Computationally Defining 'harbinger' via Contextual Vocabulary Acquisition

Albert Goldfain
CSE663: Advanced KRR
ag33@cse.buffalo.edu

December 12, 2003

Abstract

The Contextual Vocabulary Acquisition (CVA) project is an effort to computationally model the task a human reader faces when encountering an unknown word in a given passage. An attempt is made by the reader to define the word using only the context of the passage and an appropriate set of background rules. The SNePS semantic network system is used as a knowledge representation language for our reader (a computational SNePS based agent named CASSIE). After being given the SNePS representation of the passage and background rules, CASSIE uses a specific CVA definition algorithm based on the part of speech of the unknown word. In this study, we apply the CVA noun definition algorithm to define the unknown word "harbinger". We provide an annotated demo of CASSIE's interaction with the SNePS representation of the passage. We then compare CASSIE's results to results of tests performed on human readers who are faced with the same task (we refer to such tests as verbal protocols). We conclude that CASSIE's definition of "harbinger" from the given passage is an acceptable one. A speculation is offered as to whether or not the form of the given passage assists CASSIE's in producing such results. Additional ideas for potential research are also discussed.

1 The CVA Project and SNePS

As human beings, we must admit that we are not equipped with either the patience or the diligence to look up every unfamiliar word we read. We encounter similar barriers when dealing with computational agents. It is impossible to provide a computational agent with an exhaustive lexicon for any human language. All spoken and written languages used in the real world are inherently dynamic. From a cognitive science perspective, it is much more satisfying to have a computational agent discover for itself the meanings of new words rather than having each new word "hard coded". Contextual Vocabulary Acquisition (CVA) is an alternative to such hard coding. Given a passage containing an unknown word, we apply CVA by defining the unknown word using contextual information and our previous background knowledge.

The method of computational CVA we use is as follows:

1. We represent a sufficient set of background rules and facts to adequately understand the concepts in the passage (except, of course, the target unknown word)

2. We then represent the passage itself

3. We allow a computational agent named CASSIE to "read" the passage, forming an internal representation as she goes

4. Finally we invoke the appropriate definition algorithm and allow CASSIE to make an attempt at defining the unknown word

We must ensure that our representations for steps (1) and (2) are neutral and unbiased with respect to our definition algorithm. We cannot hand CASSIE the result on a silver platter by disguising the definition as either background knowledge or part of the passage. We also should make sure that we are representing the passage *as given*, not a CVA-friendly version of the passage. To do otherwise would make our system a "toy system" in which all passages must be taken from the CVA-friendly subset of English. The end product of this process is CASSIE's *attempted* definition of the unknown word.

The CVA project investigates both the performance and the implementation of context based definition algorithms. The algorithmic procedure used depends on the part of speech of the unknown word. In this paper, we invoke the noun definition algorithm. This is the most developed algorithm in the CVA suite of algorithms.

When an agent can rely on the robustness and accuracy of its CVA algorithms, it can begin to confidently use the resulting definitions. Ideally, the definitions of unknown words will become "known" after several encounters (in future passages). According to Rapaport and Ehrlich, CVA should be powerful enough so that "...a meaning for a word *can* be determined from any context, can be *revised* and refined upon further encounters with it, and '*converges*' to a dictionary-like definition given enough context and exposures to it" (Rapaport and Ehrlich 2000). These "known" words can then be used to help discover the meaning of a new unknown word. Thus, if our CVA definition algorithms are reasonably

strong, we will have a self sustaining tool for building a larger vocabulary.

