Defining “labyrinth” through Contextual Vocabulary Acquisition

Joe Salazar
CSE 663: Advanced Topics in Knowledge Representation and Reasoning
December 15, 2006

Abstract

Contextual Vocabulary Acquisition (CVA) is the act of deriving the meaning of a word from the context in which it appears. The goal of the CVA project is to develop a computational model of this task. To accomplish this, the SNePS knowledge representation and reasoning system is used to model both the mind of a reader and the process of learning a new word. A cognitive agent known as Cassie is given a representation of a passage and an appropriate set of background knowledge. She then comes up with an approximate definition using reasoning and a CVA algorithm designed for a particular part of speech. In this paper, an algorithm for nouns is used to define the word "labyrinth." The results of the algorithm are evaluated by comparing them to a dictionary definition and verbal protocol results.

Introduction to Contextual Vocabulary Acquisition and SNePS

Suppose you are reading a book and you come across a word that you are not familiar with. Most people will not immediately consult a dictionary for a definition. Rather, they will either ignore the word if it is unimportant or try to figure out what it means from the context in which it appears. By doing the latter, a reader is engaging in what is known as contextual vocabulary acquisition (CVA). Using a combination of their own relevant background knowledge and the information in the sentence, the reader attempts to derive a meaning for the unknown word. If the reader is repeatedly exposed to the word in varying contexts, their derivation will improve and will gradually "converge" towards a dictionary-like definition. (Rapaport and Ehrlich 2000: 5)

The goal of the CVA project is to develop a computational theory of vocabulary acquisition in a natural-language-understanding system. By developing algorithms for this task, a better understanding of context can be gained and can provide a foundation for a curriculum which takes the guesswork out of CVA. (Rapaport and Kibby 2002: 3) Of utmost importance is modeling the activity of learning a new word. (Rapaport and Ehrlich 2000: 1) To do so, we enlist the help of Cassie, the cognitive agent of SNePS.

SNePS is an "intensional, propositional, semantic-network knowledge-representation and reasoning system used for research in artificial intelligence and in cognitive science." (Shapiro and Rapaport, 1995) The structure of this semantic network and the features it provides make it an appropriate choice for modeling cognitive activities. For instance, the SNePS Inference Package (SNIP) allows Cassie to perform both conscious and subconscious reasoning in the form of inferences. It also allows her to engage in belief revision in order to handle any contradictory information that she might encounter. (Rapaport and Ehrlich 2000: 8) At any given moment, the structure of the network as a whole represents Cassie's beliefs at that time. (Shapiro 1989: 3)

This particular project aims to test the functionality of the noun algorithm by giving Cassie a passage to read, a set of appropriate background knowledge and a noun to define. The results of this algorithm will be compared against both a dictionary definition and the definitions of human test subjects performing the same task in order to evaluate its performance.

The Passage

The source passage for this project was taken from the short story "The Two Kings and the Two Labyrinths" by Jorge Luis Borges:

"It is said by men worthy of belief (though Allah's knowledge is greater) that in the first days there was a king of the isles of Babylonia who called together his architects and his priests and bade them build him a **labyrinth** so confused and so subtle that the most prudent men would not venture to enter it, and those who did would lose their way." (Borges: 263)

A sentence of this length is difficult to understand, let alone represent. The following revised version was used instead:

"A king built a **labyrinth** so confused and so subtle that men who entered it would lose their way."

Special care must be taken when revising a passage in this manner. There may be a loss of information important to the proper understanding of a sentence if it is oversimplified. Even worse, the meaning of the sentence may change significantly; leaving a problem to solve that is altogether different from the original passage. In this case, verbal protocol testing (discussed in the following section) indicates that this revision does not suffer from either of these problems.

In Rapaport and Kibby 2002, Kibby enumerates six levels of “knowing a concept.” The third level is “knowing [the] concept and another word (or words) that signifies it, but not the specific word in the text to signify it.” He states that this particular level is well-suited to CVA since the context will usually be helpful.

This passage is a good example of the third level. While younger readers may be familiar with the concept of a maze, they may not be familiar with the term “labyrinth” even though the two are closely related. In fact, the Oxford English Dictionary uses the term “maze” as part of its definition of labyrinth:

labyrinth, *n.*

1. A structure consisting of a number of intercommunicating passages arranged in bewildering complexity, through which it is difficult or impossible to find one's way without guidance; a maze.
    a. With references to the structures so named in classical antiquity.
    b. In mod. landscape gardening, a maze formed by paths bordered by high hedges. (OED)

Verbal Protocols

In order to determine what background information to represent in SNePS, several people were asked participate in what were known as “verbal protocols.” Basically, each subject was given the passage to read and asked to give a definition of the word and reasons why they arrived at that conclusion. However, the test results are not very useful if the subject already knows the definition of the word. Since “labyrinth” is not terribly uncommon, it was replaced with the made-up word “mecasoph” for each test. This ensured that the test subjects had to deal with a “Level 3” word regardless of whether they knew the definition of the word or not.

The following responses are representative of the most common definitions given:

- “a maze” (nearly everyone used this at some point during their definition)
- “A building or structure which disrupts the senses”
- “Some sort of confusing building”
- “A labyrinth of confusing tunnels”

Since most of the test subjects came up with an appropriate definition, it is clear that the context is helpful. However, each test subject came up with different reasons when asked to rationalize their answers:

- “It’s large enough to be entered by a man, so it must be a building.”
- “It’s a building that’s confusing.”
- “It was built to be confusing.”
- “Those who entered it got lost.”
- “Mazes are designed to be confusing and people who enter them tend to lose their way.”
- “The king is representative of a time period in which mazes were more common.”
- “The mention of the king reminded me of a hedge maze in a castle.”

Apparently there is quite a lot of contextual information for such as short sentence! It seems that people arrived at the same definition using different sets of background knowledge. Some people keyed in on the fact that it was a confusing building while others focused on the fact that people who entered it got lost. Still others made a connection to mazes in antiquity because of the mention of the king.

So, what background knowledge should be represented? For the purposes of this project, the information pertaining to “confusing buildings” will be all that is included. Having Cassie arrive at the same conclusion using a different knowledge base is an idea for future work on this project.

Background Knowledge

Now that a decision has been made as for what to represent, the issue of how to represent it needs to be dealt with. The noun algorithm is familiar with a set of commonly used relations known as case frames. Any information represented using these relations should show up in the definition when the algorithm is invoked. Node diagrams and semantics for each case frame used in this project are detailed in Appendix A. The following code sections use what is known as the SNePS User Language (SNePSUL) which is similar in style to Lisp.

