A SNePS Representation for Computationally
Determining the Meaning of "Aphelion Point" from Context

By Bill Duncan (wdduncan@buffalo.edu)
CSE 663: Advanced Knowledge Representation
December 8, 2008

Abstract

Contextual Vocabulary Acquisition (CVA) is an interdisciplinary research project directed by William J. Rapaport, of the University at Buffalo's Department of Computer Science and Engineering, and Michael W. Kibby, of the University at Buffalo's Department of Learning and Instruction. The goals of CVA are to (1) develop a deeper understanding of how human readers use contextual information to acquire meanings of unknown words, and (2) to use this research to develop strategies which will help students learn new words. In relation to goal (1), CVA uses SNePS (a propositional semantic network based knowledge representation and reasoning system) to develop algorithms for inferring the meanings of unknown words. These algorithms are then implemented in a computational agent named "Cassie" who then "guesses" (i.e., computes) the meaning of an unknown word contained in a passage (also represented in SNePS).
In this paper, I describe my CVA project for Professor Rapaport's Fall 2008 CSE 663, "Advanced Knowledge Representation", class. The purpose of this project was to choose a passage containing an unknown word; represent the passage in SNePS; develop algorithms for computing the meaning of the unknown word; implement these algorithms in Cassie; and finally have Cassie read the passage and compute a meaning. The word I chose was "aphelion", and using the algorithms I developed Cassie inferred that "aphelion" was equivalent to the farthest point in Earth's orbit. While this result is correct, it also became clear that more work was needed on Cassie's noun definition algorithm, and in developing case frames for representing distance and time relations.

1. Introduction

Contextual Vocabulary Acquisition (hereafter CVA) is an interdisciplinary research project directed by William J. Rapaport, of the University at Buffalo's Department of Computer Science and Engineering, and Michael W. Kibby, of the University at Buffalo's Department of Learning and Instruction[1]. The goals of CVA are twofold. The first is to develop a deeper understanding of how human readers use contextual information to acquire meanings of unknown words. The second is to use this research to develop strategies which will help students learn new words[2].

This paper is aimed at the first of these two goals (i.e., the goal of understanding how humans learn words from contextual clues). It describes the use a computational agent named "Cassie" to read a passage containing an unknown word. Cassie then to applies her background knowledge to the passage and "guesses" (i.e., computes) a meaning for the unknown word. Both the passage and Cassie's background knowledge were represented using SNePS: a propositional semantic network based knowledge representation and reasoning system developed by Stuart Shapiro and the SNePS Research Group at the University at Buffalo[3].

The work presented here is based on the final project for Professor Rapaport's Fall 2008 CSE 663 (Advanced Knowledge Representation) class. The project consisted of three major phases:

1. The selection of an unknown word and an actual passage using the word.
2. An informal experiment in which human subjects were asked to read the passage containing the unknown word, and verbalize their reasoning process for guessing the meaning of the unknown word.

[1] http://www.cse.buffalo.edu/~rapaport/CVA/cva.html (2008.01.10)

[2] http://www.cse.buffalo.edu/~rapaport/CVA/cvadescription.html (2002.08.22)

[3] http://www.cse.buffalo.edu/sneps (2008.08.21)

3. Using the information gained from phased two to represent the passage containing the unknown word in the SNePS knowledge-representation and reasoning system, and seeing if Cassie (the SNePS computational cognitive agent) could determine (i.e., compute) a meaning of the unknown word from the context of the passage.

2. The Definition and Passage for "aphelion point"

The first phase of the CVA project consisted of two parts: choosing of an unknown word, and finding of an actual passage which used the unknown word. The word I chose was "aphelion", or (as things turned out) the phrase "aphelion point". Since I did not know the definition, I consulted Merriam-Webster's 11th Collegiate Dictionary. It defined "aphelion" as "… the point in the path of a celestial body (as a planet) that is farthest from the sun".[4]

For the second part (i.e., finding a passage) I did an internet search, and found the following passage on the "Windows to the Universe" website[5]:

> Earth reaches perihelion in early January each year, and passes through its aphelion point near the start of July. ... Earth is about 3% further from the Sun at aphelion than it is at perihelion.

However, for the purposes of the project this passage was not adequate for two reasons. First, it contained the word "perihelion" (i.e., the closest point of a planet's orbit) which, if the reader was not already familiar with the word, would also have to be defined. Second, the information concerning percentages (i.e., "… 3% further …) would have been difficult to represent in SNePS. Thus, as per Professor Rapaport's suggestion, the following two changes were made. First, the word "perihelion" was removed from the first sentence and the definition for "perihelion" was inserted in its place. Second, the second sentence was dropped. This resulted in the following passage:

[4] Meriam-Webster's 11th Collegiate Dictionary.

[5] The passage is a modified version of the information found at: http://www.windows.ucar.edu/tour/link=/physical_science/physics/mechanics/orbit/perihelion_aphelion.html&eed=high (2008.12.08)

> Earth reaches its closest point to the sun in early January each year, and passes through its aphelion point near the start of July.

One other potential problem concerned whether "aphelion" was being used as a noun or an adjective. The Merriam-Webster definition (cited above) classified "aphelion" as a noun, but in the original passage, the word "aphelion" was acting as an adjective in the phrase "aphelion point". Technically, this is redundant, since the definition for "aphelion" states that it is a "point". However, I wanted to remain true passage. So, I broadened my CVA project to pertain to either the meaning of the noun "aphelion" or the phrase "aphelion point".

3. Testing the Passage on Human Subjects

3.1 Description of the Informal Experiment

The second phase of the CVA project involved an informal experiment on human subjects. The purpose of this informal experiment was to gain data on how human subjects go about determining the meaning for "aphelion point". This information would then be applied to developing an algorithm.

The informal experiment consisted of the following steps:

1. I would confirm that the subject did not know the meaning of the word "aphelion".
2. I would have the subject read the passage either aloud or silently.
3. I would have the subject verbalize their reasoning process for determining the meaning of "aphelion" or "aphelion point", and I would write down the subject's statements.

3.2 Results of the Informal Experiment

Of the five subjects tested, all five settled on a definition equivalent with "farthest point from sun" (e.g., "most distant point from the sun", "the point farthest from the sun", etc…). In general their reasoning processes can be summed up as follows:

1. The phrase "closest point" implies something about the orbit.
2. January is in the beginning of the year.
3. July is half-way point in year.
4. Since July is half-way through the calendar year, "aphelion" is somehow "opposite" of January.
5. Thus, the "aphelion point" is the opposite of the “closest point”.
6. So, the "aphelion point" must be furthest point in the orbit.

4. Representing the Passage in SNePS

4.1 Further Refinement of the Passage

The third phase of the CVA project focused on representing the passage which the human subjects read in SNePS, and for Cassie to infer a meaning for “aphelion point”. However, before it was possible to do this, the following three problems needed to be addressed. First, the word “Earth” is left out of the second part of the passage (i.e., “… and its aphelion point ...”). While human readers easily recognized this part of the passage refers to “Earth”, I was not sure how to represent this in SNePS. So, to work around this, I broke the passage up into two sentences with each sentence beginning with the word “Earth”.

Second, the passage contained the phrases “in early” and “near the start”. While these phrases are important for indicating time periods within the months to which they refer, they do not appear to be germane to the meaning of the overall passage. Thus, they were eliminated.

Finally, the first part of the passage contained the phrase “to the sun”. While this phrase would have been important if the passage was not talking about Earth, it is extraneous information. For, all the human subjects were cognizant of the fact that if Earth “reaches point *x*”, that point is in relation to the sun. Thus, the phrase “to the sun” was eliminated.[6]

[6] It is important to note that if the passage was about an unknown celestial body the phrase “to the sun” may have played a more essential role.

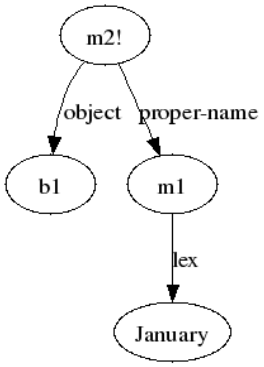
After these further modifications, the final passage to be represented in SNePS was: [7]

> Earth reaches its closest point in January.
> Earth reaches its aphelion point in July.