We use the SNePS semantic network system for representing our passage and background knowledge (for more information on SNePS, visit `http://www.cse.buffalo.edu/sneps`). SNePS is a very natural choice for this type of application. SNePS generates propositional semantic networks, so it is a good fit for modeling any language-centric problem. We code networks using SNePSUL, a LISP-like user language which defines a set of nodes and the relations (arcs) between them. The resulting network is viewed as a snapshot of CASSIEs mind (or, if the reader finds "mind" to be too strong a term, we may say "knowledge base"). For our application, the snapshots of interest are the one of CASSIEs network before reading the passage and the one after reading the passage. She will attempt to define the unknown word immediately after reading the passage. The power of the resulting "definitions" in SNePS is that they will be more than just atomic building blocks. A definition will be a network of interrelated concepts. Quillian's first application for semantic networks was to create a model which is "built up within a computer by 'recoding' a body of information from an ordinary dictionary into a complex network of elements and associations interconnecting them" (Quillian 1967). The same can be said of what we are attempting in SNePS. Interrelations between the passage and the unknown word are the direct result of CASSIE reading the passage. With each new context, new interrelations are formed and old ones are refined. This is why definitions will tend to strengthen (i.e. converge) as an unknown word is encountered in different contexts.

2 The Passage and a Dictionary Definition

Our source passage is taken from the web site

`http://cnet.windsor.ns.ca/Environment/Advocates/Anim/robin.html`:

> "The American Robin is called the 'harbinger of Spring' because of its early northward migration, which brings its arrival before most other migratory birds, and because of its size and song, along with its habit of living close to human development, it is usually the first of the summer birds to be noted by humans"

The unknown target noun we have selected from this passage is "harbinger".

Stylistically speaking, the passage is a run-on sentence, making it hard for even humans to read. We therefore have condensed the original into the following passage:

> "The American Robin is called the 'harbinger of Spring' because of its early northward migration, which brings its arrival before most other migratory birds ... it is usually the first of the summer birds to be noted by humans"

Clearly, we have not mutilated the semantic content of this passage by removing the section we did. The revised passage contains a subset of the original passage's contextual information. We therefore conclude that with enough time both the human and CASSIE could work through the longer version.

Since we are working towards a "dictionary-like definition", we give one here (courtesy of `http://www.dictionary.com`):

> harbinger-*n.* One that indicates or foreshadows what is to come; a forerunner.

Other usages are given, but this is clearly the one intended for this passage.

3 Verbal Protocols

The passage was given to several human subjects who did not know the meaning of the word harbinger. The reader was asked to read the passage and give a definition for the word harbinger. The following two results (for a male subject *CC* and a female subject *AL*) represent typical responses:

CC: "...harbinger means one who brings something or something which is the first thing you see because it arrives early..."

AL: "...a harbinger is an early indicator of some time period ...so the American Robin is the harbinger for the time period Spring"

We immediately notice that both of these definitions are very close to our dictionary definition. Also, we see that both types of responses use contextual information from the source passage (the words 'first', 'early', 'arrive' and 'Spring' are all mentioned in the responses). Clearly context is being used as a crutch by the human readers and, in this particular passage, human CVA yields a workable definition for harbinger. This result is a harbinger of good news for CASSIE's CVA prospects!

The use of the word "bringer" in *CC*'s response may be an artifact of the lexicographic similarity between "bringer" and the final part of "harbinger", but it is hard to tell in ret-

rospect. Also, the use of "time period" in *AL*'s response suggests that she is ready to use her definition to identify entities such as the "harbinger of Winter" or the "harbinger of Dawn" (if such entities exist). This is not something we should expect from CASSIE. If our passage is doing anything contextually, it is providing reasons why the American Robin is called the harbinger of *Spring*. *AL* may be making this leap in reasoning because she sees the passage contains two temporal concepts (the seasons Spring and Summer). Even though we will be using the CVA noun algorithm to define the noun "harbinger", as far as CASSIE is concerned, we may as well treat the noun phrase "harbinger of Spring" as our "atomic" target for definition. In other words, harbingers of Spring are all CASSIE will get a handle on by reading the given passage.

4 Our SNePS Representation

4.1 SNePS Case Frames: Syntax and Semantics

The CVA noun definition algorithm recognizes a small set of SNePS case frames. Of these we will use the following in our representation:

- *agent/act/action*
- *lex*
- *member/class*
- *object1/rel/object2*
- *object/proper-name*
- *object/property*
- *object/rel/possessor*
- *superclass/subclass*

The syntax and semantics for these case frames can be found at `http://www.cse.buffalo.edu/ stn2/cva/case-frames`. We will strictly adhere to the syntax and semantics given there.