The first piece of background knowledge is described as follows:

```
;; =========================================
;; B1: "If something is confused and subtle,
;;      then it is confusing."
;; =========================================

(describe (assert forall $z
       &ant ((build object *z property (build lex "confused"))
             (build object *z property (build lex "subtle")))
       cq (build object *z property (build lex "confusing"))))
```

The words “confused” and “confusing” are very close lexicographically and some of the test subjects may have simply interchanged the meaning of the two. However, their definitions are similar enough where such a swap does not have a major impact on the meaning of the sentence. The “object/property” case frame is appropriate here since all parts of the implication deal with properties that the labyrinth may have.

The second piece of background knowledge is represented as follows:

```
;; =============================================
;; B2: "If x enters y then x is smaller than y."
;; =============================================

(describe (assert forall $v forall $w
       ant (build agent *v act (build action (build lex
                "enters") object *w))
       cq (build object1 *v rel (build lex "smaller than")
object2 *w)))

;; ======================================================
;; B2': "If x is smaller than y then y is larger than x."
;; ======================================================

 (describe (assert antonym (build lex "smaller than")
           antonym (build lex "larger than")))
```

The fact that an agent can enter an object implies that the agent is smaller than the object. Since there is no predefined relation for “size” given, the “object1/rel/object2” case frame must be used. The relation to be defined is “smaller than” and operates on the agent and the object in the implication. Note that other definitions of “enter” are ignored here, such as “entering a war” or “entering a contest.” In these cases, a relation based on physical size is not appropriate.

Also, since the agent is smaller than the object, the object is larger than the agent. This property is represented using the “antonym/antonym” case frame with the concept of “larger than” being the opposite of the concept of “smaller than.”

The third piece of background knowledge is represented as follows:

```
;; ==========================================
;; B3: "If x builds y then y is a structure."
;; ==========================================

(describe (assert forall ($m $n)
       ant (build agent *m act
               (build action (build lex "build") object *n))
       cq (build member *n class (build lex "structure"))))
```

The first clue as to the identity of the object is that it was built by some agent. The fact that the agent was a king seems to indicate that it's a fairly sizeable construction project. At the very least it's a structure of some sort, so that's the implication that is represented here.

The final piece of background knowledge is represented as follows:

```
;; ================================================================
;; B4:"If x is a structure and x is confusing then x is a maze."
;; ================================================================

(describe (assert forall $p
       &ant ((build object *p property (build lex "confusing"))
             (build member *p class (build lex "structure")))
       cq (build member *p class (build lex "maze"))))
```

The OED definition given in an earlier section refers to a "bewildering structure" which is not easy to navigate without help. A "confusing structure" is nearly the same thing, so the implication that is represented is that all things that are structures and are confusing are mazes.

Representing the Passage

In order to construct the sentence in SNePS, it first needed to be decomposed into smaller parts which could be more easily represented. The passage for Cassie to read was separated into the following components:

- "A king built a labyrinth…"
- "…a labyrinth so confused and so subtle..."
- "…that men who entered it would lose their way."

Dividing the passage in this fashion allowed each component to be easily represented using the case frames that the noun algorithm is familiar with.

The first part is represented as follows:

```
;==========================================
; P1: "A king built a labyrinth..."
;==========================================

(describe (add agent #king act (build action
          (build lex "build") object #labyrinth)))

(describe (add member *king class (build lex "king")))
(describe (add member *labyrinth class (build lex "labyrinth")))
```

The case frame used for this particular representation was “agent/act/action/object.” The king was the agent, the action was to “build” the object and the object was the labyrinth. The “member/class” case frame is also used, as the king is represented by a base node which is a member of the class of kings. The labyrinth is also represented by a base node which is a member of the class of labyrinths.

Most people who read the sentence implicitly understood that the king did not build the labyrinth on his own; he ordered his subjects to build it for him. However, this is not explicitly represented in the network because the test subjects did not use this information to come up with their definition.

The second part is represented as follows:

```
;=================================================
; P2: ...a labyrinth so confused and so subtle...
;=================================================

(add object *labyrinth property (build lex "confused"))
(describe (add object *labyrinth property (build lex "subtle")))
```

The “object/property” case frame was the representation of choice here. The labyrinth is described as having the properties of being confused and being subtle.

The final section is represented as follows:

```
;=====================================================
; P3: ...that men who entered it would lose their way.
;=====================================================

(describe (add forall $y
          &ant ((build member *y class (build lex "human"))
                (build agent *y act (build action (build lex
                "enters") object *labyrinth)))
          cq (build object *y property (build lex "lost"))))
```

A universal quantifier is required to properly represent this sentence. For all men, if those men enter the labyrinth then those men are lost. The “member/class”, “agent/act/action/object” and “object/property” case frames are all in use here. The agent is a member of the class of men, the act is “entering” the object, the object is the labyrinth and the implied property of the object entering the labyrinth is “being lost.”

Results and Further Work

The CVA algorithms work on the principle that “the meaning of a term is its location in the network of background and story information.” They search through the network to find the appropriate information for a definition. (Rapaport and Ehrlich 2000: 11) In the case of the noun algorithm, this information includes things such as class inclusions, properties, structure, function and actions either performed by it or upon it.

When asked for a definition, Cassie returned the following information:

```
^(
--> defineNoun "labyrinth")
 Definition of labyrinth:
 Possible Class Inclusions: structure, maze,
 Actions performed on a labyrinth: king build,
 Possible Properties: confused, subtle, confusing,
nil
```

Appropriately, Cassie returned a definition that was similar to the dictionary definition. She came to the conclusion that the confusing structure in question was a maze using similar reasoning to most of the verbal protocol subjects. While the definition still has ties to the context (in particular, the mention of the king in the definition) this is not a cause for concern. It is not unreasonable to think that a reader on their first encounter with the word would think that the presence of the king was important.

While Cassie did come up with a proper definition, it would have been interesting to see if she could have arrived at it using different contextual information. As mentioned before, the verbal protocol subjects defined the word appropriately but drew on a wide variety of background knowledge. Representing these different sets of context could be an interesting addition to the project.

Appendix A: Case Frames and Semantics

Note: All images and semantics are taken from Napieralski's Dictionary of CVA Case Frames at `http://www.cse.buffalo.edu/~rapaport/CVA/cvaresources.html`.

