from the node **m6** for generating the possessed. Now the generator jumps to state **possessive** to generate the possessor *Lucy's*. (See Figure 7 for the sketchy transition networks involving the generation of possessive phrases.) At state **possessive**, the generator calls state **np** where all noun phrase generation takes place. The state **np** recursively calls itself until the generator goes down to the node **m1** from which the *lex* arc emanates. With node **m1**, the generator generates the English word *Lucy* and *'s*.

After generating the possessor at state **possessive**, the generator jumps to state **rel** with node **m6** as the current node to generate. At state **rel**, the generator calls state **np** to generate the noun phrase *little brother*. At state **np**, the generator jumps to state **np1** and "prepares" to generate the modifier and the head separately, if there is any modifier. In the current example, the node **m4** is set to be modifier and **m5** to be head. With the node **m4**, the generator produces the English word *little*. After the modifier is generated at the state **mod** by calling the state **np**, the generator jumps to the state **head** and generates the English word *brother* with node **m5** by calling the state **np**. After generating modifier(s) and head noun, the generator pops to state **rel** completing the generation of the node headed by the **rel** arc and popping from the generation of a whole possessive phrase.

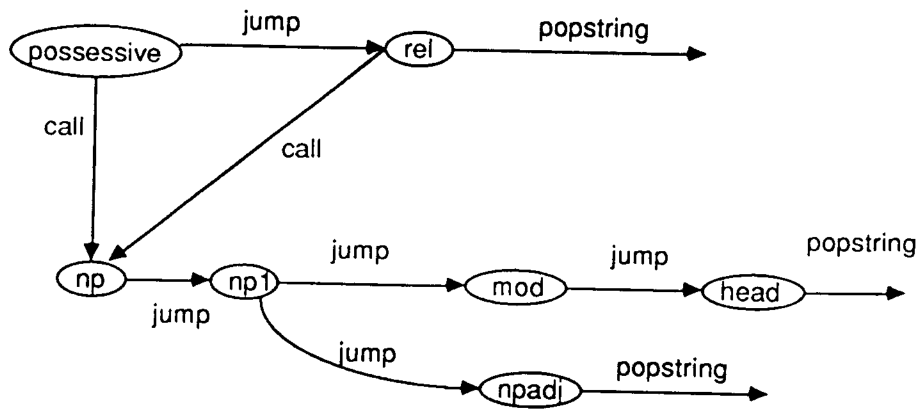

Figure 7. Nodes for generating possessive phrases

5. ADDITIONAL COMPUTATIONAL CHANGES

In addition to the changes for parsing and generating possessive phrases, I also made changes to process the embedded subject clause, embedded complement clause and comparative phrases. The following list contains the states in transition network that I have added to process possessives and other phrases mentioned above.

1. sc -- In state **npp,** if the word was *that* then the **sc** node is taken to parse an embedded clause. In **sc,** it is pushed to **sp** for sentence parsing.

2. cc -- At this state the parser parses the complement clause by pushing to **sp** to handle *X is that S* or *X is (comparative) than that S.*

3. **compare** -- In this state, parse the noun phrase following the comparative adjective such as *than NP.* Set the **complement** register to the noun to build **arg1-rel-arg2** case frame later.

4. **nprel** -- As described in the previous section, in this state the **head-mod** case frame is built for the **rel** arc of the possessive case frame.

5. **nppos** -- As shown in the previous section, the whole possessive case frame **possessor-rel-object** is built in this state.

6. **surrel** -- In this state, the comparative adjective and *than* of the *arg1-rel-arg2* case frame are generated.

7. **possessive** -- The possessor of the possessive case frame and *'s* are generated in this state.

8. **rel** -- The **rel** arc of the possessive case frame is processed to generate the relation involved in the possessive phrase.

9. **mod** -- The adjective qualifying the head of the possessive phrase is generated.

10. **head** -- The head noun of the possessive phrase is generated.

11. Different networks are built for the following sentences:

(4)

a. That John is taller than Mary is Kevin's favorite proposition.
b. Kevin's favorite proposition is that John is taller than Mary.

With sentence (4a), CASSIE first identifies an entity (intensional object) as Kevin's favorite proposition and then finds what relation that entity has with *Kevin*. With sentence (4b), CASSIE first builds the base object described as *Kevin's favorite proposition* and then it finds out what that proposition is. For (4b), an explicit **equiv-equiv** case frame is used to note that the entity described as *Kevin's favorite proposition* is equivalent to the *that*- clause, whereas for (4a) the proposition *that*- clause is the object of the possessive case frame. This asymmetry is shown in Figures 8 and 9.