4.2 Case Frames

Before proceeding to the SNePS representation of the passage, it is necessary to say a little about case frames. Basically, a case frame in SNePS is way of representing a proposition as a graph. The nodes represent entities, and the arcs (or edges) represent relations. This use of case frames allows for Cassie to infer information not explicit in the representation. That is, since case frames are representations of propositions, implicit information can be deduced by making various inferences on the propositions.

As a simple example, consider a case frame which represents the proposition that "there is something named 'January'". In order to represent this proposition, one can use an object/proper-name case frame which graphically looks like:

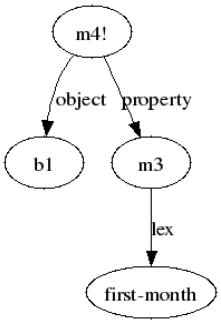


This graph (i.e., case frame) is then represented in SNePS by using the SNePS user language (SNePSUL) to define the relationships and entities. In this example, the SNePSUL code for representing the object/proper-name case frame above is:

[7] It might be interesting to do some more experiments in which human subjects read the passage with the phrases "in early", "near the start", and "to the sun" removed to see if this had a different result. However, I am doubtful that the result would differ.

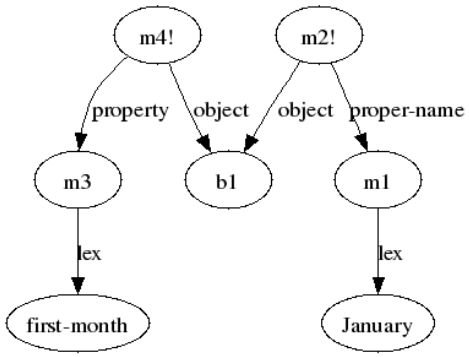
(assert object #january
proper-name (build lex January))

To see how inference works, let us say that the object #janurary has the property of being the first month in the year. Using an object/property case frame, the SNePSUL code and the graphical representation would like this[8]:

SNePSUL code: (assert object *january property (build lex first-month))	

From this new proposition and the previous one, one thing which should be obvious is that the object (i.e., *january) which has the proper name "January" also has the property of being the first month. However, this information is not explicit, but implicit. That is, it is not explicit in either of the two propositions, but it deduced by making inferences on both propositions taken together. When considered from the standpoint of case frames, the implicit information can be inferred by tracing the arcs. This is easily seen when you look at a graphical representation combining the two propositions (note: *january is represented by node b1):

[8] "*january" is used in order to tell SNePS that we are referring to the previously created entity "#january".



This information can also be queried using various SNePSUL commands such as "find" and "deduce". Because it is beyond the scope of this paper, I will not give examples of the SNePSUL code for doing this, but hopefully this example gives you a feel for case frames and their use. For a full description of SNePSUL, you can consult the "SNePS 2.7 User Manual" located at http://www.cse.buffalo.edu/sneps/Manuals/manual27.pdf.

4.3 Representing the Passage in SNePS

As mentioned in the previous section, it is necessary for information to be represented as propositions (i.e., case frames) in order for Cassie to make inferences. Thus, in order for Cassie to infer a meaning for "aphelion point", the two sentences were represented as the following eight propositions (P1 – P4 are associated the first sentence, and P5 – P8 are associated with the second sentence):

I. Earth reaches its closest point in January.
   P1 There is something named "Earth".
   P2 Something has the property "closest point".
   P3 Earth has the thing "closest point" via the relationship "distance point".
   P4 Earth reaches the "closest point" in January.

II. Earth reaches its aphelion point in July.
   P5 There is something named "Earth".
   P6 Something has the property "aphelion point".
   P7 Earth has the thing "aphelion point" via the relationship "distance point".
   P8 Earth reaches the "aphelion point" in July.

From these propositions, I determined the following cases frames were needed to represent the passage:

1. object/proper-name
2. object/property
3. object/rel/possessor
4. agent/act/action/object/time

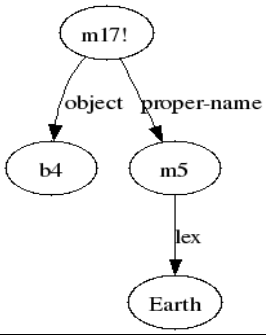
Of the case frames above, the first three are standard while the last one is a non-standard case frame. The distinction between standard and non-standard case frames is that standard case frames are recognized by Cassie's noun definition algorithm. Ideally, I would have liked to use only standard case frames to represent the passage, but the need to represent time information did not permit this.

Using these case frames, the propositions were then represented in SNePS using SNePSUL. Below, I have listed both the SNePSUL code and the case frame used for each of the propositions P1 – P8. I have also included comments and a graphical representation, and for ease of identification the propositions are italicized. For a fuller description of the syntax and semantics of these case frames, please see Appendix A:[9]

[9] It is important to note that the SNePSUL "add" command was used. This necessary to make Cassie use forward inferencing. For a description of forward inferencing, please consult the "SNePS 2.7 User Manual" located at http://www.cse.buffalo.edu/sneps/Manuals/manual27.pdf.

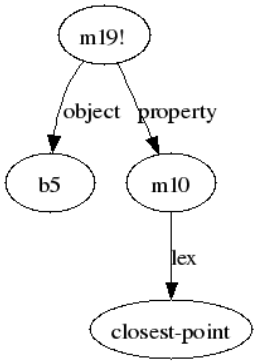
I. FIRST SENTENCE: Earth reaches its closest point in January.

P1. There is something named "Earth".

SNePSUL code:	Case frame: object/proper-name
(add object #earth proper-name (build lex Earth))	

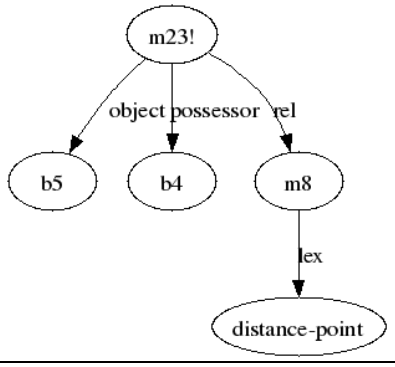
Comment for P1: The standard object/proper-name case frame was used to represent proposition P1. In the SNePSUL code, the entity #earth is associated with node b4. Think of names beginning with the "#" sign to be analogous to defining constants in a programming language. The name "earth" I associated with the "#" for my (the human) readability. I could have named it anything I wanted. The b4 was assigned by the SNePS system. The node at the end of a lex arc (i.e., "Earth") identifies the word in the lexicon used to talk about b4.

P2. Something has the property "closest point".

SNePSUL code:	Case frame: object/property
(add object #closest-point property (build lex closest-point))	

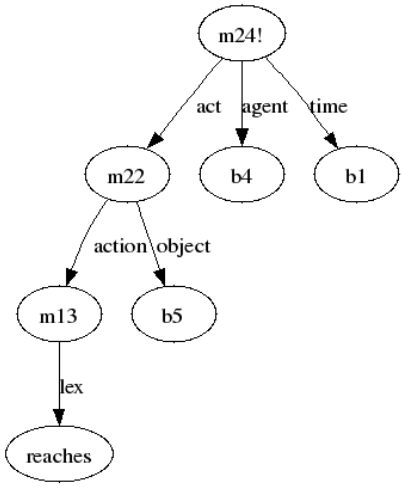
Comment for P2: The standard object/property case frame was used to represent proposition P2. In the SNePSUL code, the enitity #closest-point is associated with node b5. (For a description of the lex arc see comment for P1).

P3. Earth has the thing "closest point" via the relationship "distance point".