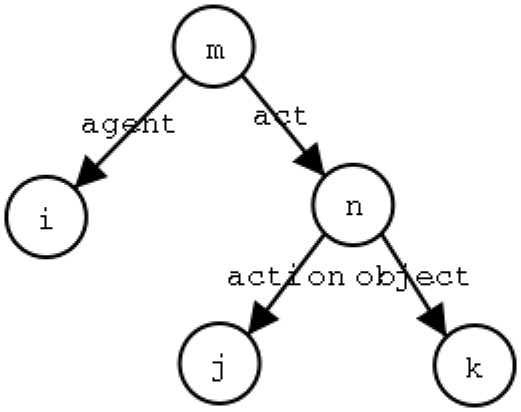

Figure 1: "agent/act/action/object" frame

Semantics

[[m]] is the proposition that agent [[i]] performs action [[j]] with respect to object [[k]].

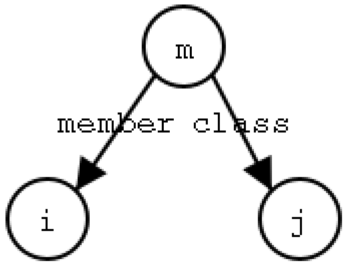

Figure 2: "member/class" frame

Semantics

[[m]] is the proposition that [[i]] is a member of the class [[j]].

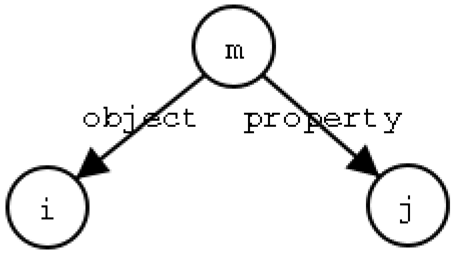

Figure 3: “object/property” frame

Semantics

[[m]] is the proposition that [[i]] has the property [[j]].

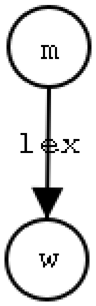

Figure 4: “lex” frame

Semantics

[[m]] is the concept expressed by uttering [[w]].

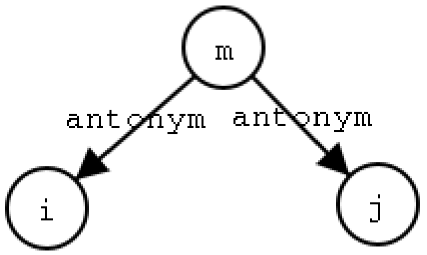

Figure 5: “antonym/antonym” frame

Semantics

[[m]] is the proposition that some concepts [[i]] and [[j]] are antonyms.

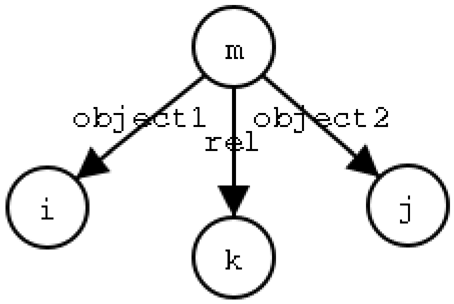

Figure 6: “object1/rel/object2” frame

Semantics

[m]] is the proposition that [[i]] is related by relation [[k]] to [[j]].

Appendix B: Network Diagrams

These network diagrams were created by using the SNePSUL show command.

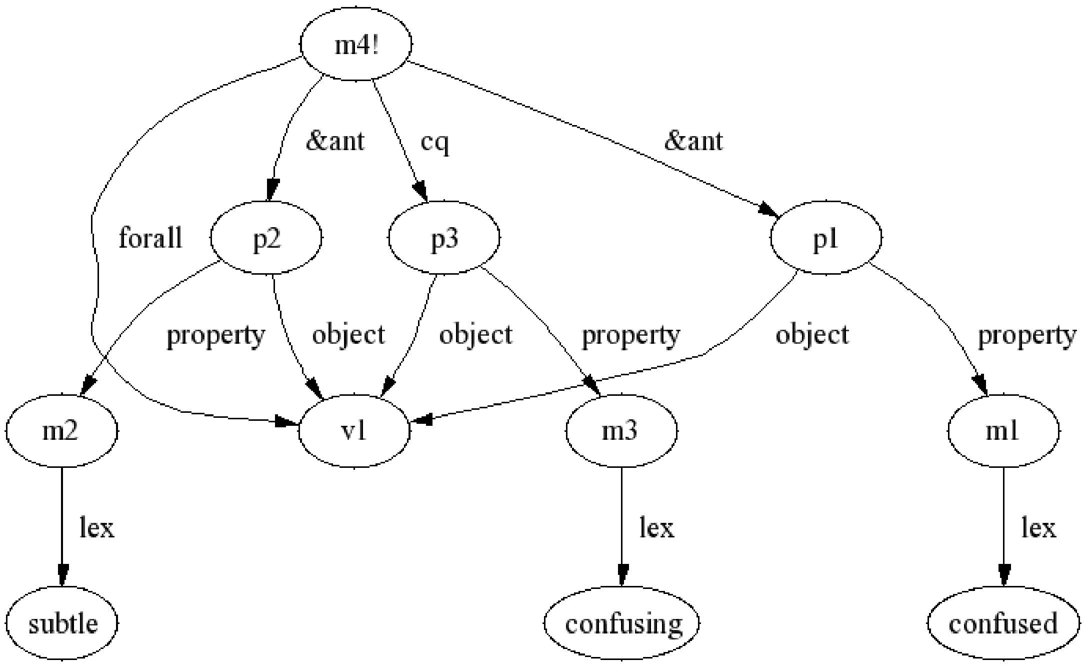

Figure 7: “If something is confused and subtle then it is confusing.”

