Understanding It All:
Defining "Pyrrhic Victory" from Context

Brian Anger
December 12, 2003
CSE663—Advanced Knowledge Representation

Abstract

As part of an ongoing project at the University at Buffalo developing a computational theory and algorithm to perform contextual vocabulary acquisition (CVA), an example passage will be discussed and how it can be defined using the current algorithm. The unknown noun phrase for this case study is "Pyrrhic victory." Included is a short background on CVA and the Semantic Network Processing System (SNePS). The goal of this case study is to represent a complicated passage in an uncomplicated way using SNePS and determine what background information is necessary to provide an adequate definition using the CVA noun definition algorithm. This document presents a simple way to represent the passage along with the background knowledge needed to generate an acceptable definition with the CVA noun definition algorithm. Finally, it will offer some short-term goals and some possible long-term tasks that will surely be part of the future of the CVA project at the University at Buffalo.

1. INTRODUCTION

1.1. Contextual Vocabulary Acquisition

How would you teach a child to read? You would likely teach them to associate letters with their phonetic interpretation, the most commonly used approach to teach someone how to read. If at this point you tell the child that he or she now knows how to read, you would not be telling him or her the truth. Understanding the sounds of words only makes that child a word caller. It is only when they understand the semantics of words that they can truly be called

readers. Surely they could use a dictionary to help understand the meanings of words, but more often than not they will be without one, especially when reading things other than books. It is also very time-consuming to have to look up each and every unknown word. This is why we teach them to try to figure out the meaning of the unknown word from the surrounding words, the unknown word's context. This method is highly ineffective because their guess for the meaning is frequently incorrect. This Contextual Vocabulary Acquisition (CVA) project at the University at Buffalo is trying to increase this efficiency by using an example of the scientific method.

According to Rapaport, "contextual vocabulary acquisition is the active, deliberate acquisition of word meanings from text by reasoning from contextual clues, prior knowledge, language knowledge, and hypotheses developed from prior encounters (if any) with the word, but without external sources of help such as dictionaries or people" (Rapaport 2002: 3). To increase the efficiency of CVA in readers, a computational theory and algorithm must be developed so that a system that understands natural language can provide a definition for an unknown word based on what background knowledge it has for the context of the unknown word. If an algorithm can be developed, it is possible that we could develop a similar (and hopefully simpler) algorithm that may be taught to new readers to help them understand the meanings of unknown words. The computational theory and algorithm to perform CVA is being developed in the SNePS Research Group (SNeRG) at the University at Buffalo.

1.2. The Semantic Network Processing System

The Semantic Network Processing System (SNePS) is capable of representing nearly everything in natural language, making it the perfect accompaniment to a CVA project that desires to represent natural language. The primary data structure is the semantic network, and

SNePS has commands for building these networks and manipulating nodes in them. We can call this network a propositional semantic network because the propositions are represented by nodes rather than by arcs, as they may be in other semantic networks. This allows us to make reference to propositions just as we would single words or proper names. In SNePS, each intensional concept is represented by a unique node and each node represents a unique intensional concept. These concepts can encompass anything that we like, such as the proposition "Mary is a girl."

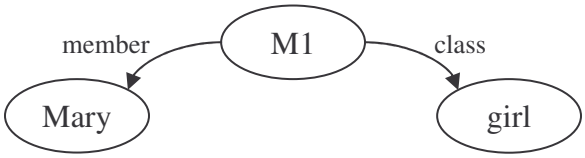

Figure 1: Mary is a girl.

In figure 1, the lower left node represents Mary and the lower right node represents the concept of being a girl. What makes this a propositional semantic network is that the proposition that, or concept of, "Mary is a girl" is represented by node `M1`.

Before we can represent the passage containing the unknown word and even before we can represent the background knowledge, we must decide on what "case frames" we intend to use. Generally, a case frame consists of two or more arcs that are to be used to represent propositions. For instance, in figure 1, the case frame used is called the member-class case frame. The semantics of this case frame implies that `M1` is the proposition that Mary is a member of the class "girl." Other commonly used case frames in SNePS include subclass-superclass, object-property, and agent-act-action-object. The semantics for these case frames can be found online in Scott Napieralski's Dictionary of CVA SNePS Case Frames[1]. SNePS, by design, allows you to define any case frames that you would like. However, the CVA noun definition

[1] http://www.cse.buffalo.edu/~stn2/cva/case-frames/

algorithm is only able to interpret those in Napieralski's dictionary, so only those will be used in the CVA example later (along with a couple others to be discussed).

The main form of interaction with SNePS is through Cassie, its computational cognitive agent. Cassie's memory consists of the knowledge that we enter into SNePS using a user interface language called SNePSUL. Cassie will believe any propositions that we give her using the command `assert`. We can add propositions and trigger forward inference in Cassie using the command `add`. We can also create unasserted nodes using the command `build`. At least one of these three commands will be used in nearly every SNePSUL command made in the CVA example later. A complete manual for SNePS and SNePSUL are available online on the SNePS Research Group website[2].

Node-based reasoning while Cassie "reads" the passage in our CVA example is done by the SNePS inference package (SNIP). It is capable of performing both backward and forward inference, that is, it can try to prove instances of consequents or it can find consequences of antecedents. When a rule is activated in SNIP, it will remain activated unless explicitly deactivated. In the SNePS system, the activated rules are collected into an "active connection graph," which is then used to carry out the inference (Martins 2002: 190). In the CVA example, rules will be created (along with other background information) to assist in the interpretation of the context by Cassie.

2. THE TASK

2.1. The Passage

I have chosen to define "Pyrrhic victory" using the context of the following passage: "In *The Pity of War* (1998), Ferguson argued that British involvement in World War I was

[2] http://www.cse.buffalo.edu/sneps/Bibliography/bibliography.html#Manuals

unnecessary, far too costly in lives and money for any advantage gained, and a Pyrrhic victory that in many ways contributed to the end of the Empire" (Harsanyi 2003: 45). The reason I selected this passage (other than that it is one of the few passages in which "Pyrrhic victory" is used) is that it is a complicated sentence. It contains a list of properties for a given subject, has an indirect relation between the proper name "the Empire" and the property "British," and compares a conceptual property with an object (too costly in lives and money for any advantage gained). William J. Rapaport, CVA project director at the University at Buffalo, suggested that the passage be simplified by changing the comparison and removing unnecessary words. Therefore, the actual passage that I chose to represent was "British involvement in World War I was unnecessary, more costly in lives and money than the advantage gained, and a Pyrrhic victory that contributed to the end of the Empire."

I found it very difficult to parse the passage and provide Cassie with the best representation of it. I focused on the term "involvement" and used it as a base node in representing the passage. There exists some involvement, and the following claims can be made:

- the involvement was British;
- the involvement was in World War I;
- the involvement was unnecessary;
- the involvement was costly in lives and money;
- there was an advantage gained from the involvement;
- the involvement was more costly in lives and money than the advantage gained;
- the involvement was a Pyrrhic victory;
- and, the involvement contributed to the end of the Empire.