6. LIMITATIONS OF THIS PROJECT

1. The current project only handles **'s** type possessive phrases, ignoring all other possessive devices. This means further work must be done on possessive expressions in order to handle other types of syntactic devices, such as the *of* phrase and *have*. The difficulty arises immediately by just observing the various possibilities in interpreting *of* phrases:

(5)

a. The legs *of* a table (possession)
b. one *of* the pioneers
c. a discussion *of* semantic theory
d. a collection *of* sentences

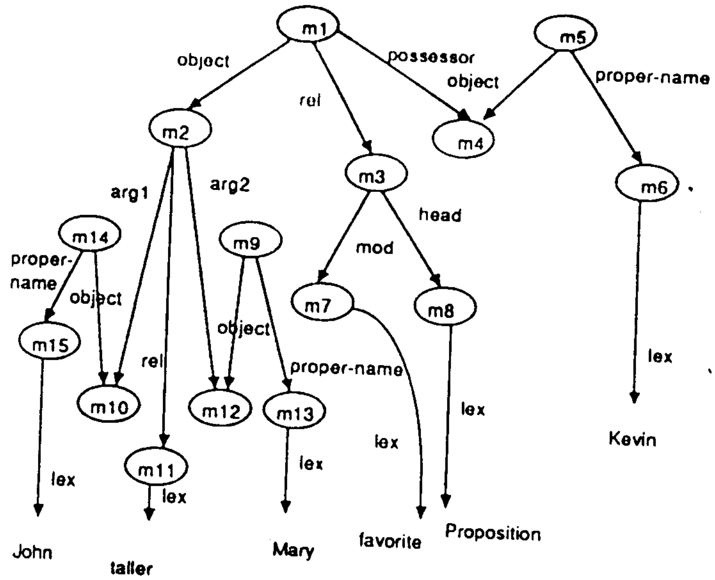

Figure 8. Representation of *That John is taller than Mary is Kevin's favorite proposition*

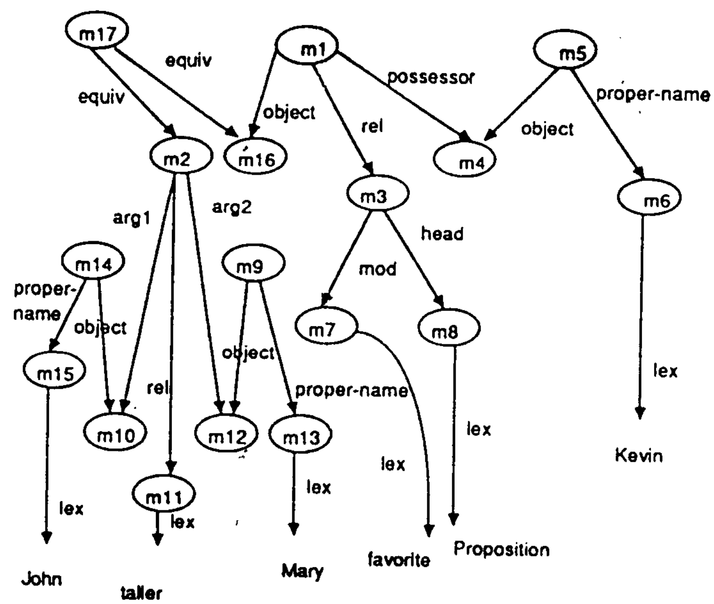

Figure 9. Representation of *Kevin's favorite proposition is that John is rich*

Even the 's type of possessive phrase, which is considered a reduced version of a complete sentence (see the following example), is not handled:

(6)

a. John's blowing bubbles made us laugh.

REFERENCE

Barnden, John A. A Viewpoint Distinction in the Representation of Propositional Attitudes. *Proceedings of Fifth National Conference on Artificial Intelligence.* 1986.

Bates, M. The Theory and Practice of Augmented Transition Network Grammars. In L. Bolc, ed. *Natural Language Communication with Computers.* Springer Verlag, Berlain, 1978, 191-259.

Li, N. Pronoun Resolution in SNePS. *SNeRG Technical Note #18.* SUNY at Buffalo, 1987.

Maida, A. S. and Shapiro, S. C. Intensional Concepts in Propositional Semantic Networks, *Cognitive Science 6(4)*, 1982, 291-330.

Shapiro, S. C. The SNePS Semantic Network Processing System, In N. V. Findler, ed. *Associative Networks: The Representation and Use of Knowledge by Computers.* Academic Press, New York, 1979, 179-203.

Shapiro, S. C. Generalized Augmented Transition Network Grammars For Generation from Semantic Networks. *The American Journal of Computational Linguistics.* 1982.

Shapiro, S. C. and Rapaport, W. J. SNePS Considered as a Fully Intensional Propositional Semantic Network. *Proceedings of Fifth National Conference on Artificial Intelligence.* 1986.

Shapiro, S. C. and Rapaport, W. J. Intensional Knowledge Representation in SNePS: Models and Minds. *forthcoming.*

Winograd, T. *Language as A Cognitive Process Vol 1: Syntax.* Addison Wesley. 1983.

The logical subject of the act of *blowing bubbles* is expressed with the **'s**. Such possessive nominals are not handled in the current project.

2. The possessive pronoun is not handled in the current project at all. This future work should be able to handle first, correct parsing, second, identification of the right referent, and third, generation of referent in the possessive form. This work has to be incorporated with the work on pronoun resolution in general. (Li 1987)

3. The current work did not touch the inference involved with the possessive phrases. For example, the phrase *John's sister's husband* should be identified as the same as *John's brother-in-law* through the knowledge base and inference. Another example we might encounter in larger discourse are narratives like: the narratives like:

(7)

USER	:	John owns a car.
		It is yellow.
		Is John's car yellow?
SYSTEM	:	Yes, John's car is yellow.

We want the pronoun **it** in the second input sentence to be interpreted as *John's car*, even though there is no explicit input phrase *John's car*, so that CASSIE can answer the third input question correctly as shown above. This involves the inference from *own* $(X\ Y)$ to the possession and generates or identifies the possession with possessive phrases.