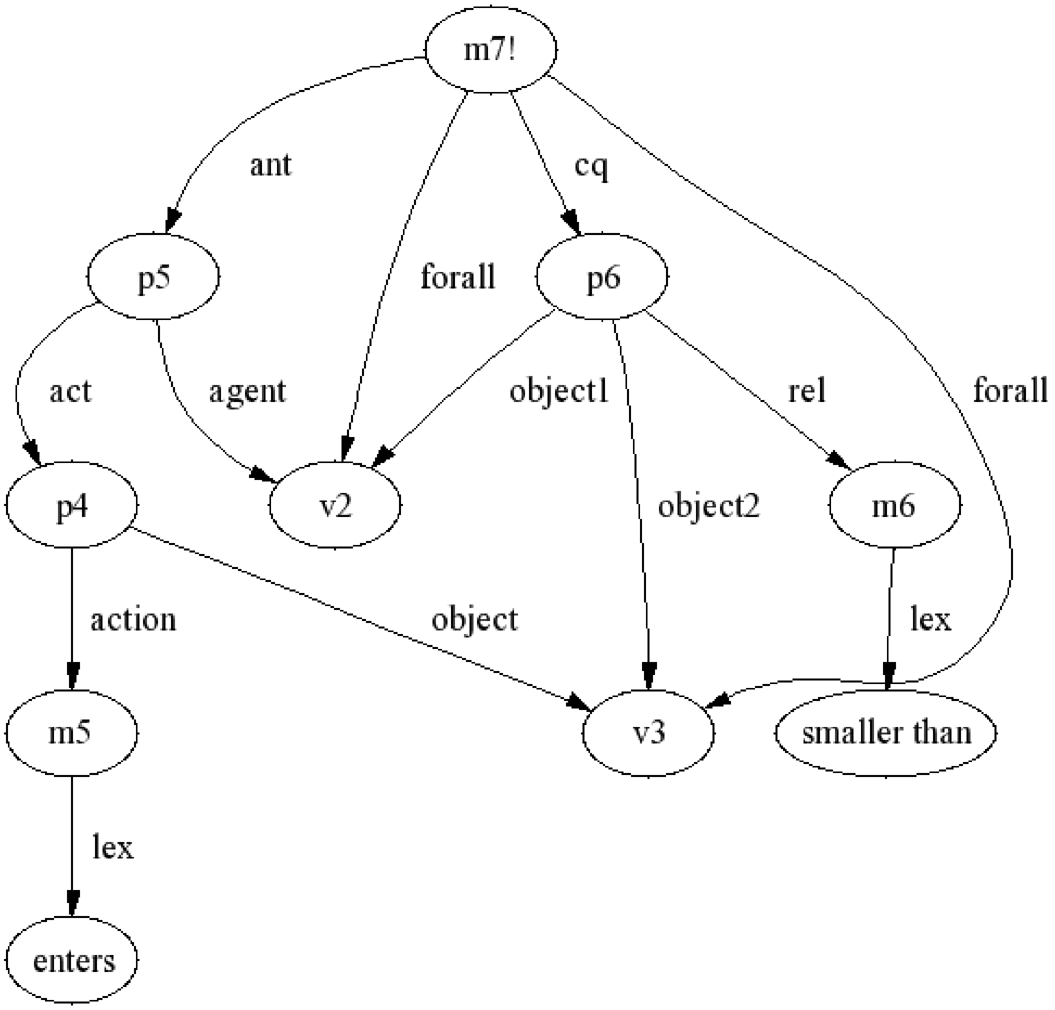

Figure 8: “If x enters y then x is smaller than y.”

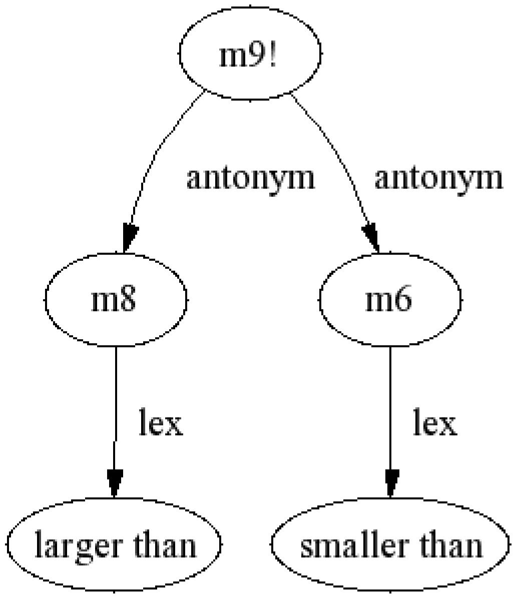

Figure 9: “The concept of ‘larger than’ and ‘smaller than’ are antonyms.”