In addition, we also will need the following case frames from the *SNePS case frame dictionary*:

- *forall/ant/cq*

- *min/max/arg*

A PDF version of the *SNePS Case Frame Dictionary* can be found at `http://www.cse.buffalo.edu/sneps/`. This document gives the syntax and semantics for these case frames. Finally we will use the following "non-standard" case frames in our representation

- *agent/act/time*
- *before/after*
- *mod/head*
- *equiv/equiv*
- *cause/effect*
- *action/from/to*

The syntax and semantics for these case frames is given in Appendix A.

4.2 Background Knowledge Representation

CASSIE is first given an adequate set of background knowledge about the concepts presented in the passage. This knowledge is expressed as a set of rules or facts and should be as general as possible. In fact, we are only representing a slice of information which we feel is necessary to approach the given passage. The amount of background knowledge can grow exponentially with the number of unique concepts in the passage. Clearly some knowledge engineering is involved in selecting the most appropriate subset for our background information. Our immediate representation of this background knowledge brings it temporarily to

the "foreground" (i.e. activates it for use) so that CASSIE can apply CVA.

A complete SNePSUL representation of CASSIE's background knowledge is given in Appendix B and the semantic network diagrams for this representation are given in Appendix C. What follows is a high level description of the background knowledge. We will use first order logic (FOL) to describe some of this knowledge and "pretend-it's-English" semantics will be more than adequate for our present discussion [1]

The first few background rules provide some factual information for CASSIE.

(BGI) The American Robin is a bird. In FOL: $Bird(AmericanRobin)$

(BGII) Spring is a season. In FOL: $Season(Spring)$

(BGIII) Summer is a season. In FOL: $Season(Summer)$

(BGIV) Spring is before Summer. In FOL: $Before(Spring, Summer)$

(BGV) The American Robin is a migratory bird. In FOL: $MigratoryBird(AmericanRobin)$

Clearly CASSIE would be missing a great deal if any of these five were missing from her knowledge base. American Robin is expressed as a single entity, rather than using a mod-head case frame like we do for other concepts such as "migratory birds" (see below for details). This can be done because we do not refer to any other kinds of robin in the passage. We could also add a fact that the American Robin is migratory, but she will have

[1] Here, we are just using FOL to illustrate the background knowledge we are providing for CASSIE. Our actual SNePSUL representation will rigorously follow the case frame syntax and semantics

enough information to infer this from (BGI) and (BGV).

Next we provide the following important rule:

(BGVI) If x is z's y (that is, a y of z) then x is a y.

This common sense rule will link the American Robin and the unknown word harbinger. We will represent the fragment "The American Robin is called the harbinger of Spring ..." as the possessive "The American Robin is Spring's Harbinger ..." (see the passage representation section below for a further discussion of this representation). This background rule must therefore also take the possessive form.

In contrast to the previous rules, (BGVII) may seem unnatural at first:

(BGVII) If x is a y bird and y is not a season, then x is y.

This rule will allow CASSIE to pull apart the "mod-head hierarchies" we use for representing noun-noun and adjective-noun phrasings in our passage (see below for details). We want this rule to allow CASSIE to infer that the American Robin is migratory since it is a migratory bird (BGV). The "y is not a season" part of the antecedent forbids CASSIE from concluding incorrect concepts such as: "A Summer bird both a bird and summer". Unfortunately, such a rule will need to be expanded/altered the instant we move on to another passage. Also, this piece of knowledge is not easily representable in FOL. However, it is not

an unreasonable piece of background information.

The next rule makes the association between the adjective "migratory" and the verb "migrate"

(BGVIII) If x is migratory then x migrates. In FOL: $\forall x[Migratory(x) \supset Migrates(x)]$

The next four background rules define some important concepts dealing with the relationship "first of".