SNePSUL code:	Case frame: object/rel/possessor
(add object *closest-point rel (build lex distance-point) possessor *earth)	

Comment for P3: The standard object/rel/possessor case frame was used to represent proposition P3. What this proposition says is that the entity *earth (node b4) possesses (or has) a thing *closest-point (node b5). The concept of "distance-point" (node m8) identifies how *earth and *closest-point are related. It is important to mention that the name "distance-point" does nothing special for reasoning process in SNePS. Rather, I named the relationship "distance-point" as way to indicate to the human reader that "closest-point" is a kind of "distance-point". In the SNePSUL code, the entities *closest-point and *earth refer to an entities previously created using a "#". That is, #closest-point was previously created in P2, and #earth was previously created in P1. (For a description of the lex arc see comment for P1).

P4. Earth reaches the closest point in January.

SNePSUL code:	Case frame: agent/act/action/object/time
(add agent *earth act (build action (build lex reaches) object *closest-point) time *january)	

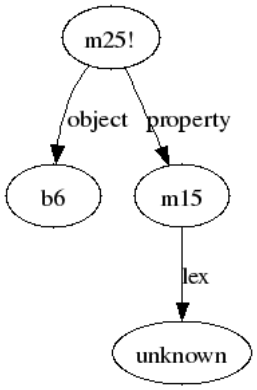
Comment for P4: The non-standard agent/act/action/object/time case frame was used to represent proposition P4. This non-standard case frame was based on the standard agent/act/action/object case frame, but with a time relation added to indicate when the action was done by the agent. What this proposition says is that *earth (node b4) performs the action of reaching (node m13) the *closest-point (node b5) during *january (node b1). (For a description of "*" see comment for P3. For a description of the lex arc see comment for P1).

II. SECOND SENTENCE: Earth reaches its aphelion point in July.

P5. There is something named "Earth".

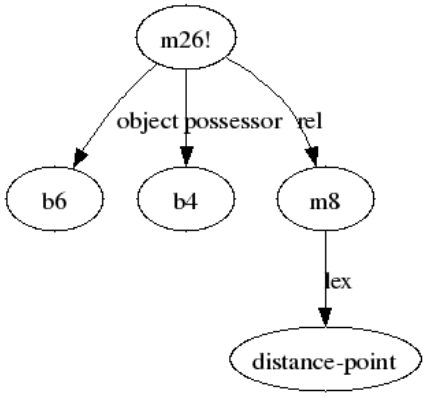
Comment for P5: Since "Earth" was previously named in P1, it was not necessary to represent this proposition.

P6. Something has the property "aphelion point".

SNePSUL code:	Case frame: object/property
(add object #aphelion-point property (build lex unknown))	

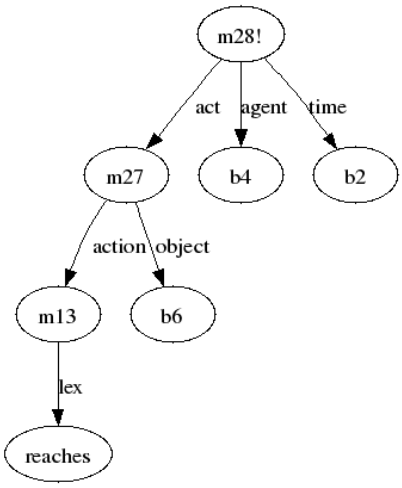
Comment for P6: The standard object/property case frame was used to represent proposition P6. What this proposition says is that the entity #aphelion-point (node b6) has the property of being unknown (node m15). Notice that the use of the lex arc is playing a slightly different role than in the previous case frames. Normally, the word at the end of a lex arc represents the word in the lexicon used talk about the object (see comment for P1). However, in this case, it identifies that the meaning of entity #aphelion-point is not known. That is, it is the word for which Cassie is to compute a meaning. (For a description of "#" see comment for P1).

P7. Earth has the thing "aphelion point" via the relationship "distance point".

SNePSUL code:	Case frame: object/rel/possessor
(add object *aphelion-point rel (build lex distance-point) possessor *earth)	

Comment for P7: The standard object/rel/possessor case frame was used to represent proposition P7. What this proposition says is that the entity *earth (node b4) possesses (or has) a thing *aphelion-point (node b6). The concept of "distance-point" (node m8) identifies how *earth and *aphelion-point are related. (For a description of the "distance-point" relationship see comment for P3. For a description of the lex arc see comment for P1).

P8. Earth reaches the "aphelion point" in July.

SNePSUL code:	Case frame: agent/act/action/object/time
(add agent *earth act (build action (build lex reaches) object *aphelion-point) time *july)	

Comment for P8: The non-standard agent/act/action/object/time case frame was used to represent proposition P8. This non-standard case frame was based on the standard agent/act/action/object case frame, but with a time relation added to indicate when the action was done by the agent. What this proposition says is that *earth (node b4) performs the action of reaching (node m13) the *aphelion-point (node b6) during *july (node b2). (For a description of "*" see comment for P3. For a description of the lex arc see comment for P1).

4.4 Representing Background Information

After representing the passage in SNePS, the next step was to represent pertinent background information. The reason for this is that it is not enough to only represent the passage in SNePS. The passage by itself does provide enough information for Cassie to make an educated guess about the meaning of "aphelion point". What is also needed some prior background knowledge for Cassie to use in conjunction with the passage. Recall, part of the goal of the CVA project is gain a deeper understanding of how human readers use contextual clues to figure out the meanings of unknown words. Thus, if Cassie is going to be used as the cognitive agent, some background information has to be supplied which can act as contextual clues.

A difficulty with this part of the project was trying to gauge how much background information needed to be represented. If too little was represented, the inference made by Cassie would be too simple and not helpful for understanding human subjects. If too much was represented, the project would run the risk of not being able to be completed by the end of the semester. So, by working with Professor Rapaport, the following six prior knowledge (PK) propositions were decided upon:

PK1. Something is named January

PK2. Something is named July

PK3. if something has proper name "Earth", then there is something which has the property "farthest point" and "farthest point" is possessed by "Earth" via the relationship "distance point"

PK4. for all points x and y, if x has property "closest point" and y has property "farthest point" then point x is the opposite of point y

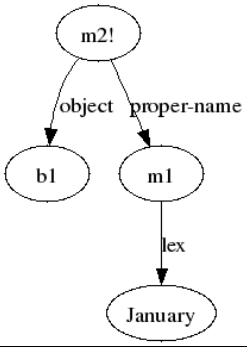
PK5. for all points x, y, and planets p, if p reaches point x in January and p reaches point y in July then point x is the opposite of point y

PK6. for all x, y, and z, if x is the opposite of y and x is the opposite of z and z has the property "unknown", then y is equiv to z

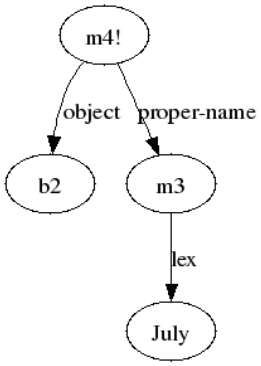
One important difference between these propositions and the propositions used for representing the passage is the use of rules in propositions PK3 – PK6. What these rules specify is how Cassie going to connect the new information contained in the read passage with old information already present. That is, they specify how Cassie is going to infer new propositions.

The forall/ant/cq case frame was used for representing these rules, and for readability, I have written the rules in pseudo-first order logic. The SNePSUL representations for PK1 – PK8 are listed below along with a comment section for each. Again, the propositions are italicized for ease of identification. This is similar to the approach used in the previous section (i.e., P1 – P8), but I will pay less attention to describing the SNePSUL code since much of that has already been covered. More emphasis, then, will be placed on describing why the prior knowledge was included and the purpose it served.

PK1. Something is named January.