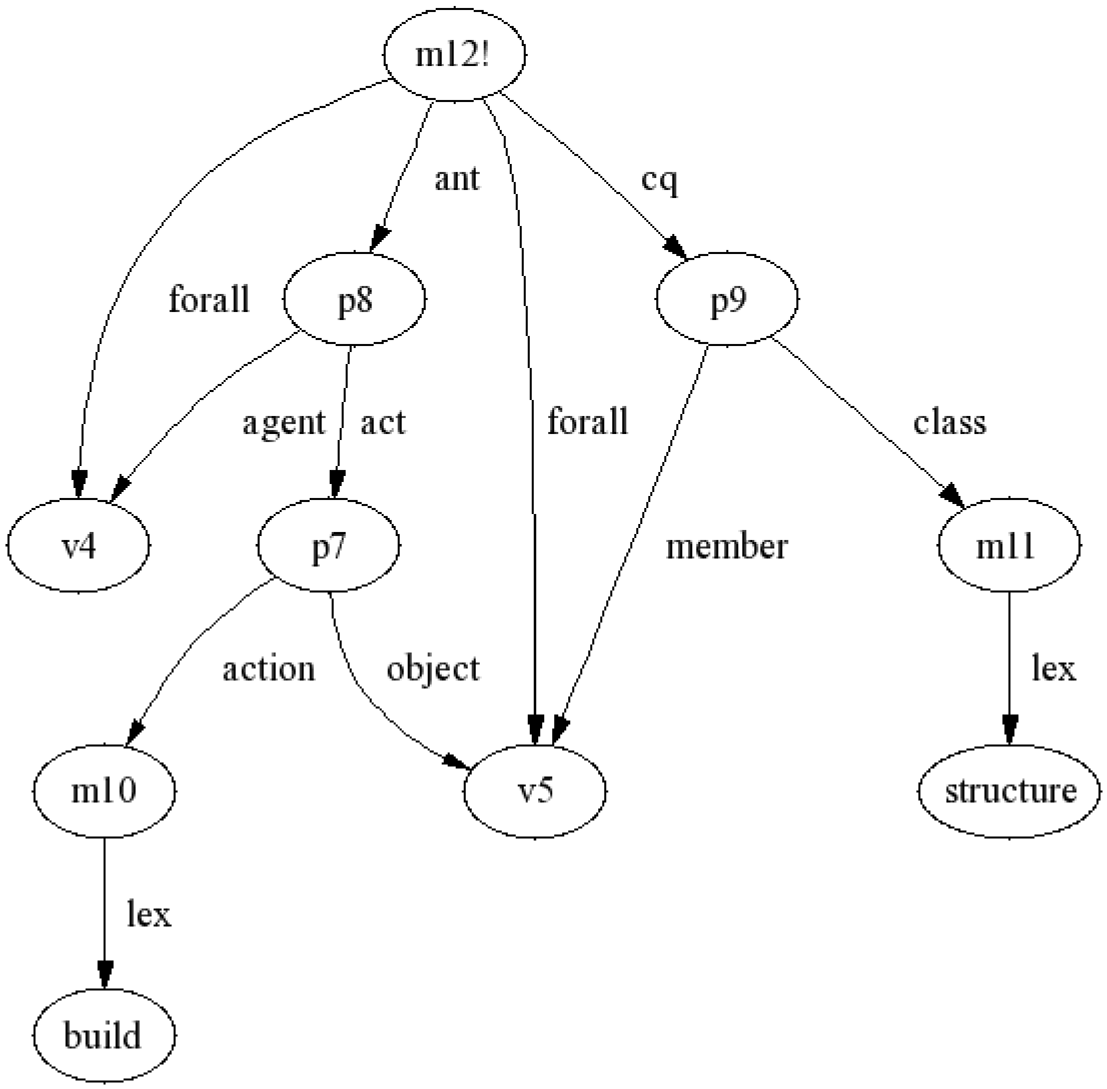

Figure 10: “For all x and for all y, if y builds x then x is a structure.”

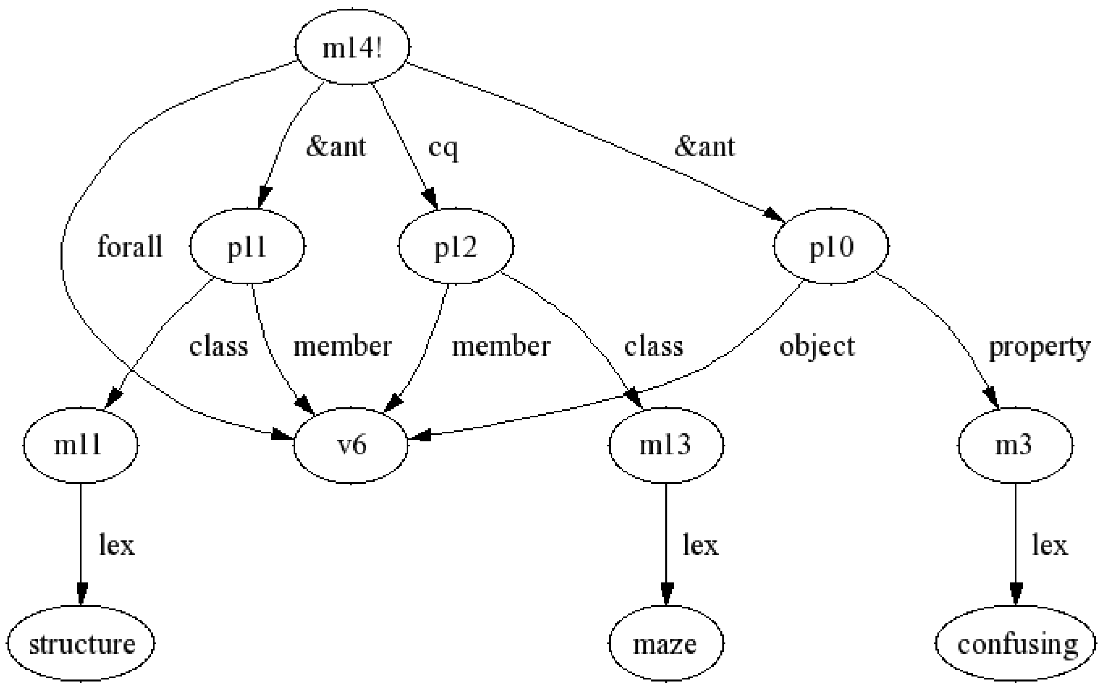

Figure 11: “For all x, if x is a structure and x is confusing then x is a maze.”

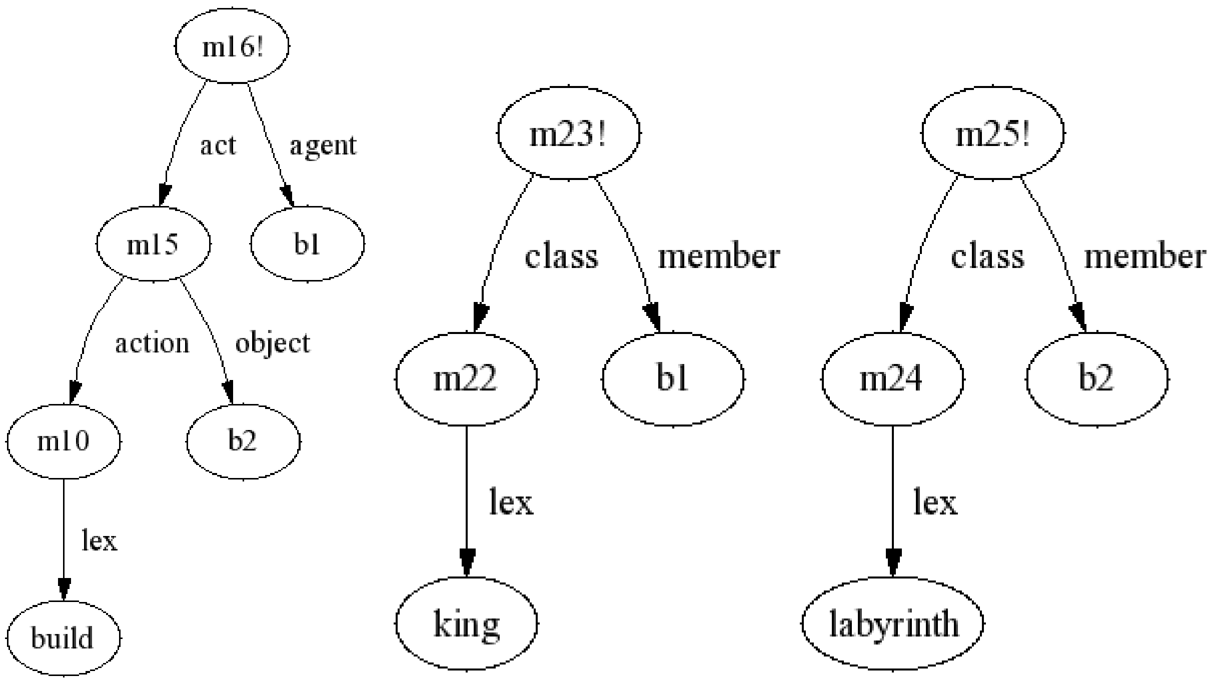

Figure 12: “A king built a labyrinth…”