The eight claims above are the propositions that I chose to represent while Cassie is "reading" the passage. However, for Cassie to truly understand the sentence, we must give her background knowledge before asking her to "read." I believe that this background knowledge can be separated into two distinct categories, one containing all the rules necessary to interpret a sentence and one containing the other propositions (mostly about objects and relations). One such background rule is that if an empire is British, it is ruled by Great Britain. The other propositions may include things such as there exists some involvement, the Empire is British, and World War I is a war. Without background knowledge, the CVA noun definition algorithm would provide a very limited output.

2.2. Project Goals

From the very beginning my goals for this project have been simple: represent a complicated passage in an uncomplicated way using SNePS and determine what background information is necessary to provide an adequate definition using the CVA noun definition algorithm. Providing background rules and developing a CVA noun definition algorithm to work in tandem with them is and will always be the easiest (but not easy) part of getting an intelligent agent such as Cassie to read and understand words that are not already defined. This is why I chose my first goal: providing an uncomplicated representation of a complicated passage. It is very difficult to parse natural language and generate a complicated SNePS representation. To remain uncomplicated, I limited myself to the following four case frames for representing the passage:

- member-class, representing class instances,
- object-property, representing a property of some object,
- object1-rel-object2, representing a relation between two objects,

- and agent-act-action-object, representing action between an agent and an object.

By minimizing the number of case frames that may be used, we can simplify the natural language parser that will need to automatically generate representations. While a natural language parser may never generate correct representations, especially anytime in the near future, any work that can be done now to simplify such a parser is worthwhile. My second goal for this project was determining what would be necessary if a simplified natural language parser existed. After representing the passage as simply as I could the first time, the problem of determining the necessary background knowledge to provide an adequate definition from the simple representation presented itself. The approach I used to accomplish this goal was to provide more than enough background information to Cassie to get an adequate definition from the CVA noun definition algorithm, then remove unnecessary information and "tweak" the remaining information to improve the definition. However, I chose to leave some unnecessary information in, as you will see later, so that if the CVA noun definition algorithm or more background rules are provided, a better definition could be provided. The goals of my project were fairly limited; however the work done in this project can easily be extended to provide more information regarding contextual vocabulary acquisition.

3. DEFINING "PYRRHIC VICTORY"

3.1. Human Protocols

In preparation for defining the project with the CVA noun definition algorithm, I took the passage to several people who had not encountered the phrase "Pyrrhic victory." They were asked to read the simplified passage, define "Pyrrhic victory," and express their reasons for giving the definition they did. All of the people asked defined Pyrrhic victory in an acceptable way, most commonly as an "unnecessary victory," saying that a Pyrrhic victory could not be a

good thing. All of the subjects used the word "unnecessary" as the start of their reasoning, but further developed their definition by looking more at the surrounding context. One person partially defined it as a display of power. Another person defined Pyrrhic victory as "a useless, short-lasting victory that is not a good idea." His reasoning came from "the word 'unnecessary,' the suggested ratio of lives lost to [advantage gained], and the ending 'the end of the Empire.'" The three key reasons he used would become the base of how I would try to express the meaning of "Pyrrhic victory."

While I could continue and hope that you have been able to define "Pyrrhic victory" from context, I doubt that you will have produced the exact meaning of the noun phrase. It is here that I will offer to you the actual definition of "Pyrrhic victory." According to the American Heritage Dictionary, a Pyrrhic victory is "a victory that is offset by staggering losses." Dictionary.com goes on to further explain the origins of the phrase: "A Pyrrhic victory is so called after the Greek king Pyrrhus, who, after suffering heavy losses in defeating the Romans in 279 B.C., said to those sent to congratulate him, 'Another such victory over the Romans and we are undone'" (Dictionary.com 2003).

3.2. Representing Background Knowledge in SNePS

Without any background knowledge, the CVA noun definition algorithm would produce very vague definitions of the words we ask of it. It would only be able to define the unknown word using the words that have a direct relation to it in the given passage. We choose to give Cassie background knowledge so the definition would be closer to one that a human may give. As humans, we have spent every day of our lives accumulating background knowledge, whether it is the court case that ruled states did not have to enforce the return of fugitive slaves (Prigg v.

Pennsylvania), who won the 2003 Stanley Cup (the New Jersey Devils), or the meaning of a Pyrrhic victory.

Since we would like Cassie to use this background knowledge as she reads the passage, we must give it to her before we give her the passage. Through mostly trial and error, I produced a set of propositions that would act as the base for both the representation of the background rules and of the passage:

- there are objects called Great Britain, The Empire, and World War I;
- there exists some advantage and some involvement;
- Great Britain is a political unit;
- The Empire is British;
- The Empire is an empire;
- there exists some end of the Empire;
- World War I is a war;
- and, the expression "Pyrrhic victory" consists of the adjective "Pyrrhic" and the noun "victory."

All of these propositions would be used either by the CVA noun definition algorithm or by the background rules SNIP uses for forward inference when Cassie is "reading" the passage. While most of these propositions are simply common to us (when would the Empire not be an empire and World War I not be a war?), Cassie is unfamiliar with the terms and must be taught what they mean just as we were once taught.

I will spare you the tediousness that would come if I explained the necessary SNePSUL commands to teach Cassie these things, but I must explain a few of the interesting problems I ran into representing this background information. First of all, "the Empire" may not always be

British (i.e. Roman Empire) and its ruling political unit depends heavily on the other terms in the passage including the timeframe of the passage. I could not find a rule capable of associating "the Empire" with the British Empire for this passage, so I chose to assert "the Empire" to always be British. The second strange proposition is that there exists some end of the Empire. I chose to define this end as a member of the class "end of an empire." However, the end of an empire is a virtual concept, rather than a physical one, which is how I am choosing to represent it so that it may be the object in the agent-act-action-object case frame. It seemed far too difficult to represent it such that an agent could perform an act on a virtual concept and to get the CVA noun definition algorithm to recognize the action and provide information about it. For convenience' sake, I represented it as an object.

The last proposition in my background knowledge is that the expression "Pyrrhic victory" consists of the adjective "Pyrrhic" and the noun "victory." Because I wanted to define the noun phrase "Pyrrhic victory" as a noun, I had to keep it together when Cassie is "reading" the passage. However, with help from Rapaport, I was able to build a new case frame that would be used to represent the structure of expressions and could be used in association with a rule to provide the necessary information that a Pyrrhic victory is a victory, without having to define "Pyrrhic victory." I used a newly defined case frame expr-str-adj-noun to tell Cassie that the expression "Pyrrhic victory" has structure consisting of an adjective "Pyrrhic" and a noun "victory." This case frame can be drawn like this:

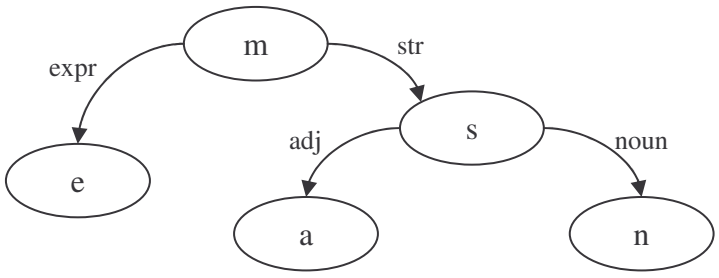

Figure 2: expr-str-adj-noun case frame.

This case frame has the following semantics: [[m]] is the proposition that the expression [[e]] has structure [[s]] consisting of an adjective [[a]] and a noun [[n]]. I have a background rule that will build a subclass-superclass proposition based on the information in this case frame.