(BGIX) If x is the first of y then x is an indicator that y has arrived.

(BGX) If x is the first of y then x is a y.

(BGXI) If x is the first of y and z is the first of y then x and z are equivalent.

(BGXII) If x is the first of y and x is not equivalent to z then z is not the first of y.

(BGIX) allows us to tie the concept of being 'first' with the concept of arrival in our passage. The careful reader may notice that (BGIX) is not true in general. For example, a soldier who is a scout for a large army may be stationed well ahead of the army. If enemy forces discover his presence, they will not assume that his army has arrived (they may be several days behind). To get around this, we could add a background fact about migration and say that birds usually travel together. In groups that travel together, we could then say that the first member will indicate the arrival of the group. We also could rephrase this rule using the modal logic notion of possibility (via the *mode/object* case frame in SNePS). This was

attempted, but the result was simply that the noun algorithm ignored this rule.

(BGX) through (BGXII) are much more natural. (BGX) indicates that the first of a set is a member of that set. (BGXI) and (BGXII) preserve the uniqueness of the first of a set (i.e. force each set into having only one first).

The final background rule defines the concept "other" for CASSIE:

BG(XIII) If x is y and z is another y then x is not equivalent to z

4.3 Passage Representation

In designing a suitable representation for the passage itself, it is useful to think of the harbinger passage in terms of the following four segments:

$$\underbrace{\text{“}\textit{The American Robin is called the `harbinger of Spring'}}_{I}\ \underbrace{\textit{because of its early northward migration}}_{II},$$
$$\underbrace{\textit{which brings its arrival before most other migratory birds}}_{III}$$
$$\underbrace{\textit{... it is usually the first of the summer birds to be noted among humans}\text{”}}_{IV}$$

When viewed in this way, we see that segment II is the cause of both segment I and segment III. This causal relationship ties the first three segments together. Segment IV is refines what the first three segments are saying by introducing the all important concept of the American Robin being the "first" in a set.

Figures 14-16 in Appendix C illustrate the SNePS network diagrams for the passage rep-

resentation. Here we will give our SNePSUL representation. We represent segment I as follows:

```
;-------------------------------------------------------
; I. THE AMERICAN ROBIN IS CALLED THE "HARBINGER OF SPRING"
;-------------------------------------------------------

(describe (add object *AmericanRobin
                  rel (build lex harbinger) = HARBINGERS
                  possessor *Spring) = ARCHOS)
```

We use the possessive construction here to represent "The American Robin is Spring's harbinger". This is semantically equivalent to "The American Robin is the harbinger of spring" [2]. Even though the passage only says that the American Robin is *called* the harbinger of Spring, it is clear to the human reader that the American Robin *is* the harbinger of Spring. Thus, to make CASSIE's life a little easier, we assume she is not reading a text where entities can be called something that they really are not (a sufficient set of additional background rules might handle such cases anyway).

We also acknowledge the fact that we are representing a single American Robin rather than the entire class of American Robins. Every Robin in the set of American Robins is also a "harbinger of Spring". It is clear that, even if we were to represent the entire class here, we could perform a universal instantiation with respect to the "harbinger of Spring" property

[2] in fact, the case frame semantics for object/rel/possessor are exactly what is called for in "[[x]] is the [[y]] of [[z]]" constructions

to obtain a single American Robin[3].

We now turn to our representation of segment II:

```
;-------------------------------------------
; II. BECAUSE OF ITS EARLY NORTHWARD MIGRATION
;-------------------------------------------
(describe (add cause (add agent *AmericanRobin
                                act (build action *MIGRATES) = MIGRATION
                        ) = ENM
                  effect *ARCHOS
          )
)

(describe (add object *MIGRATION property (build lex early)))
(describe (add action *MIGRATES from #STARTLOC to #ENDLOC))
(describe (add object1 *ENDLOC rel (build lex north) object2 *STARTLOC))
```