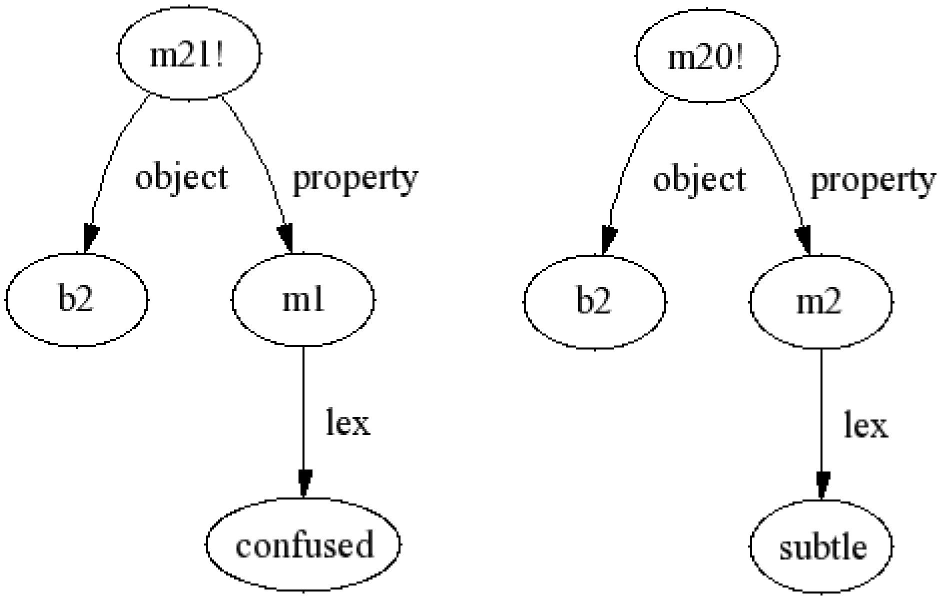

Figure 13: "…a labyrinth so confused and so subtle…"

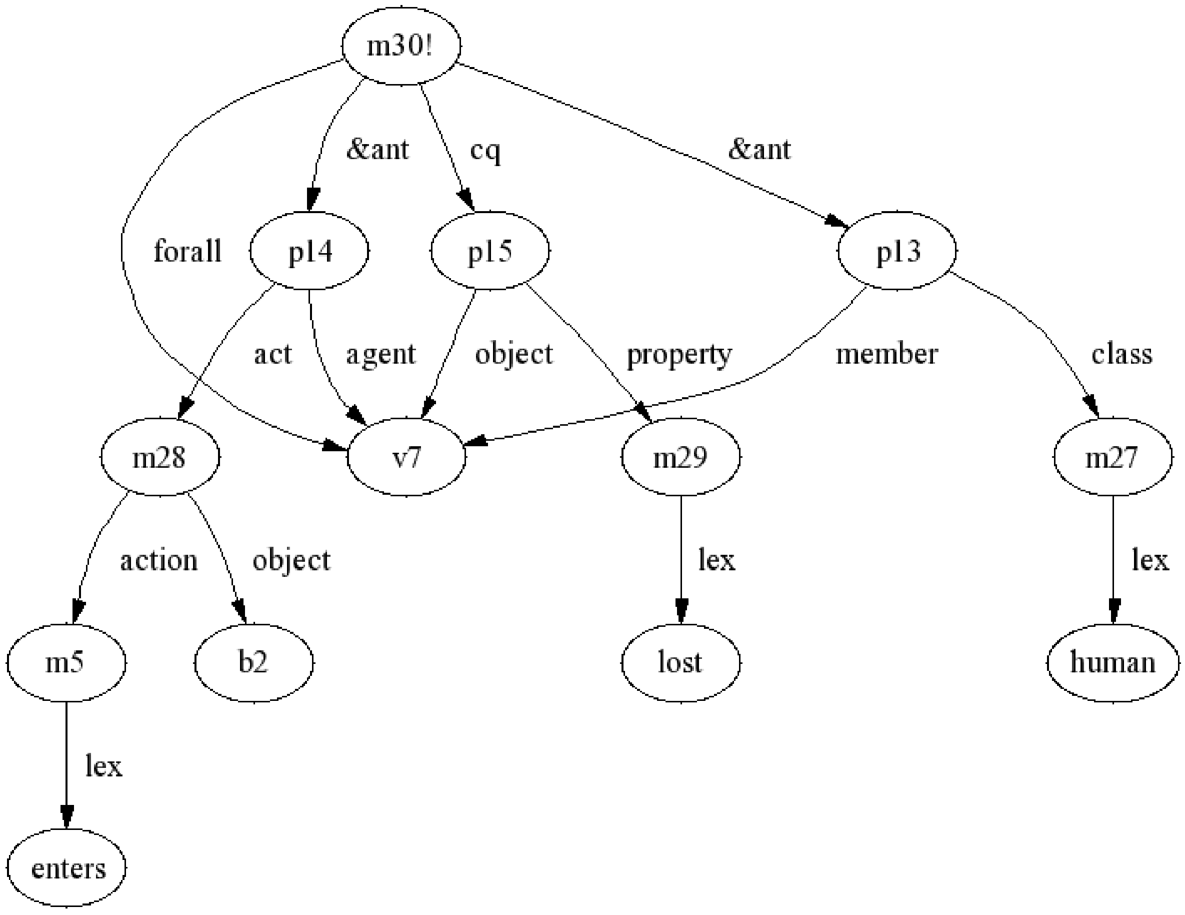

Figure 14: "…that all who entered it lost their way."