3.3. Representing Background Rules in SNePS

Forward inference in SNIP can only provide Cassie with extra knowledge if we build in the right background rules to help interpret the passage. Through five rules, I was able to give Cassie extra knowledge that helped her define "Pyrrhic victory." While I created many more than five rules, only five of the rules helped Cassie in the CVA noun definition algorithm. I will discuss two rules that did not help her later in this section.

The five rules which did help Cassie are:

- an adjective-noun phrase where the adjective is not abnormal is a subclass of the noun's superclass (ADJ-NOUN);
- if an empire is British, then it is ruled by Great Britain (BRITISH-EMPIRE);
- if an involvement is British, then it is Great Britain's involvement (BRITISH-INVOLVEMENT);
- if something contributes to the end of an empire, then that something weakens the political unit that rules the empire (END-OF-EMPIRE);

- and, if X is a member of the class Y and Y is a subclass of the superclass Z, then X is a member of the class Z (MEMBER-SUPERCLASS).

The first of these rules, ADJ-NOUN, produces the proposition that a Pyrrhic victory is a victory using the case frame I discussed at the end of the previous section. If you look at the SNePSUL code for this in the Pyhrric.demo code appendix, you will notice that I ignored the abnormality part of this rule. This, again, was done to simplify the inference of the desired proposition. The desired consequent of this rule is to have "Pyrrhic victory" be a subclass of the superclass "victory."

The second and third rules, BRITISH-EMPIRE and BRITISH-INVOLVEMENT, were used to simply make an attachment between Great Britain and its empire or its involvement, respectively. They both are built upon the fact that each object had or would have the "British" property. The BRITISH-EMPIRE rule has the desired consequent of "Great Britain" (object1) "rules" (rel) "the Empire" (object2). The BRITISH-INVOLVEMENT rule has the desired consequent "Great Britain" is the possessor of the "involvement."

The fourth rule, END-OF-EMPIRE, provides Cassie with a direct correlation between the object that contributes to the end of the empire and the political unit that rules that empire. This will allow the CVA noun definition to speak about how the contributing object (in this case, something that is a Pyrrhic victory) has affected the ruling political unit. The desired consequent of this rule is that the "involvement" (object1) "weakens" (rel) "Great Britain" (object2). The fifth and final used rule is simply taking the common path definition from member to superclass and allowing SNIP to assert the consequent. The desired consequent here is that the "involvement" is a member of the class "victory." I made this a rule so that SNIP would actually add this assertion to the network, giving me better visual of the completed network.

There were two rules that I developed that did not exactly work out as planned. These rules would relate the possessor of an involvement in a war to the war itself. For example, in the given passage, the rules `HEAVY-LOSSES` and `WAR-VICTORY` would allow SNIP to infer that Great Britain incurs heavy losses in World War I and that Great Britain is victorious in World War I, respectively. However, even with SNIP providing the desired consequents, the CVA noun definition algorithm would not look far enough away from the instance of a Pyrrhic victory it was given (the involvement) to draw any conclusions about the how the possessor of the involvement related to the war in which they were involved. I felt that these rules and consequents would be useful in defining "Pyrrhic victory," but I did not have the time to get them incorporated into the definition.

3.4. Representing the Passage in SNePS

As previously mentioned, I chose to represent eight propositions to simulate Cassie's "reading" of the passage. All eight of the propositions include the "involvement" as I tried to keep the subject of the passage as the node base for the entire representation. As I tried to maintain a simple representation of the passage, all but one of the propositions used very simple case frames—object-property, object1-rel-object2, and member-class—and only one of these seven has any complexity to it. The eighth proposition required the agent-act-action-object case frame, which is only slightly more complex than the other three case frames used.

The first proposition, the involvement was British, was represented with the standard object-property case frame—the "involvement" has the property "British." The second proposition represented, the involvement was in World War I, used the object1-rel-object2 case frame to show that the "involvement" (object1) was "in" (rel) "World War I" (object2). Both the third and fourth propositions used the object-property case frame. In the case that the

involvement was unnecessary, the "involvement" has the property "unnecessary;" in the case that the involvement was costly in lives and money, the "involvement" has the property "costly in lives and money." The fifth proposition, the advantage was gained from the involvement, was represented using object1-rel-object2, where the "advantage" (object1) was "gained from" (rel) the "involvement" (object2). Again, I selected all these case frames because I thought they were the simplest to work with and felt that they would be the simplest for a natural language parser to create.

The sixth proposition was by far the most difficult to represent. While I could represent "more costly in lives and money than the advantaged gained" using a single proposition, there was no standard way I could then declare that proposition to be a property of the object "involvement." I tried many ways to break apart the "more costly … advantage gained" section of the passage, but each time I did I was left with no standard way to get the "involvement" into a proposition with it. Through Rapaport's guidance, I chose to represent this troubled proposition using object1-rel-object2, so that I could maintain the simplicity of the passage representation. However, to use this case frame, I had to use a mod-head proposition for the rel-concept node, just so I could separate the "more than" comparison from the "costly in lives and money" portion of the passage. The final representation for this proposition is shown below:

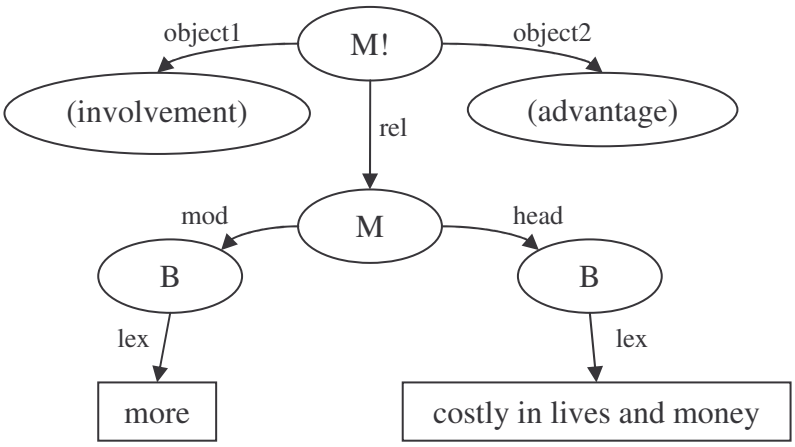

Figure 3: The involvement was more costly in lives and money than the advantage gained.

The proposition that the involvement was a Pyrrhic victory is the key to this representation working with the CVA noun definition algorithm. I represented this proposition using the member-class case frame such that the "involvement" is a member of the class "Pyrrhic victory." The CVA noun definition algorithm recognizes the "involvement" as an instance of a Pyrrhic victory and the properties and actions of the "involvement" become possible properties and possible actions of a Pyrrhic victory. Since I knew the algorithm would do this, my goal was to represent properties and actions of the involvement such that "Pyrrhic victory" could be defined.