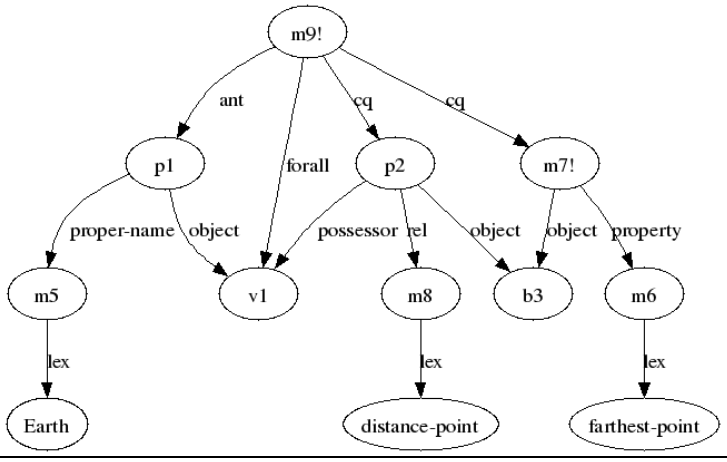
SNePSUL code:	Case frame: object/proper-name
(assert object #january proper-name (build lex January))	

PK2. Something is named July.

SNePSUL code:	Case frame: object/proper-name
(assert object #july proper-name (build lex July))	

Comments for PK1 and PK2: The purpose of PK1 and PK2 was to provide Cassie with some basic facts about the world: namely, that there exists to things (we call them months) named January and July. Also, since this kind of prior knowledge was exhibited by the human subjects, these propositions basic facts were warranted.

PK3. if something has proper name "Earth", then there is something which has the property farthest point" and "farthest point" is possessed by "Earth" via the relationship "distance point"

SNePSUL code:	Case frame: forall/ant/cq
(assert forall $x &ant(build object *x proper-name (build lex Earth)) cq ((build object #farthest-point property (build lex farthest-point)) (build object *farthest-point rel (build lex distance-point) possessor *x)))	

Comment for PK3: The purpose of this rule was to allow Cassie to infer that Earth reaches a farthest point in its orbit. The important thing to notice about this rule is that it is not a fact in the same way which PK1 and PK2 are facts. In order for Cassie to use this rule new information has

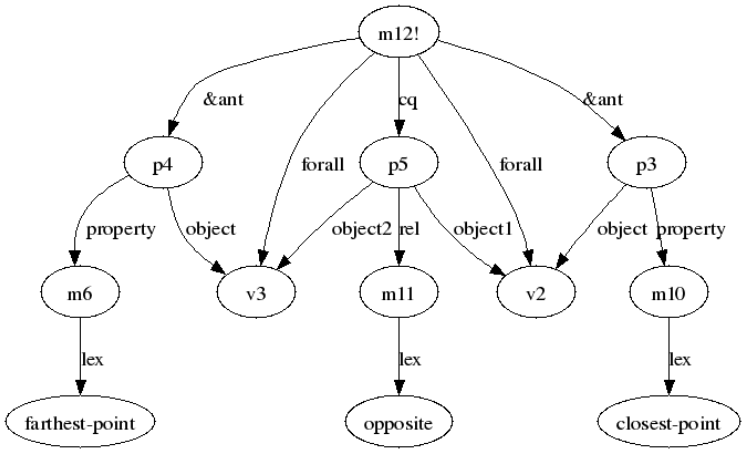
to be acquired that there is in fact something named Earth. Once this information is acquired, Cassie can then infer two new propositions: (1) there is something called a "farthest point", and (2) Earth has (or possesses) the thing "farthest point" as a kind of "distance point" relationship. One criticism which may be apt here, is that this rule grants to much background information. In response to this, I counter that this kind of background knowledge was present in the human subjects, and moreover most college students are aware that Earth has an elliptical orbit that consists of a farthest point from the sun.

PK4. for all points x and y, if x has property "closest point" and y has property "farthest point", then point x is the opposite of point y

SNePSUL code:

```
(assert forall ($x $y)
   &ant(build object *x
           property
             (build lex closest-point))
   &ant(build object *y
           property
             (build lex farthest-point))
   cq (build rel (build lex opposite)
             object1 *x
             object2 *y))
```

Case frame: forall/ant/cq



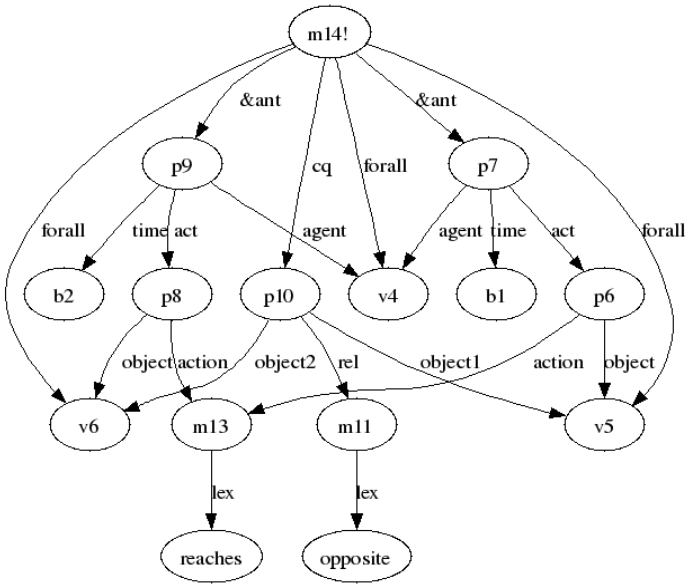
Comment for PK4: The purpose of this rule was to allow Cassie to infer that the concepts expressed by "closest-point" and "farthest-point" are opposite extremes of one another. That is, if one point is the closest and another point is the farthest, then when the two points are pictorial represented on some kind of scale (e.g., a ruler) the two points will be on opposite ends of the scale. Again, it is important to notice that this information is not explicit. Rather, Cassie would infer it once presented with propositions containing the properties of "closest-point" and "farthest-point".

PK5. for all points x, y, and planets p, if p reaches point x in January and p reaches point y in July, then point x is the opposite of point y

SNePSUL code:

```
(assert forall ($planet $x $y)
   &ant(build time *january
           agent *planet
           act (build action
                       (build lex reaches)
                     object *x))
  &ant(build time *july
           agent *planet
           act (build action
                       (build lex reaches)
                    object *y))
  cq (build rel (build lex opposite)
              object1 *x
              object2 *y))
```

Case frame: forall/ant/cq



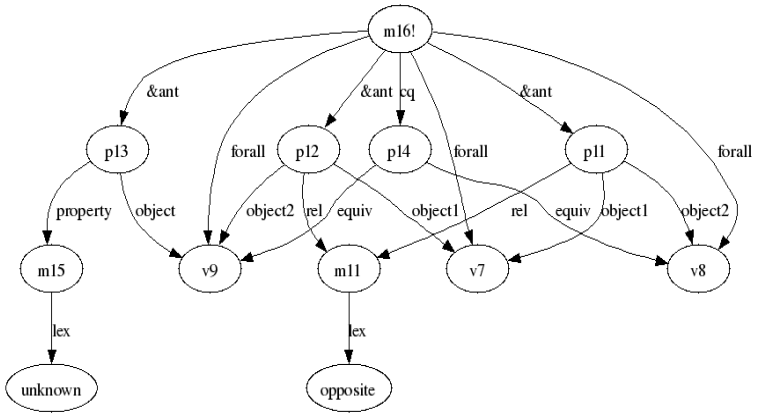
Comment for PK5: Similar to PK4, the purpose for this rule was to allow Cassie to infer that two things are opposite extremes of one another. However, whereas PK4 pertained to distance relationships, PK5 pertains to the twelve month cycle of the year. That is, since January is the first month of the year and July is the seventh month, the two are at opposite extremes of another in the calendar year.

PK6. for all x, y, and z, if x is the opposite of y and x is the opposite of z and z has the property "unknown", then y is equiv to z

SNePSUL code:

```
(assert forall ($x $y $z)
  &ant(build rel
                (build lex opposite)
          object1 *x
          object2 *y)
  &ant(build rel
                (build lex opposite)
           object1 *x
           object2 *z)
  &ant(build object *z
           property
              (build lex unknown))
  cq (build equiv *y equiv *z))
```

Case frame: forall/ant/cq



Comment for PK6: The purpose of this rule was to allow Cassie to make an inference on something which has the property of being "unknown". It makes this inference by looking for the relationship of "opposite" which the unknown object (z) and another object (y) both have with respect to a third object (x). If these relationships (i.e. x opposite y, and x opposite z) are found, then Cassie infers that y and z are equivalent. This kind of reasoning was exhibited by the human subjects when they inferred the meaning of aphelion must be equivalent to the opposite of closest point.