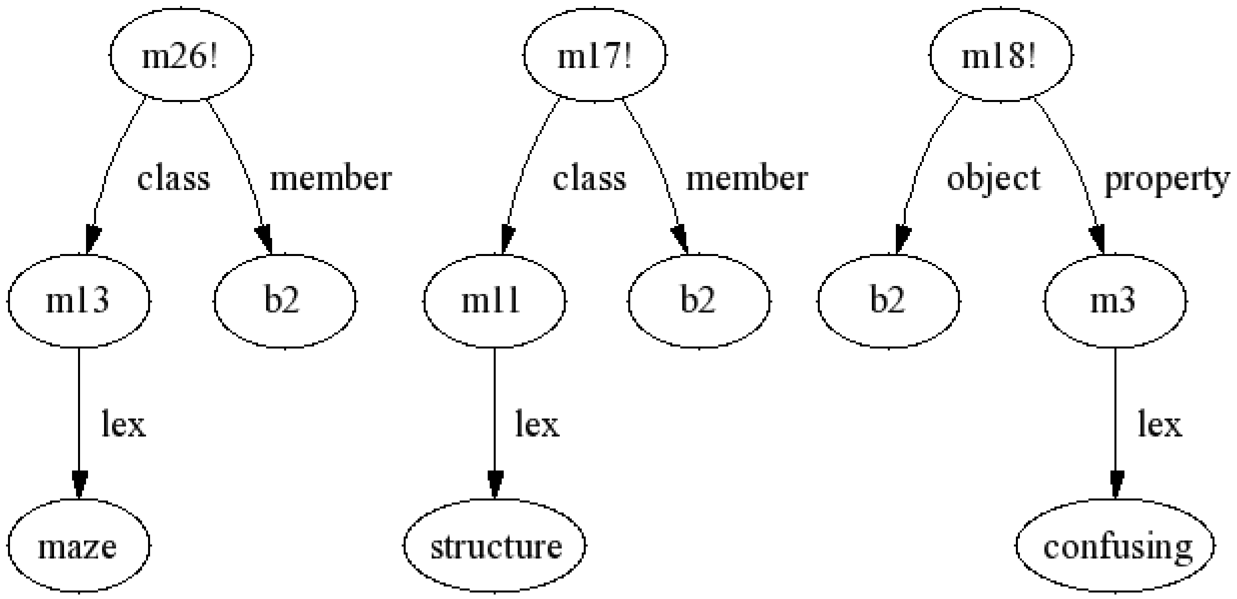

Figure 14: Propositions that Cassie inferred.

Appendix C: Demonstration of Program

```
yeager {~/CSE663/SNePS} > acl
International Allegro CL Enterprise Edition
8.0 [Solaris] (Jun 30, 2006 12:35)
Copyright (C) 1985-2005, Franz Inc., Oakland, CA, USA.  All
Rights Reserved.

This development copy of Allegro CL is licensed to:
   [4549] University at Buffalo

;; Optimization settings: safety 1, space 1, speed 1, debug 2.
;; For a complete description of all compiler switches given the
;; current optimization settings evaluate (explain-compiler-
settings).
;;---
;; Current reader case mode: :case-sensitive-lower
[2] cl-user(3): :ld /projects/shapiro/Sneps/sneps262
;     Loading /projects/shapiro/Sneps/sneps262.cl
;       Loading /projects/shapiro/Sneps/Sneps262/load-sneps.lisp
;         Loading
;            /projects/snwiz/Install/Sneps-2.6.1/load-logical-
pathnames.lisp
Loading system SNePS...10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
SNePS-2.6 [PL:1a 2006/12/04 04:07:47] loaded.
Type `(sneps)' or `(snepslog)' to get started.
[2] cl-user(4): (sneps)


   Welcome to SNePS-2.6 [PL:1a 2006/12/04 04:07:47]

Copyright (C) 1984--2004 by Research Foundation of
State University of New York. SNePS comes with ABSOLUTELY NO
WARRANTY!
Type `(copyright)' for detailed copyright information.
Type `(demo)' for a list of example applications.

   12/15/2006 12:09:40

* (demo "labyrinth.demo")

File /home/unmdue/jsalazar/CSE663/SNePS/labyrinth.demo is now
the source of input.


 CPU time : 0.03
```

```
* ;
================================================================
=======
; FILENAME:     labyrinth.demo
; DATE:         11/25/06
; PROGRAMMER:   Joe Salazar

;; this template version:     snepsul-template.demo-20061005.txt

; Lines beginning with a semi-colon are comments.
; Lines beginning with "^" are Lisp commands.
; All other lines are SNePSUL commands.
;
; To use this file: run SNePS; at the SNePS prompt (*), type:
;
;       (demo "WORD.demo" :av)
;
; Make sure all necessary files are in the current working
; directory or else use full path names.
;
================================================================
=======

; Turn off inference tracing.
; This is optional; if tracing is desired, then delete this.
^(
--> setq snip:*infertrace* nil)
nil



 CPU time : 0.00

*
; Load the appropriate definition algorithm:
;; UNCOMMENT THE ONE YOU *DO* WANT
;; AND DELETE THE OTHER!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
^(
--> load "/projects/rapaport/CVA/STN2/defun_noun.cl")
t



 CPU time : 0.49

* ; ^(load "/projects/rapaport/CVA/STN2/defun_verb.cl")
```

```
; Clear the SNePS network:
;(resetnet)

; OPTIONAL:
; UNCOMMENT THE FOLLOWING CODE TO TURN FULL FORWARD INFERENCING
ON:
;
; ;enter the "snip" package:
; ^(in-package snip)
;
; ;turn on full forward inferencing:
; ^(defun broadcast-one-report (represent)
;    (let (anysent)
;      (do.chset (ch *OUTGOING-CHANNELS* anysent)
;                (when (isopen.ch ch)
;                (setq anysent
;                      (or (try-to-send-report represent ch)
;                          anysent)))))
;    nil)
;
; ;re-enter the "sneps" package:
; ^(in-package sneps)

; load all pre-defined relations:
(intext "/projects/rapaport/CVA/STN2/demos/rels")
Loading file /projects/rapaport/CVA/STN2/demos/rels.


 CPU time : 0.71

*
; load all pre-defined path definitions:
(intext "/projects/rapaport/CVA/mkb3.CVA/paths/paths")
Loading file /projects/rapaport/CVA/mkb3.CVA/paths/paths.
before implied by the path (compose before
                            (kstar (compose after- ! before)))
before- implied by the path (compose (kstar (compose before- !
after))
                             before-)
after implied by the path (compose after
                           (kstar (compose before- ! after)))
after- implied by the path (compose (kstar (compose after- !
before))
                            after-)
sub1 implied by the path (compose object1- superclass- !
subclass
                          superclass- ! subclass)
```

```
sub1- implied by the path (compose subclass- ! superclass
subclass- !
                           superclass object1)
super1 implied by the path (compose superclass subclass- !
superclass
                            object1- ! object2)
super1- implied by the path (compose object2- ! object1
superclass- !
                             subclass superclass-)
superclass implied by the path (or superclass super1)
superclass- implied by the path (or superclass- super1-)


 CPU time : 0.01

*
; BACKGROUND KNOWLEDGE:
; =====================
; (put annotated SNePSUL code of your background knowledge here)

;; =========================================
;; B1: "If something is confused and subtle,
;;      then it is confusing."
;; =========================================
;; Protocol subjects associated confused with confusing. The
;; definitions are similar and the words are lexicographically
;; close.
;;

(describe (assert forall $z
          &ant ((build object *z property (build lex
"confused"))
                (build object *z property (build lex "subtle")))
          cq (build object *z property (build lex
"confusing"))))

(m4! (forall v1)
 (&ant (p2 (object v1) (property (m2 (lex subtle))))
  (p1 (object v1) (property (m1 (lex confused)))))
 (cq (p3 (object v1) (property (m3 (lex confusing))))))

(m4!)

 CPU time : 0.01

*
;; =============================================
;; B2: "If x enters y then x is smaller than y."
;; ================================================
;; The fact that a man can enter the unknown object provides
;; a fair amount of information about its size.
;;

(describe (assert forall $v forall $w
          ant (build agent *v act (build action (build lex
                "enters") object *w))
          cq (build object1 *v rel (build lex "smaller than")
object2 *w)))

(m7! (forall v3 v2)
 (ant
  (p5 (act (p4 (action (m5 (lex enters))) (object v3))) (agent
v2)))
 (cq (p6 (object1 v2) (object2 v3) (rel (m6 (lex smaller
than))))))

(m7!)

 CPU time : 0.00

*
;; ======================================================
;; B2': "If x is smaller than y then y is larger than x."
;; ======================================================
;;
;; The antonym case frame can be used to establish this
relationship.

(describe (assert antonym (build lex "smaller than")
                  antonym (build lex "larger than")))

(m9! (antonym (m8 (lex larger than)) (m6 (lex smaller than))))

(m9!)

 CPU time : 0.01

*
;; ==========================================
;; B3: "If x builds y then x is a structure."
;; ==========================================
;;
;; Most people figured out that it was some sort of building by
the
```

```
;; fact that it was built and that it was large enough for
people to
;; enter it.
;;
;; Merriam-Webster defines a structure as something that is
;; constructed. "Built" is close enough to "constructed."

;(describe (assert forall $p
;                 ant (build action (build lex "build") object
*p)
;                 cq (build member *p class (build lex
"structure"))))

(describe (assert forall ($m $n)
          ant (build agent *m act
                     (build action (build lex "build") object
*n))
          cq (build member *n class (build lex "structure"))))

(m12! (forall v5 v4)
 (ant
  (p8 (act (p7 (action (m10 (lex build))) (object v5))) (agent
v4)))
 (cq (p9 (class (m11 (lex structure))) (member v5))))

(m12!)

 CPU time : 0.00

*
;;
==============================================================
;; B4: "If x is a structure and x is confusing then x is a
maze."
;;
==============================================================

(describe (assert forall $p
          &ant ((build object *p property (build lex
"confusing"))
                (build member *p class (build lex "structure")))
          cq (build member *p class (build lex "maze"))))

(m14! (forall v6)
 (&ant (p11 (class (m11 (lex structure))) (member v6))
  (p10 (object v6) (property (m3 (lex confusing)))))
 (cq (p12 (class (m13 (lex maze))) (member v6))))
```

```
(m14!)

 CPU time : 0.01

*




; CASSIE READS THE PASSAGE:
; =========================
; (put annotated SNePSUL code of the passage here)

;==========================================
; P1: "A king built a labyrinth..."
;==========================================

(describe (add agent #king act (build action
          (build lex "build") object #labyrinth)))

(m17! (class (m11 (lex structure))) (member b2))
(m16! (act (m15 (action (m10 (lex build))) (object b2))) (agent
b1))

(m17! m16!)

 CPU time : 0.02

*
(describe (add member *king class (build lex "king")))

(m23! (class (m22 (lex king))) (member b1))

(m23!)

 CPU time : 0.01

* (describe (add member *labyrinth class (build lex
"labyrinth")))

(m25! (class (m24 (lex labyrinth))) (member b2))

(m25!)

 CPU time : 0.01
```

```
*
;================================================
; P2: ...a labyrinth so confused and so subtle...
;================================================

(describe (add object *labyrinth property (build lex
"confused")))

(m21! (object b2) (property (m1 (lex confused))))

(m21!)

 CPU time : 0.01

* (describe (add object *labyrinth property (build lex
"subtle")))

(m26! (class (m13 (lex maze))) (member b2))
(m20! (object b2) (property (m2 (lex subtle))))
(m18! (object b2) (property (m3 (lex confusing))))

(m26! m20! m18!)

 CPU time : 0.01

*
;=====================================================
; P3: ...that men who entered it would lose their way.
;=====================================================

(describe (add forall $y
          &ant ((build member *y class (build lex "human"))
                (build agent *y act (build action (build lex
                "enters") object *labyrinth)))
          cq (build object *y property (build lex "lost"))))
(m30! (forall v7)
 (&ant
  (p14 (act (m28 (action (m5 (lex enters))) (object b2))) (agent
v7))
  (p13 (class (m27 (lex human))) (member v7)))
 (cq (p15 (object v7) (property (m29 (lex lost))))))

(m30!)

 CPU time : 0.01