The final proposition needed to represent the passage was that the involvement contributed to the end of the Empire. This was represented using the agent-act-action-object case frame, where the agent is the "involvement," the action is "contributes to," and the object is "the end of the Empire" (a base node I created in the background knowledge). Therefore, it technically says that the "involvement" "contributes to" "the end of the Empire." Rather than duplicate the code found in the Pyrrhic.demo code appendix, I have added a graphical representation of these eight propositions below:

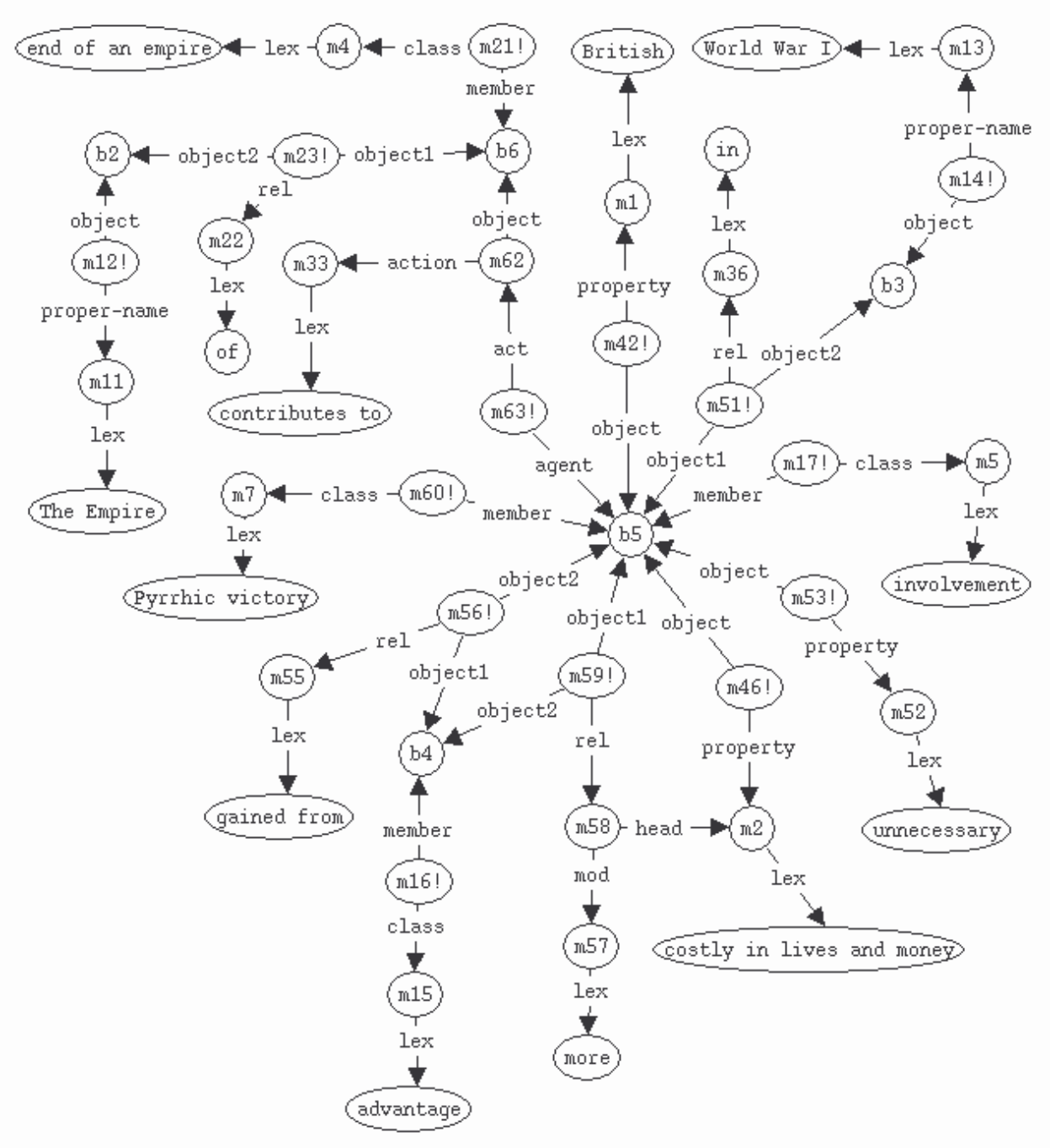

Figure 4: SNePS representation for “Pyrrhic victory” passage.

I have repeated the list of the eight propositions below, along with the labels of their nodes so that you may better interpret the graphical representation above:

- the involvement was British (m42);
- the involvement was in World War I (m51);

- the involvement was unnecessary (m53);
- the involvement was costly in lives and money (m46);
- there was an advantage gained from the involvement (m56);
- the involvement was more costly in lives and money than the advantage gained (m59);
- the involvement was a Pyrrhic victory (m60);
- and, the involvement contributed to the end of the Empire (m63).

I have also expanded upon each base node b2 through b6 so that you may understand what each of them is. The base node b2 represents the Empire; b3 represents World War I; b4 represents an advantage; b5 represents an involvement; and, b6 represents the end of the Empire. The SNePS output from running the code from the Pyrrhic.demo code appendix is also available in the Pyrrhic.demo output appendix, and all of the nodes labels are given there after they are created, so you may use that to help interpret figure 4, if necessary.

3.5. Cassie's Definition

Now that Cassie has learned all of the necessary background knowledge and the passage itself, we can ask her for a definition of "Pyrrhic victory":

```
Definition of Pyrrhic victory:
 Class Inclusions: victory,
 Possible Actions: weakens political unit, contributes to
end of an empire,
 Possible Properties: unnecessary, costly in lives and
money, British, in war,
 Possessive: political unit involvement,
```

We can see that Cassie believes "Pyrrhic victory" to be a victory that is some political unit's involvement and could possibly weaken the political unit or contribute to the end of an empire, which is a very acceptable definition for "Pyrrhic victory" (a victory that is offset by staggering losses).

I found one item of the "possible properties" list of her definition to be rather interesting: British. I completely understand how she came up with this (the involvement was British); however, I am interested in how we could represent the passage such that she does not state this as a possible property. As a human reader, when we read something like "the British involvement was a Pyrrhic victory," there is no way that we would consider a possible property of a Pyrrhic victory to be British. If all Pyrrhic victories were British, then we would have never included British in the sentence in the first place; the sentence would simply be "the involvement was a Pyrrhic victory" because the Pyrrhic victory would lead us to infer that the involvement was British. I believe that we need to find a way to distinguish between the passage being represented and the background knowledge or inferred knowledge that is represented. As explained with British above, the properties of all Pyrrhic victories would not be listed alongside Pyrrhic victory as properties of some victory; therefore, we must not allow Cassie to believe that they are properties of Pyrrhic victory.

4. FUTURE WORK

4.1. Short-Term

There are a few odds and ends that need to be cleaned up in the background knowledge and passage representation that I have developed. Regarding background knowledge, the representation of the "end of the Empire" needs to be improved. I'm fairly uncomfortable with saying there is an instance of an end of an empire and then saying that instance is of "the Empire." One possible improvement to this representation would include the use of a skolem function to represent the end of an empire. Clearly much more work needs to be done regarding the expression structure for which the expr-str-adj-noun case frame was developed. A more general case frame needs to be developed to contain the structure.

The other short term improvement that I would like to see made is changing the two background rules mentioned earlier that do not assist in the definition. They were used to relate a political unit directly to the war it is involved in, though there may need to be some reference to the instance of a Pyrrhic victory for Cassie to be concerned with its consequents.