Another way representing this rule is as a kind of pseudo-syllogism:

x is an opposite of y
x is an opposite of z
_____
z is equivalent to y

It is important to note, that this not a general form of valid reasoning. For example, New York City and Philadelphia are both opposites (in a geographic sense) of Los Angles, but it does not follow from this that New York City is equivalent to Philadelphia. However, the importance

of this rule is not in producing valid reasoning, but in modeling the reasoning performed by the human subjects. That is, that human subjects inferred that the aphelion was the farthest point from point because July is "opposite" to January and the farthest point is "opposite" to closest point.

5. Cassie's Results

In the proceeding section, I explained both the representation of my passage and the pertinent background information in SNePS. However, the question still remains about what Cassie inferred for the meaning of "aphelion point", and how the inference was made.

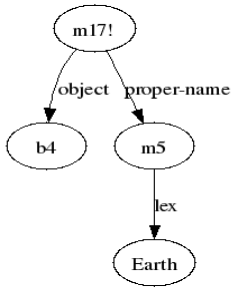
Based on the background rules, what Cassie should have inferred was that "aphelion-point" is equivalent to "farthest-point". This is in fact what Cassie inferred, but what is interesting is the way in the inference was made. For, as the propositions for the passage were read by Cassie, Cassie used the background knowledge was used to make new inferences.

To see this, let us take a look at the output obtained from running the program aphelion-point.demo. The propositions read by Cassie are labeled C1 – C8 ("C" for Cassie), and similar to the previous section, the propositions are italicized for ease of readability and commentary is provided. Before-and-after graphs are used to focus on how the application of prior knowledge produced new inferences. Propositions which do not include graphs are the same as the corresponding graphs in P1 – P8. For the complete text of my program, please see Appendix B, and for a complete text of a demonstration of my program, please see Appendix C.

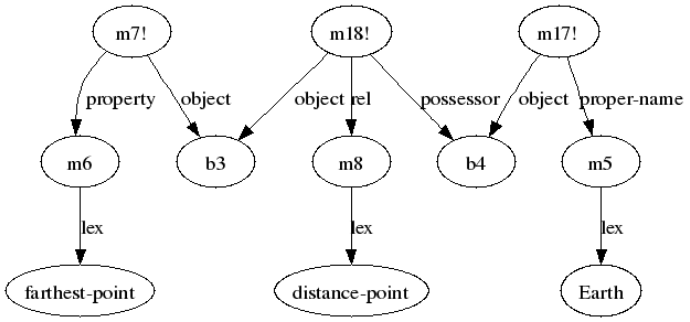
I. FIRST SENTENCE: Earth reaches its closest point in January.

C1. There is something named "Earth".

Before PK3:



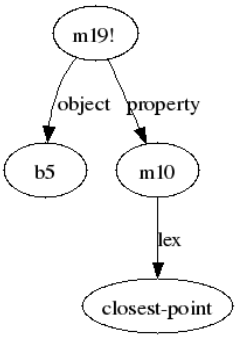
After PK3:



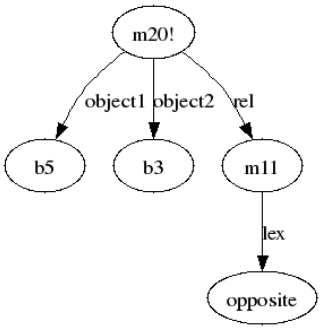
Comment for C1: When given proposition P1, Cassie applied rule PK3 which says that if there is something named "Earth" then it has a farthest-point. This is represented after-graph which shows farthest-point (node b3) related to Earth (node b4) via the relationship "distance-point".

C2. Something has the property "closest point".

Before PK4:



After PK4:



Comment for C2: When given P2, Cassie applied rule PK4 which says that if something is a closest-point and something is an farthest-point then they are opposites of one another. This represented in the after-graph which shows closest-point (node b5) and farthest-point (node b3) are related via "opposite".

C3. Earth has the thing "closest point" via the relationship "distance point".

C4. Earth reaches the closest point in January.

II. SECOND SENTENCE: Earth reaches its aphelion point in July.

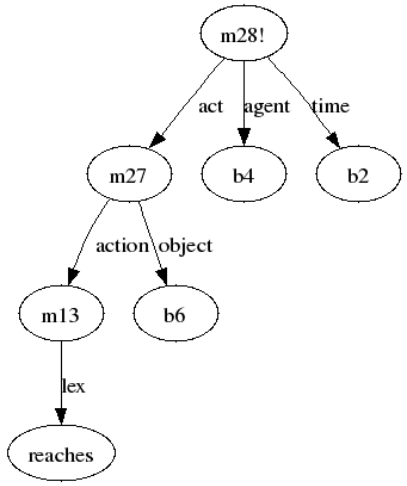
C5. There is something named "Earth".

C6. Something has the property "aphelion point".

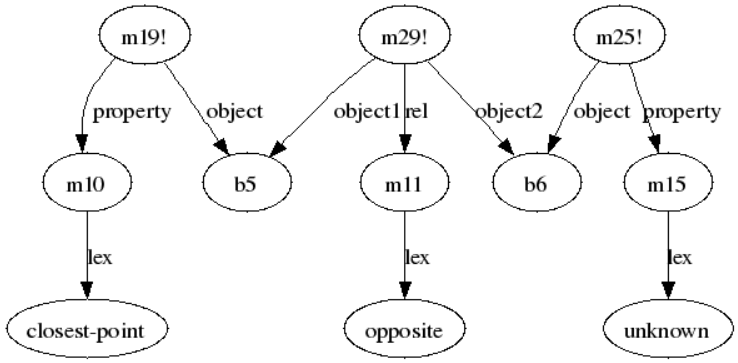
C7. Earth has the thing "aphelion point" via the relationship "distance point".

C8. Earth reaches the "aphelion point" in July.

Before PK5:

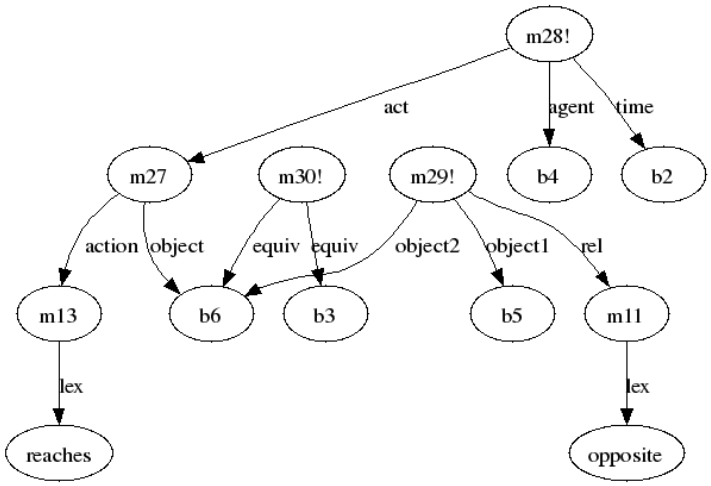


After PK5:

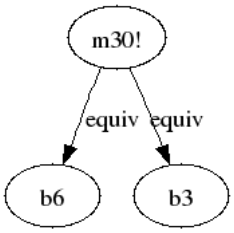


Comment for C8: Unlike the C1 and C2 above, C8 involves multiple inferences (PK5 and PK6). First, Cassie applied PK5 which says that if Earth reaches a point in January and another different point in July then those two points are opposites. This is represented in the after-graph which show closest-point (node b5) and aphelion-point (node b6) related as "opposite".

As a result of this new inference (i.e., the result of PK5) Cassie was then able apply PK6 which says basically says that if objects y and z are opposites of object x, then y and z are equivalent. The resulting graph looks like this:



This graph is a bit complicated to read, but the thing I wish to draw your attention to is node m30! (a close up is below) which represents the proposition that aphelion-point (node b3) and farthest-point (node b6) are equivalent.