The standard agent/act/action construct is used for representing the American Robin's migration. The earliness of the migration is represented as a property of the act. We represent the direction northward by adding the from/to arcs to the action, and indicating that the end location is "north of" the starting location. A very subtle point should be noted about the phrase "early northward migration". Because the human readers of this passage are all reading this passage in America, and because we are dealing with an *American* Robin (which presumably returns to *America* in the springtime), the "early northward migration" is a natural thing to read. A reader in Australia, for example, who is used to birds returning in the Spring by traveling southward, would have to reorient their background knowledge

[3]Rapaport indicates that such representations will change in the upcoming SNePS3

to the specific context of the passage. This "early northward migration" is a cause of the American Robin being called the harbinger of Spring (abbreviated in the SNePSUL as a pointer to ARCHOS). This is also a direct cause for the effect which we represent in segment III:

```
;-----------------------------------------------------------------
; III. WHICH BRINGS ITS ARRIVAL BEFORE MOST OTHER MIGRATORY BIRDS
;-----------------------------------------------------------------
(describe (add before #ARRIVETIME1 after #ARRIVETIME2))
(describe (add cause *ENM
                  effect (add agent *AmericanRobin
                              act (build action *ARRIVAL)
                              time *ARRIVETIME1
                         )
          )
)

(describe (add agent (build mod (build lex other)
                            head *MIGRATORYBIRDS
                     )

               act (build action *ARRIVAL)
               time *ARRIVETIME2
          )
)
```

Here, we first represent two arrival times as base nodes, one before the other. We then "assign" the American Robin's arrival to the earlier time by using the temporal time arc on the agent/act case frame. To the later arrival time we "assign" the arrival of "other migratory birds". We notice that the action of arrival is shared by both the American Robin and the "other migratory birds", even though the events are separated temporally. The phrase "other migratory birds" is built up as a "mod/head" hierarchy as shown. The

background knowledge will handle the fact that the American Robin is *a* migratory bird, and that it is not one of the *other* migratory birds.

Finally we represent segment IV:

```
;-------------------------------------------------------------------
; IV. IT IS USUALLY THE FIRST OF THE SUMMER BIRDS TO BE NOTED BY HUMANS
;-------------------------------------------------------------------
(describe (add object1 *AmericanRobin
                  rel *FIRSTOF
                  object2 #SummerBirdsNotedByHumans
          )
)
(describe (add subclass *SummerBirdsNotedByHumans
                  superclass (build mod *Summer head *BIRDCLASS)
          )
)
(describe (add subclass *SummerBirdsNotedByHumans
                  superclass (build mod (build lex notedBy) head (build lex humans))
          )
)
```

To simplify this representation, we drop the word "usually" (it is not even clear if "usually" is captured by any modal logic operators). We represent the set of "summer birds noted by humans" as a base node, and assert that the American Robin is the first of this set. We then refine the base node by making it a subclass of the class of summer birds as well as a subclass of the class of all things noted by humans.

5 Results

After providing the SNePS representation for both the background information and the harbinger passage, we invoke the CVA noun definition algorithm and obtain the following output:

```
; Ask Cassie what "harbinger" means:
^(
--> defineNoun "harbinger")
 Definition of harbinger:
 Possible Class Inclusions: AmericanRobin, bird, m14, b8,
 Possible Actions: arrive, migrate,
 Possible Properties: indicates m52, first b8,
 Possessive: season,
```

We may fill in node names to obtain the following human readable definition:

```
; Ask Cassie what "harbinger" means:
^(
--> defineNoun "harbinger")
 Definition of harbinger:
 Possible Class Inclusions: AmericanRobin, bird, migratory bird, Summer birds noted by h
 Possible Actions: arrive, migrate,
 Possible Properties: indicates arrival of Summer birds noted by humans, first Summer bi
 Possessive: season,
```

As we expect, CASSIE's definition is almost an "instance" of the dictionary definition in that it really defines what a migratory-bird-Spring-harbinger is. CASSIE does not generalize beyond the current context, but the end result (in particular the possible actions and properties) are strong.