4.2. Long-Term

I would like to see two major things accomplished in the long-term that have stemmed out of my work in the CVA project. The first task, which I hope is not as difficult as the second one will be, is to find a better way to represent things that are the context of the unknown word, that is, those things in the passage being read. There needs to be a way to distinguish between background knowledge and the passage if we want to remove British from the possible list of properties for a Pyrrhic victory. This will require both an adjustment in how we choose to represent things in the passage and in how the algorithm defines the word using its context.

The longer task is the development of a natural language parser that can create simple graphical representations of passages using the four case frames I used and possibly a few more. If this could be built, the task of determining necessary background information would be easy and an algorithm could be developed to do just that. Unfortunately, there needs to be a balance between the background information and the parser simplicity because, in a perfect world, the parser would have created the background information. Therefore, the parser-generated representation of the passage can be no simpler and no more complex than the background information used. This development will require a lot of work and it is the future of the CVA project.

References Cited

Dictionary.com (2003), “Dictionary.com/Word of the Day: Pyrrhic victory”, 2003 July 16: [http://dictionary.reference.com/wordoftheday/archive/2003/07/16.html].

Martins, João P. (2002), Knowledge Representation.

Harsanyi, David (2003), “The Old Order”, National Review 2003 May 5: 45-46.

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

APPENDIX A: PYRRHIC.DEMO CODE

Please note that any comments in the code are colored in indigo for your convenience.

```
;;; Brian Anger (bdanger@cse.buffalo.edu)
;;; Project: Contextual Vocabulary Acquisition
;;; File: Pyrrhic.demo
;;; Revision: 1.0, 12-Dec-03
;;;
;;; Actual passage:
;;; "In _The Pity of War_ (1998), Ferguson argued that British involvement in
;;; World War I was unnecessary, far too costly in lives and money for any
;;; advantage gained, and a Pyrrhic victory that in many ways contributed to
;;; the end of the Empire."
;;;
;;; Simplified passage:
;;; "British involvement in World War I was unnecessary, more costly in lives
;;; and money than the advantage gained, and a Pyrrhic victory that contributed
;;; to the end of the Empire."
;;;

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

;; Reset the network.
(resetnet t)

;; Turn off infer trace.
^(setq snip:*infertrace* nil)

;; Turn off singular path inference.  Redefine the function to return nil so
;; that forward inference will not be limited.
^(in-package snip)
^(defun broadcast-one-report (rep)
   (let (anysent)
     (do.chset (ch *OUTGOING-CHANNELS* anysent)
               (when (isopen.ch ch)
                 (setq anysent (or (try-to-send-report rep ch) anysent)))))
   nil)
^(in-package sneps)

;; Load pre-defined relations.
(intext "/projects/rapaport/CVA/STN2/demos/rels")

;; Load pre-defined path definitions.
(intext "/projects/rapaport/CVA/mkb3.CVA/paths/paths")

;; Define relations for the structure of various expressions.
(define mod head)
(define expr str adj noun)

;; Build frequently-used lex nodes.
(describe (build lex "British")                    = lexBritish)
(describe (build lex "costly in lives and money") = lexCostlyLM)
(describe (build lex "empire")                     = lexEmpire)
(describe (build lex "end of an empire")           = lexEndEmpire)
(describe (build lex "involvement")                = lexInvolvement)
(describe (build lex "political unit")             = lexPolUnit)
(describe (build lex "Pyrrhic victory")            = lexPV)
(describe (build lex "victory")                    = lexVictory)
```

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

;; Introduce various objects.
(describe (assert object #bBritain   proper-name (build lex "Great Britain")))
(describe (assert object #bTheEmpire proper-name (build lex "The Empire")))
(describe (assert object #bWW1       proper-name (build lex "World War I")))

;; There exists some advantage.
(describe (assert member #bAdv class (build lex "advantage")))

;; There exists some involvement.
(describe (assert member #bInv class *lexInvolvement))

;; "Great Britain" is a political unit.
(describe (assert member *bBritain class *lexPolUnit))

;; "The Empire" is British.
(describe (assert object *bTheEmpire property *lexBritish))

;; "The Empire" is an empire.
(describe (assert object *bTheEmpire class *lexEmpire))

;; There exists some end of the Empire.
(describe (assert member #bEnd class *lexEndEmpire))
(describe (assert object1 *bEnd rel (build lex "of") object2 *bTheEmpire))

;; "World War I" is a war.
(describe (assert member *bWW1 class (build lex "war")))

;; The expression "Pyrrhic victory" consists of an adjective and a noun.
(describe (assert expr *lexPV str (build adj  (build lex "Pyrrhic")
                                         noun *lexVictory)))

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

;; RULE: ADJ-NOUN
;; An ADJ-NOUN where ADJ is not abnormal is a NOUN.
(describe (assert forall ($vANExpr $vAdj $vNoun)
            ant (build expr *vANExpr str (build adj *vAdj noun *vNoun))
            cq  (build subclass *vANExpr superclass *vNoun)))

;; Desired consequent of ADJ-NOUN rule:
;; (describe (assert subclass *lexPV superclass *lexVictory))

;; RULE: BRITISH-EMPIRE
;; If an empire is British, then it is ruled by Great Britain.
(describe (assert forall ($vBritEmp)
            &ant ((build object *vBritEmp
                         property *lexBritish)
                  (build object *vBritEmp
                         class *lexEmpire))
            cq   (build object1 *bBritain
                        rel (build lex "rules")
                        object2 *vBritEmp)))

;; Desired consequent of BRITISH-EMPIRE rule:
;; (describe (assert object1 *bBritain rel (build lex "rules")
;;                   object2 *bTheEmpire))
```

```
;; RULE: BRITISH-INVOLVEMENT
;; If X is an involvment and X is British, then it is of Great Britain.
(describe (assert forall ($vBritInv)
            &ant ((build member *vBritInv
                         class *lexInvolvement)
                  (build object *vBritInv
                         property *lexBritish))
            cq   (build object *vBritInv
                        rel *lexInvolvement
                        possessor *bBritain)))

;; Desired consequent of BRITISH-INVOLVEMENT rule:
;; (describe (assert object *bInv possessor *bBritain))

;; RULE: END-OF-EMPIRE
;; If X contributes to the end of the empire E and the political unit P rules
;; the empire E, then X weakens P.
(describe (assert forall ($vContributer $vAnEnd $vPolUnit $vAnEmpire)
            &ant ((build agent *vContributer
                         act (build action (build lex "contributes to")
                                    object *vAnEnd))
                  (build member *vAnEnd
                         class *lexEndEmpire)
                  (build object1 *vAnEnd
                         rel (build lex "of")
                         object2 *vAnEmpire)
                  (build object1 *vPolUnit
                         rel (build lex "rules")
                         object2 *vAnEmpire))
            cq   (build agent *vContributer
                        act (build action (build lex "weakens")
                                   object *vPolUnit))))

;; Desired consequent of END-OF-EMPIRE rule:
;; (describe (assert object1 *bInv rel (build lex "weakens")
;;                   object2 *bBritain))