This result (i.e., the proposition represented by node m30!) was expected. However, as has been shown in this section, what is most interesting is not the result, but the types of inferences Cassie made in obtaining the result. Like the human subjects, Cassie used background knowledge in order to "guess" (or compute) the meaning of "aphelion point".

6. Future Work

Although Cassie made an inference between farthest-point and aphelion-point, there is still much work (both short term and long term) which needs to be done. In the short term,

Cassie's defineNoun algorithm needs to be modified to handle the equiv-equiv and agent/act/action/object/time case frames I used in this project. For, instead of using the defineNoun algorithm to find the inferred meaning, I had to ask Cassie if two equivalent nodes existed using the "findassert" command: (findassert equiv ?x equiv ?x)

In the long term, a more complex set of case frames needs to be developed for representing many of the concepts in my passage. For example, I made liberal use of the relationships "distance-point", "opposite", and "reaches". These are complex notions in and of themselves, and my representations do not delve into their complexities. Also, my representation of an action occurring at a particular time is too general. An adequate representation of this time relation needs more detail.

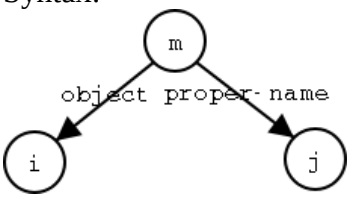
Appendix A: Case Frames

The following is a description of the syntax and semantics of the case frames used for representing the CVA passage.[10] In all case frames, *m* represents a previously unused identifier node, and *i, j, k, m, n,* and *t* are individual nodes used to represent entities. The result is a network (i.e., a graph of nodes and edges) with *m* being a structured proposition node.

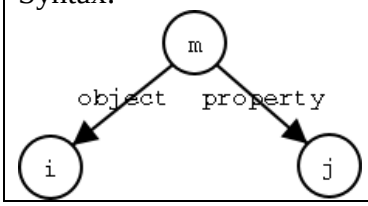
1. lex

Syntax:	Semantics:
	[[m]] is the concept expressed by uttering [[w]] Example: *A concept expressed in English by the word "brachet"* (build lex "brachet")

2. object/proper-name

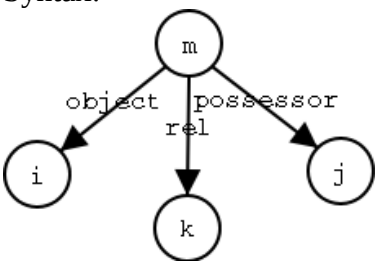
Syntax:	Semantics:
	[[m]] is the proposition that [[i]] is called by the proper name [[j]]. Example: *Something is named "John".* (build object #john proper-name (build lex "John"))

3. object/property

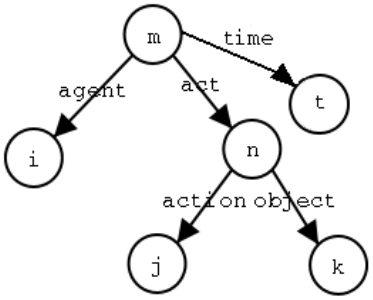
Syntax:	Semantics:
	[[m]] is the proposition that [[i]] has the property [[j]]. Example: *The brachet is white.* (assert object #brachet property (build lex "white"))

[10] http://www.cse.buffalo.edu/~rapaport/CVA/CaseFrames/case-frames/

4. object/rel/possessor

Syntax:	Semantics:
	[[m]] is the proposition that [[i]] is a [[k]] of [[j]]; **or:** [[m]] is the proposition that [[i]] is [[j]]'s [[k]]; **or:** [[m]] is the proposition that [[i]] is possessed (in some unspecified way) by [[j]] and the relationship of [[i]] to [[j]] is [[k]]. Example: *Fido is John's dog.* (assert object #Fido rel (build lex "dog") possessor #john)

5**.** agent/act/action/object/time

Syntax:	Semantics:
	[[m]] is the proposition that agent [[i]] performs action [[j]] with respect to object [[k]], and the action [[j]] occurs during or at time [[t]]. Example: *Now, Sir Tor picks up the brachet.* (assert agent #SirTor act (build action (build lex "pick up") object #brachet) time #now)