As a "sanity check" (and to further examine how the noun algorithm works) we ran the noun definition algorithm on "American Robin" and "season" with the following results:

```
* ^(defineNoun "AmericanRobin")

-->  Definition of AmericanRobin:
 Possible Class Inclusions: bird, m14, harbinger, b8,
 Possible Actions: arrive, migrate,
 Possible Properties: indicates m52, first b8,
 Possessive: season harbinger,
nil

* ^(defineNoun "season")

-->  Definition of season:
 Named Individuals: Summer, Spring,
nil
```

It is clear that some of the harbinger definition "spills over" into the American Robin definition (and vice versa) because the two entities are tied together by (BGVI) and passage segment I. If American Robin had been the unknown noun, this passage would have refined the definition to include "season harbinger" (listed under possessive).

6 Further Study and Speculation

The results we obtained could be strengthened significantly by the addition of more background knowledge. The two human protocol definitions given above both abstract away from the treatment of the seasons as points (or sets of point intervals) in time. We cannot expect this from CASSIE since we have only given her the two temporal entities "Spring" and "Summer" to work with. More background knowledge could allow CASSIE to consider the harbinger of any temporal concept (like AL did above).

We also have avoided making a conceptual tie between being "first" and being "early". To say that being "early" implies being "first" is simply wrong, and to say that being "first" in a set implies being "early" is only true if the entire set under discussion is not late. An attempt was made at adding such a concept using modal logic, but CASSIE seemingly ignored the modal logic background rules when applying the noun definition algorithm.

We speculate that another reason for our positive result was the form of the passage itself. Passages of the form x is y because of z are inherently "CVA-friendly". When our goal is to define y, it is clear that all of the context we will need will come from concepts in z. Our background information can draw on background facts we know about x. Having the luxury of such a structure is a definite benefit to the knowledge engineer. It would be an interesting exercise to catalogue such "CVA-friendly" structures in natural language (this would also help answer the question "Is the contextual strength of a syntactic form quantifi-

able"). Clearly, if CVA is going to work on a large scale, it *must* work for "CVA-friendly" passages.

We implemented this project on the passage representation side of CVA. The noun definition algorithm itself was treated as a black box. Based on the results alone, we speculate that the algorithm "found its contextual data" by branching out from the unknown word in the network (and adding relevant properties and class inclusions). It would be beneficial to now spend some time "under the hood" of the noun definition algorithm to see if any improvements can be made.

We conclude with the realization that "working" CVA is a very lofty goal on both the representation and implementation sides. A passage which is easy to read may be very difficult to represent (and vice versa). The knowledge engineer must make difficult choices in selecting the background knowledge and representing the source passage. The algorithm implementer essentially provides a "best effort" attempt to use the contextual data, without having the ability to alter its representation. Still, we remain optimistic that CVA can be incrementally improved to a "near human" level of performance.

Appendix A: Syntax and Semantics for Non-Standard Case Frames

Syntax

If x1 . . . x6, y1 . . . y6, z1 and z6 are individual nodes and m1 . . . m6 are identifiers not previously used, then each of the following are networks and m1 . . . m6 are structured proposition nodes.

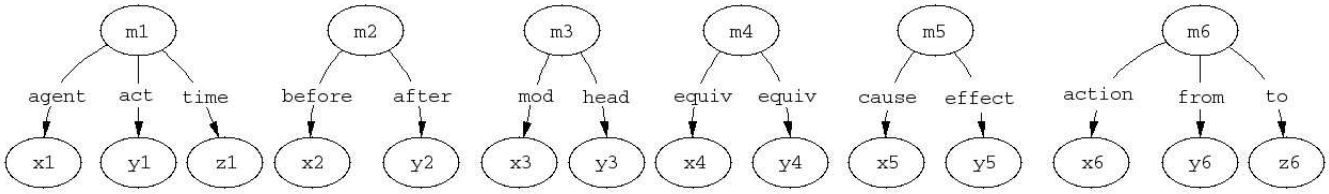

Figure 1: "Case Frame Syntax"