*
```

```
; Ask Cassie what "WORD" means:
;; UNCOMMENT THE ONE YOU *DO* WANT
;; AND DELETE THE OTHER!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
^(
--> defineNoun "labyrinth")
 Definition of labyrinth:
 Possible Class Inclusions: structure, maze,
 Actions performed on a labyrinth: king build,
 Possible Properties: confused, subtle, confusing,
nil



 CPU time : 0.21

End of /home/unmdue/jsalazar/CSE663/SNePS/labyrinth.demo
demonstration.
```

References

Borges, Jorge Luis & Hurley, Andrew (trans.) (1998), *Collected Fictions* (Penguin: New York)

Rapaport, William J., & Ehrlich, Karen (2000), "A Computational Theory of Vocabulary Acquisition", in Lucja M. Iwanska & Stuart C. Shapiro (eds.), Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language (Menlo Park, CA/Cambridge, MA: AAAI Press/MIT Press): 347-375.

Rapaport, William J., & Kibby, Michael W. (2002b), "ROLE: Contextual Vocabulary Acquisition: From Algorithm to Curriculum".

Shapiro, Stuart C. and Rapaport, William J. (1995), "An Introduction to a Computational Reader of Narrative" [PDF], in Judith Felson Duchan, Gail A. Bruder, & Lynne E. Hewitt (eds.), *Deixis in Narrative: A Cognitive Science Perspective* (Hillsdale, NJ: Lawrence Erlbaum Associates): 79-105.

Shapiro, Stuart C. (1989), "The CASSIE Projects: An Approach to Natural Language Competence" [Postscript], in João P. Martins & Ernesto M. Morgado (eds.), *EPIA 89: 4th Portugese Conference on Artificial Intelligence Proceedings*, Lecture Notes in Artificial Intelligence 390 (Berlin: Springer-Verlag): 362-380.