Appendix B: My SNePS Program for aphelion-point (aphelion-point.demo)

```
; ========================================================================
; FILENAME: aphelion-point.demo
; DATE:           12/08/2008
; PROGRAMMER:     Bill Duncan

;; 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 "aphelion-point.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)

; Load the appropriate definition algorithm:
^(load "/projects/rapaport/CVA/STN2/defun_noun.cl")

; Clear the SNePS network:
(resetnet t)

; 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:
; NB: If "intext" causes a "nil not of expected type" error,
;     then comment-out the "intext" command and then
;          uncomment & use the load command below, instead
;^(load "/projects/rapaport/CVA/STN2/demos/rels")
(intext "/projects/rapaport/CVA/STN2/demos/rels")
```

```
; load all pre-defined path definitions:
(intext "/projects/rapaport/CVA/mkb3.CVA/paths/paths")

; BACKGROUND KNOWLEDGE:
; =====================

;; The following propositions (PK1 - PK6)
;; represent the prior knowledge (abbreviated "PK")
;; used for determining the meaning of
;; aphelion point (note: PK3 - PK6 are rules
;; for inferring new propositions).
;; The case frame used to represent the propostion is included
;; with each proposition.
;; For more information on these case frames see
;; http://www.cse.buffalo.edu/~rapaport/CVA/CaseFrames/case-frames/
;; http://www.cse.buffalo.edu/sneps/Manuals/manual27.pdf
;; http://www.cse.buffalo.edu/sneps/.

;; ************************************
;; PK1. Something is named January
;; case frame: object/proper-name
;; ************************************
(describe
  (assert object #january
          proper-name (build lex January)))

;; ************************************
;; PK2. Something is named July
;; case frame: object/proper-name
;; ************************************
(describe
  (assert object #july
          proper-name (build lex July)))

;; ****************************************
;; PK3. (Rule):
;; if something has proper name "Earth"
;; then there is something which has the property "farthest point"
;;      and "farthest point" is possessed by "Earth"
;;      via the relationship "distance point"
;; case frame: forall/ant/cq
;; ****************************************
(describe
  (assert forall $x
    &ant(build object *x
                proper-name (build lex Earth))
    cq ((build object #farthest-point
                property (build lex farthest-point))
        (build object *farthest-point
                rel (build lex distance-point)
                possessor *x))))

;; ************************************
;; PK4. (Rule):
;; for all points x and y
```

```
;; if x has property "closest point"
;;     and y has property "farthest point"
;; then point x is the opposite of point y
;; case frame: forall/ant/cq
;; ************************************
(describe
  (assert forall ($x $y)
    &ant(build object *x
               property (build lex closest-point))
    &ant(build object *y
               property (build lex farthest-point))
    cq (build rel (build lex opposite)
                  object1 *x
                  object2 *y)))

;; ************************************
;; PK5. (Rule):
;; for all points x, y, and planets p
;; if p reaches point x in January
;;    and p reaches point y in July
;; then point x is the opposite of point y
;; case frame: forall/ant/cq
;; ************************************
(describe
  (assert
    forall ($planet $x $y)
      &ant(build time *january
                 agent *planet
                 act (build action (build lex reaches)
                                   object *x))
      &ant(build time *july
                 agent *planet
                 act (build action (build lex reaches)
                                   object *y))
      cq (build rel (build lex opposite)
                object1 *x
                object2 *y)))

;; ************************************
;; PK6. (Rule):
;; for all x, y, and z
;; if x is the opposite of y
;;    and x is the opposite of z
;;   and z has the property "unknown"
;; then y is equiv to z
;; case frame: forall/ant/cq
;; ************************************
(describe
  (assert
    forall ($x $y $z)
      &ant(build rel (build lex opposite)
                  object1 *x
                  object2 *y)
      &ant(build rel (build lex opposite)
                  object1 *x
                  object2 *z)
```

```
      &ant(build object *z
                 property (build lex unknown))
      cq (build equiv *y equiv *z)))


; CASSIE READS THE PASSAGE:
; =========================

;; Here is the orignial passage:
;; Earth reaches its closest point in January.
;; Earth reaches its aphelion point in July.

;; The above two sentences are broken
;; down into the following eight propositions:

;; FIRST SENTENCE:
;; Earth reaches its closest point in January.
;; Breakdown:
;; P1. There is something named "Earth".
;; P2. Something has the property "closest point".
;; P3. Earth has (possesses) the thing "closest point"
;;     via the relationship "distance point"
;; P4. Earth reaches  the closest point in January.

;; SECOND SENTENCE:
;; Earth reaches its aphelion point in July.
;; Breakdown:
;; P5. There is something named "Earth".
;; P6. Something has the property "aphelion point".
;; P7. Earth has (possesses) the thing "aphelion point"
;;     via the relationship "distance point"
;; P8. Eath reaches  the aphelion point in July.

;; These propositions are represented in SNePSUL as follows.
;; The case frame used to represent the propostion is included
;; with each proposition.
;; For more information on case frames see
;; http://www.cse.buffalo.edu/~rapaport/CVA/CaseFrames/case-frames/
;; http://www.cse.buffalo.edu/sneps/Manuals/manual27.pdf
;; http://www.cse.buffalo.edu/sneps/.

;; ************************************
;; P1. There is something named "Earth".
;; case frame: object/proper-name
;; ************************************
(describe
  (add object #earth
       proper-name (build lex Earth)))

;; ************************************
;; P2. Something has the property "closest point".
;; case frame: object/property
;; ************************************
(describe
  (add object #closest-point
       property (build lex closest-point)))
```

```
;; ************************************
;; P3. Earth has (possesses) the thing "closest point"
;;     via the relationship "distance point"
;; case frame: object/rel/possessor
;; ************************************
(describe
  (add object *closest-point
       rel (build lex distance-point)
       possessor *earth))

;; ************************************
;; P4. Earth reaches the closest point in January.
;; case frame: agent/act/action/object/time
;; note: This is a non-standard case frame.
;;       The time relation has been added to
;;       the standard agent/act/action/object
;;       case frame in order to denote the time
;;       at which the action occurred.
;; ************************************
(describe
  (add agent *earth
       act (build action (build lex reaches)
                  object *closest-point)
       time *january))

;; ************************************
;; P5. There is something named "Earth".
;; case frame: object/proper-name
;; ************************************
;; This does not need to be represented because it
;; has already been represented in P1 above.

;; ************************************
;; P6. Something has the property "aphelion point".
;; case frame: object/property
;; ************************************
(describe
  (add object #aphelion-point
       property (build lex unknown)))


;; ************************************
;; P7. Earth has (possesses) the thing "aphelion point"
;;     via the relationship "distance point"
;; case frame: object/rel/possessor
;; ************************************
(describe
  (add object *aphelion-point
       rel (build lex distance-point)
       possessor *earth))

;; ************************************
;; P8. Eath reaches  the aphelion point in July.
;; case frame: agent/act/action/object/time
;; note: This is a non-standard case frame.
```

```
;;       The time relation has been added to
;;       the standard agent/act/action/object
;;       case frame in order to denote the time
;;       at which the action occurred.
;; ************************************
(describe
  (add agent *earth
       act (build action (build lex reaches)
                  object *aphelion-point)
       time *july))

; Ask Cassie what "aphelion-point" means:
;^(defineNoun "aphelion-point")
;; The defineNoun algorithm will not recognize equiv-equiv relationships.
;; So, instead find two nodes which stand in equiv-equiv relationhsip.
;;  That is, the nodes are equivelant to each other.
(describe (findassert equiv ?x equiv ?x))
```