Semantics

- [[m1]] is the proposition that agent [[x1]] performs [[y1]] at time [[z1]]
- [[m2]] is the proposition that time [[x2]] is before time [[y2]]
- [[m3]] is the proposition that [[x3]] is a modifier for [[y3]]
- [[m4]] is the proposition that [[x4]] and [[y4]] are equivalent
- [[m5]] is the proposition that [[x5]] is the cause of effect [[y5]]
- [[m6]] is the proposition that the direction of action [[x6]] is from [[y6]] to [[z6]]

Appendix B: Script of Running Demo

```
Script started on Tue Dec 09 23:02:20 2003
pollux {~/cse663} > acl

International Allegro CL Enterprise Edition
6.2 [Solaris] (Oct 28, 2003 9:00)
Copyright (C) 1985-2002, Franz Inc., Berkeley, CA, USA.  All Rights Reserved.

This development copy of Allegro CL is licensed to:
   [4549] SUNY/Buffalo, N. Campus

;; Optimization settings: safety 1, space 1, speed 1, debug 2.
;; For a complete description of all compiler switches given the
;; current optimization settings evaluate (explain-compiler-settings).
;;---
;; Current reader case mode: :case-sensitive-lower
cl-user(1): (load "/projects/snwiz/bin/sneps")
; Loading /projects/snwiz/bin/sneps.lisp
Loading system SNePS...10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
SNePS-2.6 [PL:0a 2002/09/30 22:37:46] loaded.
Type `(sneps)' or `(snepslog)' to get started.
t
cl-user(2): (sneps)


   Welcome to SNePS-2.6 [PL:0a 2002/09/30 22:37:46]

Copyright (C) 1984--2002 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.

   12/9/2003 23:03:06

* (demo "harbinger.demo")

File /home/csgrad/ag33/cse663/harbinger.demo is now the source of input.


 CPU time : 0.00

* ; ======================================================================
; FILENAME: harbinger.demo
; DATE: 12/8/03
; PROGRAMMER: Albert Goldfain

; Lines beginning with a semi-colon are comments.
; Lines beginning with "^" are Lisp commands.
; All other lines are SNePS commands.
;
; To use this file: run SNePS; at the SNePS prompt (*), type:
;
; (demo "harbinger.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 noun definition algorithm:
^(
--> load "/projects/rapaport/CVA/STN2/defun_noun.cl")
; Loading /projects/rapaport/CVA/STN2/defun_noun.cl
t



 CPU time : 0.18

*
; Clear the SNePS network:
(resetnet)

Net reset - Relations and paths are still defined


 CPU time : 0.00

*

; load all pre-defined relations:
;(intext "/projects/rapaport/CVA/STN2/rels")
(intext "myRels")
File myRels is now the source of input.


 CPU time : 0.01

* act is already defined.
action is already defined.
effect is already defined.
object1 is already defined.
object2 is already defined.

(a1 a2 a3 a4 act action after agent antonym associated before cause
 class direction effect equiv etime from head in indobj instr into lex
 location kn_cat manner member members mode mod object objects object1
 objects1 object2 on onto part place possessor proper-name property
 purpose rel skf stime subclass superclass synonym time to whole)

 CPU time : 0.02

*

End of file myRels


 CPU time : 0.02
```

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


 CPU time : 0.00

*
before implied by the path (compose before
                            (kstar (compose after- ! before)))
before- implied by the path (compose (kstar (compose before- ! after))
                             before-)


 CPU time : 0.00

*
after implied by the path (compose after
                           (kstar (compose before- ! after)))
after- implied by the path (compose (kstar (compose after- ! before))
                            after-)


 CPU time : 0.00

*
sub1 implied by the path (compose object1- superclass- ! subclass
                          superclass- ! subclass)
sub1- implied by the path (compose subclass- ! superclass subclass- !
                           superclass object1)