;; RULE: HEAVY-LOSSES
;; If X is an involvement, W is a war, X is in W, X is costly in lives and
;; money, and X is possessed by Y, then Y incurs heavy losses in W.
(describe (assert forall ($vAnInv $vWar $vPossessor)
            &ant ((build member *vAnInv
                         class *lexInvolvement)
                  (build member *vWar
                         class (build lex "war"))
                  (build object1 *vAnInv
                         rel (build lex "in")
                         object2 *vWar)
                  (build object *vAnInv
                         property *lexCostlyLM)
                  (build object *vAnInv
                         possessor *vPossessor))
            cq   (build object1 *vPossessor
                        rel (build lex "incurs heavy losses in")
                        object2 *vWar)))

;; Desired consequent of HEAVY-LOSSES rule:
;; (describe (assert object1 *bBritain rel (build lex "incurs heavy losses in")
;;                   object2 *bWW1))
```

```
;; RULE: WAR-VICTORY
;; If X is an involvement, W is a war, X is in W, X is a victory, and X is
;; possessed by Y, then Y is victorious in W.
(describe (assert forall ($vAnInv $vWar $vPossessor)
            &ant ((build member *vAnInv
                         class *lexInvolvement)
                  (build member *vWar
                         class (build lex "war"))
                  (build object1 *vAnInv
                         rel (build lex "in")
                         object2 *vWar)
                  (build member *vAnInv
                         class *lexVictory)
                  (build object *vAnInv
                         possessor *vPossessor))
            cq   (build object1 *vPossessor
                        rel (build lex "is victorious in")
                        object2 *vWar)))

;; Desired consequent of WAR-VICTORY rule:
;; (describe (assert object1 *bBritain rel (build lex "is victorious in")
;;                   object2 *bWW1))

;; RULE: MEMBER-SUPERCLASS
;; If X is a member of the class Y, and Y is a subclass of the superclass Z,
;; then X is a member of the class Z.
(describe (assert forall ($vMember $vSubClass $vSuperClass)
            &ant ((build member *vMember
                         class *vSubClass)
                  (build subclass *vSubClass
                         superclass *vSuperClass))
            cq   (build member *vMember
                        class *vSuperClass)))

;; Desired consequent of MEMBER-SUPERCLASS rule:
;; (describe (assert member *bInv class *lexVictory))

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

;; The involvement was British.
(describe (add object *bInv property *lexBritish))

;; The involvement was in World War I.
(describe (add object1 *bInv rel (build lex "in") object2 *bWW1))

;; The involvement was unnecessary.
(describe (add object *bInv property (build lex "unnecessary")))

;; The involvement was costly in lives and money.
(describe (add object *bInv property *lexCostlyLM))

;; The advantage was gained from the involvement.
(describe (add object1 *bAdv
            rel (build lex "gained from")
            object2 *bInv))

;; The involvement was more costly ... than the advantage (gained from it).
(describe (add object1 *bInv
            rel (build mod (build lex "more") head *lexCostlyLM)
            object2 *bAdv))
```

```
;; The involvement was a Pyrrhic victory...
(describe (add member *bInv class *lexPV))

;; ... that contributed to the end of the Empire.
(describe (add agent *bInv
               act (build action (build lex "contributes to")
                          object *bEnd)))

;;=============================================================================
;; CASSIE'S DEFINITION
;;=============================================================================

^(defineNoun "Pyrrhic victory")
```

APPENDIX B: PYRRHIC.DEMO OUTPUT

Please note that in this appendix any input that is coming from Pyrrhic.demo is colored indigo. Also, after several SNePS outputs, I have added commentary enclosed in square brackets and colored brown like this [comment]. I have removed CPU times to shorten this appendix.

```
* (demo "~/cse663/Pyrrhic.demo")

File /home/csgrad/bdanger/cse663/Pyrrhic.demo is now the source of input.


* ;;; -*- lisp -*-
;;; Brian Anger (bdanger@cse.buffalo.edu)
;;; Project: Contextual Vocabulary Acquisition
;;; File: Pyrrhic.demo
;;; Revision: 1.0, 12-Dec-03
;;;
;;; Actual passage:
;;; "In _The Pity of War_ (1998), Ferguson argued that British involvement in
;;; World War I was unnecessary, far too costly in lives and money for any
;;; advantage gained, and a Pyrrhic victory that in many ways contributed to
;;; the end of the Empire."
;;;
;;; Simplified passage:
;;; "British involvement in World War I was unnecessary, more costly in lives
;;; and money than the advantage gained, and a Pyrrhic victory that contributed
;;; to the end of the Empire."
;;;

;; Load the noun definition algorithm.
^(
--> load "/projects/rapaport/CVA/STN2/defun_noun.cl")
; Loading /projects/rapaport/CVA/STN2/defun_noun.cl
t

[The CVA noun definition has now been loaded into Lisp.]

*
;; Reset the network.
(resetnet t)

Net reset


*
;; Turn off inference tracing.
^(
--> setq snip:*infertrace* nil)
nil

[I have disabled inference tracing to shorten the length of this demo.]

*
;; Turn off singular path inference.  Redefine the function to return nil so
;; that forward inference will not be limited.
^(
--> in-package snip)
#<The snip package>
```

```
* ^(
--> defun broadcast-one-report (rep)
   (let (anysent)
     (do.chset (ch *OUTGOING-CHANNELS* anysent)
               (when (isopen.ch ch)
                 (setq anysent (or (try-to-send-report rep ch) anysent)))))
   nil)
broadcast-one-report

[broadcast-one-report has been redefined, so forward inference is no longer limited.]

* ^(
--> in-package sneps)
#<The sneps package>



*
;; Load pre-defined relations.
(intext "/projects/rapaport/CVA/STN2/demos/rels")
File /projects/rapaport/CVA/STN2/demos/rels is now the source of input.

[SNePS is now loading the pre-defined relations for the CVA project.]

*

(a1 a2 a3 a4 after agent against antonym associated before cause class
 direction equiv etime event from in indobj instr into lex location manner
 member mode object on onto part place possessor proper-name property rel skf
 sp-rel stime subclass superclass subset superset synonym time to whole kn_cat)

*

End of file /projects/rapaport/CVA/STN2/demos/rels


*
;; Load pre-defined path definitions.
(intext "/projects/rapaport/CVA/mkb3.CVA/paths/paths")
File /projects/rapaport/CVA/mkb3.CVA/paths/paths is now the source of input.


*
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-)


*

End of file /projects/rapaport/CVA/mkb3.CVA/paths/paths


*
;; Define relations for the structure of various expressions.
(define mod head)

(mod head)

* (define expr str adj noun)

(expr str adj noun)

*
;; Build frequently-used lex nodes.
(describe (build lex "British")                    = lexBritish)


* (describe (build lex "costly in lives and money") = lexCostlyLM)


* (describe (build lex "empire")                    = lexEmpire)


* (describe (build lex "end of an empire")          = lexEndEmpire)


* (describe (build lex "involvement")               = lexInvolvement)


* (describe (build lex "political unit")            = lexPolUnit)


* (describe (build lex "Pyrrhic victory")           = lexPV)


* (describe (build lex "victory")                   = lexVictory)


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

;; Introduce various objects.
(describe (assert object #bBritain   proper-name (build lex "Great Britain")))

(m10! (object b1) (proper-name (m9 (lex Great Britain))))

(m10!)

[SNePS has assigned Great Britain to base node b1.]
```

```
* (describe (assert object #bTheEmpire proper-name (build lex "The Empire")))

(m12! (object b2) (proper-name (m11 (lex The Empire))))

(m12!)