Appendix C: Demonstration of My SNePS Program for Aphelion-Point

```
   Welcome to SNePS-2.7 [PL:1 2008/02/12 17:19:45]

Copyright (C) 1984--2007 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.

   12/2/2008 14:04:45

* (demo "aphelion-point.demo")

File /home/phigrad/wdduncan/ubcourses/adv_knowledge_rep/aphelion-point.demo
is now the source of input.


 CPU time : 0.01

* ; ============================================================================
; FILENAME:     aphelion-point.demo
; DATE:         12/08/2008
; PROGRAMMER:   Bill Duncan

;; 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 "aphelion-point.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:
^(
--> load "/projects/rapaport/CVA/STN2/defun_noun.cl")
; Loading /projects/rapaport/CVA/STN2/defun_noun.cl
t
```

```
 CPU time : 0.03

*

; Clear the SNePS network:
(resetnet t)

Net reset


 CPU time : 0.00

*

; 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:
; NB: If "intext" causes a "nil not of expected type" error,
;     then comment-out the "intext" command and then
;          uncomment & use the load command below, instead
;^(load "/projects/rapaport/CVA/STN2/demos/rels")
(intext "/projects/rapaport/CVA/STN2/demos/rels")
Loading file /projects/rapaport/CVA/STN2/demos/rels.


 CPU time : 0.00

*

; 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 : 0.00

*

; BACKGROUND KNOWLEDGE:
; =====================

;; The following propositions (PK1 - PK6)
;; represent the prior knowledge (abbreviated "PK")
;; used for determining the meaning of
;; aphelion point (note: PK3 - PK6 are rules
;; for inferring new propositions).
;; The case frame used to represent the propostion is included
;; with each proposition.
;; For more information on these case frames see
;; http://www.cse.buffalo.edu/~rapaport/CVA/CaseFrames/case-frames/
;; http://www.cse.buffalo.edu/sneps/Manuals/manual27.pdf
;; http://www.cse.buffalo.edu/sneps/.

;; ************************************
;; PK1. Something is named January
;; case frame: object/proper-name
;; ************************************
(describe
  (assert object #january
          proper-name (build lex January)))
(m2! (object b1) (proper-name (m1 (lex January))))

(m2!)

 CPU time : 0.00

*

;; ************************************
;; PK2. Something is named July
;; case frame: object/proper-name
;; ************************************
(describe
  (assert object #july
          proper-name (build lex July)))
```

```
(m4! (object b2) (proper-name (m3 (lex July))))

(m4!)

 CPU time : 0.00

*

;; ****************************************
;; PK3. (Rule):
;; if something has proper name "Earth"
;; then there is something which has the property "farthest point"
;;      and "farthest point" is possessed by "Earth"
;;      via the relationship "distance point"
;; case frame: forall/ant/cq
;; ****************************************
(describe
  (assert forall $x
    &ant(build object *x
               proper-name (build lex Earth))
    cq ((build object #farthest-point
               property (build lex farthest-point))
        (build object *farthest-point
               rel (build lex distance-point)
               possessor *x))))
(m9! (forall v1) (ant (p1 (object v1) (proper-name (m5 (lex Earth)))))
 (cq (p2 (object b3) (possessor v1) (rel (m8 (lex distance-point))))
  (m7 (object b3) (property (m6 (lex farthest-point))))))

(m9!)

 CPU time : 0.00

*

;; ****************************************
;; PK4. (Rule):
;; for all points x and y
;; if x has property "closest point"
;;     and y has property "farthest point"
;; then point x is the opposite of point y
;; case frame: forall/ant/cq
;; ****************************************
(describe
  (assert forall ($x $y)
    &ant(build object *x
               property (build lex closest-point))
    &ant(build object *y
               property (build lex farthest-point))
    cq (build rel (build lex opposite)
                  object1 *x
                  object2 *y)))
(m12! (forall v3 v2)
 (&ant (p4 (object v3) (property (m6 (lex farthest-point))))
  (p3 (object v2) (property (m10 (lex closest-point)))))
 (cq (p5 (object1 v2) (object2 v3) (rel (m11 (lex opposite))))))
```

```
(m12!)

 CPU time : 0.00

*

;; ************************************
;; PK5. (Rule):
;; for all points x, y, and planets p
;; if p reaches point x in January
;;    and p reaches point y in July
;; then point x is the opposite of point y
;; case frame: forall/ant/cq
;; ************************************
(describe
  (assert
    forall ($planet $x $y)
      &ant(build time *january
                 agent *planet
                 act (build action (build lex reaches)
                           object *x))
      &ant(build time *july
                 agent *planet
                 act (build action (build lex reaches)
                           object *y))
      cq (build rel (build lex opposite)
                object1 *x
                object2 *y)))
(m14! (forall v6 v5 v4)
 (&ant
  (p9 (act (p8 (action (m13 (lex reaches))) (object v6))) (agent v4)
   (time b2))
  (p7 (act (p6 (action (m13)) (object v5))) (agent v4) (time b1)))
 (cq (p10 (object1 v5) (object2 v6) (rel (m11 (lex opposite))))))

(m14!)

 CPU time : 0.00

*

;; ************************************
;; PK6. (Rule):
;; for all x, y, and z
;; if x is the opposite of y
;;    and x is the opposite of z
;;   and z has the property "unknown"
;; then y is equiv to z
;; case frame: forall/ant/cq
;; ************************************
(describe
  (assert
    forall ($x $y $z)
      &ant(build rel (build lex opposite)
                  object1 *x
```

```
                object2 *y)
      &ant(build rel (build lex opposite)
                  object1 *x
                  object2 *z)
      &ant(build object *z
                 property (build lex unknown))
      cq (build equiv *y equiv *z)))
(m16! (forall v9 v8 v7)
 (&ant (p13 (object v9) (property (m15 (lex unknown))))
  (p12 (object1 v7) (object2 v9) (rel (m11 (lex opposite))))
  (p11 (object1 v7) (object2 v8) (rel (m11))))
 (cq (p14 (equiv v9 v8))))

(m16!)

 CPU time : 0.00

*


; CASSIE READS THE PASSAGE:
; =========================

;; Here is the orignial passage:
;; Earth reaches its closest point in January.
;; Earth reaches its aphelion point in July.

;; The above two sentences are broken
;; down into the following eight propositions:

;; FIRST SENTENCE:
;; Earth reaches its closest point in January.
;; Breakdown:
;; P1. There is something named "Earth".
;; P2. Something has the property "closest point".
;; P3. Earth has (possesses) the thing "closest point"
;;     via the relationship "distance point"
;; P4. Earth reaches  the closest point in January.

;; SECOND SENTENCE:
;; Earth reaches its aphelion point in July.
;; Breakdown:
;; P5. There is something named "Earth".
;; P6. Something has the property "aphelion point".
;; P7. Earth has (possesses) the thing "aphelion point"
;;     via the relationship "distance point"
;; P8. Eath reaches  the aphelion point in July.

;; These propositions are represented in SNePSUL as follows.
;; The case frame used to represent the propostion is included
;; with each proposition.
;; For more information on case frames see
;; http://www.cse.buffalo.edu/~rapaport/CVA/CaseFrames/case-frames/
;; http://www.cse.buffalo.edu/sneps/Manuals/manual27.pdf
;; http://www.cse.buffalo.edu/sneps/.
```

```
;; ************************************
;; P1. There is something named "Earth".
;; case frame: object/proper-name
;; ************************************
(describe
  (add object #earth
       proper-name (build lex Earth)))
(m18! (object b3) (possessor b4) (rel (m8 (lex distance-point))))
(m17! (object b4) (proper-name (m5 (lex Earth))))
(m7! (object b3) (property (m6 (lex farthest-point))))

(m18! m17! m7!)

 CPU time : 0.00

*

;; ************************************
;; P2. Something has the property "closest point".
;; case frame: object/property
;; ************************************
(describe
  (add object #closest-point
       property (build lex closest-point)))
(m20! (object1 b5) (object2 b3) (rel (m11 (lex opposite))))
(m19! (object b5) (property (m10 (lex closest-point))))

(m20! m19!)

 CPU time : 0.01

*

;; ************************************
;; P3. Earth has (possesses) the thing "closest point"
;;     via the relationship "distance point"
;; case frame: object/rel/possessor
;; ************************************
(describe
  (add object *closest-point
       rel (build lex distance-point)
       possessor *earth))
(m23! (object b5) (possessor b4) (rel (m8 (lex distance-point))))

(m23!)

 CPU time : 0.00

*

;; ************************************
;; P4. Earth reaches the closest point in January.
;; case frame: agent/act/action/object/time
;; note: This is a non-standard case frame.
;;       The time relation has been added to
;;       the standard agent/act/action/object
```

```
;;       case frame in order to indicate the time
;;       at which the action occurred.
;; ************************************
(describe
  (add agent *earth
       act (build action (build lex reaches)
                  object *closest-point)
       time *january))
(m24! (act (m22 (action (m13 (lex reaches))) (object b5))) (agent b4)
 (time b1))

(m24!)

 CPU time : 0.01

*

;; ************************************
;; P5. There is something named "Earth".
;; case frame: object/proper-name
;; ************************************
;; This does not need to be represented because it
;; has already been represented in P1 above.

;; ************************************
;; P6. Something has the property "aphelion point".
;; case frame: object/property
;; ************************************
(describe
  (add object #aphelion-point
       property (build lex unknown)))
(m25! (object b6) (property (m15 (lex unknown))))

(m25!)

 CPU time : 0.00

*


;; ************************************
;; P7. Earth has (possesses) the thing "aphelion point"
;;     via the relationship "distance point"
;; case frame: object/rel/possessor
;; ************************************
(describe
  (add object *aphelion-point
       rel (build lex distance-point)
       possessor *earth))
(m26! (object b6) (possessor b4) (rel (m8 (lex distance-point))))

(m26!)

 CPU time : 0.00

*
```

```
;; ************************************
;; P8. Eath reaches  the aphelion point in July.
;; case frame: agent/act/action/object/time
;; note: This is a non-standard case frame.
;;       The time relation has been added to
;;       the standard agent/act/action/object
;;       case frame in order to indicate the time
;;       at which the action occurred.
;; ************************************
(describe
  (add agent *earth
       act (build action (build lex reaches)
                  object *aphelion-point)
       time *july))
(m30! (equiv b6 b3))
(m29! (object1 b5) (object2 b6) (rel (m11 (lex opposite))))
(m28! (act (m27 (action (m13 (lex reaches))) (object b6))) (agent b4)
 (time b2))

(m30! m29! m28!)

 CPU time : 0.00

*

; Ask Cassie what "aphelion-point" means:
;^(defineNoun "aphelion-point")
;; The defineNoun algorithm will not recognize equiv-equiv relationships.
;; So, instead find two nodes which stand in equiv-equiv relationhsip.
;;  That is, the nodes are equivelant to each other.
(describe (findassert equiv ?x equiv ?x))
(m30! (equiv b6 b3))

(m30!)

 CPU time : 0.00

*

End of /home/phigrad/wdduncan/ubcourses/adv_knowledge_rep/aphelion-point.demo
demonstration.
```

References

1. Rapaport, William J. (2008.01.10) "Computational Contextual Vocabulary Acquisition" [http://www.cse.buffalo.edu/~rapaport/CVA/cva.html].

2. Rapaport, William J. (2002.08.22), "CVA Project Description" [http://www.cse.buffalo.edu/~rapaport/CVA/cvadescription.html].

3. Zhang, Li (2008.08.21), "SNePS Research Group Home Page" [http://www.cse.buffalo.edu/sneps].

4. Mish, Frederick C. [chief editor] (2005), *Meriam-Webster's 11th Collegiate Dictionary* (Springfield, MA: Merriam-Webster, Incorporated).

5. Russell, Randy (2005.12.14), "Perihelion and Aphelion" [http://www.windows.ucar.edu/tour/link=/physical_science/physics/mechanics/orbit/perihelion_aphelion.html&eed=high].

6. Shapiro, Stuart C., et al. (2008.02.11) "SNePS 2.7 User Manual" [http://www.cse.buffalo.edu/sneps/Manuals/manual27.pdf].