[SNePS has assigned the Empire to base node b2.]

* (describe (assert object #bWW1       proper-name (build lex "World War I")))

(m14! (object b3) (proper-name (m13 (lex World War I))))

(m14!)

[SNePS has assigned World War I to base node b3.]

*
;; There exists some advantage.
(describe (assert member #bAdv class (build lex "advantage")))

(m16! (class (m15 (lex advantage))) (member b4))

(m16!)

[SNePS has assigned an instance of an advantage to base node b4.]

*
;; There exists some involvement.
(describe (assert member #bInv class *lexInvolvement))

(m17! (class (m5 (lex involvement))) (member b5))

(m17!)

[SNePS has assigned an instance of an involvement to base node b5.]

*
;; "Great Britain" is a political unit.
(describe (assert member *bBritain class *lexPolUnit))

(m18! (class (m6 (lex political unit))) (member b1))

(m18!)

*
;; "The Empire" is British.
(describe (assert object *bTheEmpire property *lexBritish))

(m19! (object b2) (property (m1 (lex British))))

(m19!)

*
;; "The Empire" is an empire.
(describe (assert object *bTheEmpire class *lexEmpire))

(m20! (class (m3 (lex empire))) (object b2))

(m20!)

*
;; There exists some end of the Empire.
```

```
(describe (assert member #bEnd class *lexEndEmpire))

(m21! (class (m4 (lex end of an empire))) (member b6))

(m21!)

[SNePS has assigned an instance of an end of an empire to base node b6.]

* (describe (assert object1 *bEnd rel (build lex "of") object2 *bTheEmpire))

(m23! (object1 b6) (object2 b2) (rel (m22 (lex of))))

(m23!)

[Here we have created a relation between the instance above (b6) and the Empire (b2).]

*
;; "World War I" is a war.
(describe (assert member *bWW1 class (build lex "war")))

(m25! (class (m24 (lex war))) (member b3))

(m25!)

*
;; The expression "Pyrrhic victory" consists of an adjective and a noun.
(describe (assert expr *lexPV str (build adj  (build lex "Pyrrhic")
                                        noun *lexVictory)))

(m28! (expr (m7 (lex Pyrrhic victory)))
 (str (m27 (adj (m26 (lex Pyrrhic))) (noun (m8 (lex victory))))))

(m28!)

*
;;============================================================================
;; BACKGROUND KNOWLEDGE RULES:
;;============================================================================

;; RULE: ADJ-NOUN
;; An ADJ-NOUN where ADJ is not abnormal is a NOUN.
(describe (assert forall ($vANExpr $vAdj $vNoun)
            ant (build expr *vANExpr str (build adj *vAdj noun *vNoun))
            cq  (build subclass *vANExpr superclass *vNoun)))

(m29! (forall v3 v2 v1) (ant (p2 (expr v1) (str (p1 (adj v2) (noun v3)))))
 (cq (p3 (subclass v1) (superclass v3))))

(m29!)

*
;; Desired consequent of ADJ-NOUN rule:
;; (describe (assert subclass *lexPV superclass *lexVictory))

;; RULE: BRITISH-EMPIRE
;; If an empire is British, then it is ruled by Great Britain.
(describe (assert forall ($vBritEmp)
            &ant ((build object *vBritEmp
                         property *lexBritish)
                  (build object *vBritEmp
                         class *lexEmpire))
            cq   (build object1 *bBritain
                        rel (build lex "rules")
```

```
                      object2 *vBritEmp)))

(m31! (forall v4)
 (&ant (p5 (class (m3 (lex empire))) (object v4))
  (p4 (object v4) (property (m1 (lex British)))))
 (cq (p6 (object1 b1) (object2 v4) (rel (m30 (lex rules))))))

(m31!)

*
;; Desired consequent of BRITISH-EMPIRE rule:
;; (describe (assert object1 *bBritain rel (build lex "rules")
;;                   object2 *bTheEmpire))

;; RULE: BRITISH-INVOLVEMENT
;; If X is an involvment and X is British, then it is of Great Britain.
(describe (assert forall ($vBritInv)
            &ant ((build member *vBritInv
                         class *lexInvolvement)
                  (build object *vBritInv
                         property *lexBritish))
            cq   (build object *vBritInv
                        rel *lexInvolvement
                        possessor *bBritain)))

(m32! (forall v5)
 (&ant (p8 (object v5) (property (m1 (lex British))))
  (p7 (class (m5 (lex involvement))) (member v5)))
 (cq (p9 (object v5) (possessor b1) (rel (m5)))))

(m32!)

*
;; Desired consequent of BRITISH-INVOLVEMENT rule:
;; (describe (assert object *bInv possessor *bBritain))

;; RULE: END-OF-EMPIRE
;; If X contributes to the end of the empire E and the political unit P rules
;; the empire E, then X weakens P.
(describe (assert forall ($vContributer $vAnEnd $vPolUnit $vAnEmpire)
            &ant ((build agent *vContributer
                         act (build action (build lex "contributes to")
                                   object *vAnEnd))
                  (build member *vAnEnd
                         class *lexEndEmpire)
                  (build object1 *vAnEnd
                         rel (build lex "of")
                         object2 *vAnEmpire)
                  (build object1 *vPolUnit
                         rel (build lex "rules")
                         object2 *vAnEmpire))
            cq   (build agent *vContributer
                        act (build action (build lex "weakens")
                                  object *vPolUnit))))

(m35! (forall v9 v8 v7 v6)
 (&ant (p14 (object1 v8) (object2 v9) (rel (m30 (lex rules))))
  (p13 (object1 v7) (object2 v9) (rel (m22 (lex of))))
  (p12 (class (m4 (lex end of an empire))) (member v7))
  (p11 (act (p10 (action (m33 (lex contributes to))) (object v7))) (agent v6)))
 (cq (p16 (act (p15 (action (m34 (lex weakens))) (object v8))) (agent v6))))

(m35!)
```

```
*
;; Desired consequent of END-OF-EMPIRE rule:
;; (describe (assert object1 *bInv rel (build lex "weakens")
;;                   object2 *bBritain))

;; RULE: HEAVY-LOSSES
;; If X is an involvement, W is a war, X is in W, X is costly in lives and
;; money, and X is possessed by Y, then Y incurs heavy losses in W.
(describe (assert forall ($vAnInv $vWar $vPossessor)
            &ant ((build member *vAnInv
                         class *lexInvolvement)
                  (build member *vWar
                         class (build lex "war"))
                  (build object1 *vAnInv
                         rel (build lex "in")
                         object2 *vWar)
                  (build object *vAnInv
                         property *lexCostlyLM)
                  (build object *vAnInv
                         possessor *vPossessor))
            cq   (build object1 *vPossessor
                        rel (build lex "incurs heavy losses in")
                        object2 *vWar)))

(m38! (forall v12 v11 v10)
 (&ant (p21 (object v10) (possessor v12))
  (p20 (object v10) (property (m2 (lex costly in lives and money))))
  (p19 (object1 v10) (object2 v11) (rel (m36 (lex in))))
  (p18 (class (m24 (lex war))) (member v11))
  (p17 (class (m5 (lex involvement))) (member v10)))
 (cq
  (p22 (object1 v12) (object2 v11) (rel (m37 (lex incurs heavy losses in))))))

(m38!)

*
;; Desired consequent of HEAVY-LOSSES rule:
;; (describe (assert object1 *bBritain rel (build lex "incurs heavy losses in")
;;                   object2 *bWW1))

;; RULE: WAR-VICTORY
;; If X is an involvement, W is a war, X is in W, X is a victory, and X is
;; possessed by Y, then Y is victorious in W.
(describe (assert forall ($vAnInv $vWar $vPossessor)
            &ant ((build member *vAnInv
                         class *lexInvolvement)
                  (build member *vWar
                         class (build lex "war"))
                  (build object1 *vAnInv
                         rel (build lex "in")
                         object2 *vWar)
                  (build member *vAnInv
                         class *lexVictory)
                  (build object *vAnInv
                         possessor *vPossessor))
            cq   (build object1 *vPossessor
                        rel (build lex "is victorious in")
                        object2 *vWar)))

(m40! (forall v15 v14 v13)
 (&ant (p27 (object v13) (possessor v15))
  (p26 (class (m8 (lex victory))) (member v13))
```

```
  (p25 (object1 v13) (object2 v14) (rel (m36 (lex in))))
  (p24 (class (m24 (lex war))) (member v14))
  (p23 (class (m5 (lex involvement))) (member v13)))
 (cq (p28 (object1 v15) (object2 v14) (rel (m39 (lex is victorious in))))))

(m40!)

*
;; Desired consequent of WAR-VICTORY rule:
;; (describe (assert object1 *bBritain rel (build lex "is victorious in")
;;                   object2 *bWW1))

;; RULE: MEMBER-SUPERCLASS
;; If X is a member of the class Y, and Y is a subclass of the superclass Z,
;; then X is a member of the class Z.
(describe (assert forall ($vMember $vSubClass $vSuperClass)
            &ant ((build member *vMember
                         class *vSubClass)
                  (build subclass *vSubClass
                         superclass *vSuperClass))
            cq   (build member *vMember
                        class *vSuperClass)))

(m41! (forall v18 v17 v16)
 (&ant (p30 (subclass v17) (superclass v18)) (p29 (class v17) (member v16)))
 (cq (p31 (class v18) (member v16))))

(m41!)

*
;; Desired consequent of MEMBER-SUPERCLASS rule:
;; (describe (assert member *bInv class *lexVictory))

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

[In this section, I used 'add' instead of 'assert' when teaching Cassie things.  This
allows SNIP to perform forward inference to generate new propositions other than the
one that I have added.]

;; The involvement was British.
(describe (add object *bInv property *lexBritish))

(m50! (subclass (m7 (lex Pyrrhic victory))) (superclass (m8 (lex victory))))
(m45! (object b5) (possessor b1))
(m44! (object b5) (possessor b1) (rel (m5 (lex involvement))))
(m42! (object b5) (property (m1 (lex British))))
(m28! (expr (m7)) (str (m27 (adj (m26 (lex Pyrrhic))) (noun (m8)))))
(m25! (class (m24 (lex war))) (member b3))
(m21! (class (m4 (lex end of an empire))) (member b6))
(m18! (class (m6 (lex political unit))) (member b1))
(m17! (class (m5)) (member b5))
(m16! (class (m15 (lex advantage))) (member b4))

(m50! m45! m44! m42! m28! m25! m21! m18! m17! m16!)

[Here SNIP is asked for the first time to 'add' new propositions, so the list is
rather long, however many propositions listed were already added to Cassie's
knowledge.  The ADJ-NOUN rule fires first, creating m50, and m44 is created by the
BRITISH-INVOLVEMENT rule.]

*
```

```
;; The involvement was in World War I.
(describe (add object1 *bInv rel (build lex "in") object2 *bWW1))

(m51! (object1 b5) (object2 b3) (rel (m36 (lex in))))

(m51!)

*
;; The involvement was unnecessary.
(describe (add object *bInv property (build lex "unnecessary")))

(m53! (object b5) (property (m52 (lex unnecessary))))

(m53!)

*
;; The involvement was costly in lives and money.
(describe (add object *bInv property *lexCostlyLM))

(m54! (object1 b1) (object2 b3) (rel (m37 (lex incurs heavy losses in))))
(m46! (object b5) (property (m2 (lex costly in lives and money))))

(m54! m46!)

[SNIP is able to create m54 through the HEAVY-LOSSES rule after m46 is added to the
knowledge base.]

*
;; The advantage was gained from the involvement.
(describe (add object1 *bAdv
            rel (build lex "gained from")
            object2 *bInv))

(m56! (object1 b4) (object2 b5) (rel (m55 (lex gained from))))

(m56!)

*
;; The involvement was more costly ... than the advantage (gained from it).
(describe (add object1 *bInv
            rel (build mod (build lex "more") head *lexCostlyLM)
            object2 *bAdv))

(m59! (object1 b5) (object2 b4)
 (rel (m58 (head (m2 (lex costly in lives and money))) (mod (m57 (lex more))))))

(m59!)

*
;; The involvement was a Pyrrhic victory...
(describe (add member *bInv class *lexPV))

(m61! (object1 b1) (object2 b3) (rel (m39 (lex is victorious in))))
(m60! (class (m7 (lex Pyrrhic victory))) (member b5))
(m47! (class (m8 (lex victory))) (member b5))

(m61! m60! m47!)

[SNIP adds both m61 (through the WAR-VICTORY rule) and m47 (through the MEMBER-
SUPERCLASS rule) after m60 is added to the knowledge base.

*
;; ... that contributed to the end of the Empire.
```

```
(describe (add agent *bInv
               act (build action (build lex "contributes to")
                         object *bEnd)))

(m66! (act (m65 (action (m34 (lex weakens))) (object b1))) (agent b5))
(m64! (object1 b1) (object2 b2) (rel (m30 (lex rules))))
(m63! (act (m62 (action (m33 (lex contributes to))) (object b6))) (agent b5))
(m42! (object b5) (property (m1 (lex British))))
(m23! (object1 b6) (object2 b2) (rel (m22 (lex of))))
(m21! (class (m4 (lex end of an empire))) (member b6))
(m20! (class (m3 (lex empire))) (object b2))
(m19! (object b2) (property (m1)))

(m66! m64! m63! m42! m23! m21! m20! m19!)
```

[Here, SNIP adds m66 (through the END-OF-EMPIRE rule) and m64 (through the BRITISH-EMPIRE rule) after m63 is learned. Note, however, that the creation of m64 did not require knowledge of m63, it was simply a side effect of SNIP's forward inference passing through the "the Empire" base node.]

```
*
;;=============================================================================
;; CASSIE'S DEFINITION
;;=============================================================================

^(
--> defineNoun "Pyrrhic victory")
 Definition of Pyrrhic victory:
 Class Inclusions: victory,
 Possible Actions: weakens political unit, contributes to end of an empire,
 Possible Properties: unnecessary, costly in lives and money, British, in war,
 Possessive: political unit involvement,
nil
```

[Cassie's definition of "Pyrrhic victory" is very much like we had hoped for, saying that it weakens the political unit. Thanks for bearing with me so long.]

```
*

End of /home/csgrad/bdanger/cse663/Pyrrhic.demo demonstration.
```