 CPU time : 0.00

*
super1 implied by the path (compose superclass subclass- ! superclass
                            object1- ! object2)
super1- implied by the path (compose object2- ! object1 superclass- !
                             subclass superclass-)


 CPU time : 0.00

*
superclass implied by the path (or superclass super1)
superclass- implied by the path (or superclass- super1-)


 CPU time : 0.00

*

End of file myPaths


 CPU time : 0.01

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

```
;-------------------------------------------------------
; BG I. The American Robin is a bird
;-------------------------------------------------------
(describe (add member #AmericanRobin class (build lex AmericanRobin)))

(m2! (class (m1 (lex AmericanRobin))) (member b1))

(m2!)

 CPU time : 0.00

* (describe (add member *AmericanRobin class (build lex bird)=BIRDCLASS))

(m4! (class (m3 (lex bird))) (member b1))

(m4!)

 CPU time : 0.01

* ;-------------------------------------------------------
; BG II. Spring is a season
;-------------------------------------------------------
(describe (add object #Spring proper-name (build lex Spring)))

(m6! (object b2) (proper-name (m5 (lex Spring))))

(m6!)

 CPU time : 0.00

* (describe (add member *Spring class (build lex season)=SEASONS))

(m8! (class (m7 (lex season))) (member b2))

(m8!)

 CPU time : 0.00

* ;-------------------------------------------------------
; BG III. Summer is a season
;-------------------------------------------------------
(describe (add object #Summer proper-name (build lex Summer)))

(m10! (object b3) (proper-name (m9 (lex Summer))))

(m10!)

 CPU time : 0.00

* (describe (add member *Summer class *SEASONS))

(m11! (class (m7 (lex season))) (member b3))

(m11!)

 CPU time : 0.01

* ;-------------------------------------------------------
; BG IV. Spring is before Summer (or Summer is after Spring)
;-------------------------------------------------------
(describe (add before *Spring after *Summer))
```

```
(m12! (after b3) (before b2))

(m12!)

 CPU time : 0.03

* ;-------------------------------------------------------------
; BG V. The American Robin is a migratory bird.
;-------------------------------------------------------------
(describe (add member *AmericanRobin
       class (build mod (build lex migratory)=MIGRATORYTHING
                        head *BIRDCLASS)=MIGRATORYBIRDS
          )
)

(m15! (class (m14 (head (m3 (lex bird))) (mod (m13 (lex migratory)))))
 (member b1))

(m15!)

 CPU time : 0.00

* ;-------------------------------------------------------------
; BG VI. If x is z's y then x is a y
;-------------------------------------------------------------

(describe (add forall ($x $y $z)
ant (build object *x
   rel *y
   possessor *z)
cq (build member *x class *y)))

(m16! (forall v3 v2 v1) (ant (p1 (object v1) (possessor v3) (rel v2)))
 (cq (p2 (class v2) (member v1))))

(m16!)

 CPU time : 0.01

*
;-------------------------------------------------------------
; BG VII. If x is a y bird and y is not a season then x is y
;-------------------------------------------------------------

(describe (add forall ($x $y)
&ant (build member *x
   class (build mod *y head *BIRDCLASS))
&ant (build min 0 max 0 arg(build member *y class *SEASONS))
cq (build object *x property *y)
  )
)

(m17! (forall v5 v4)
 (&ant
  (p6 (min 0) (max 0) (arg (p5 (class (m7 (lex season))) (member v5))))
  (p4 (class (p3 (head (m3 (lex bird))) (mod v5))) (member v4)))
 (cq (p7 (object v4) (property v5))))
(m15! (class (m14 (head (m3)) (mod (m13 (lex migratory))))) (member b1))

(m17! m15!)
```

```
 CPU time : 0.01

*
;-------------------------------------------------------------
; BG VIII. If x is migratory then x migrates
;-------------------------------------------------------------
(describe (add forall $x
ant (build object *x property *MIGRATORYTHING)
cq (build agent *x
                          act (build action (build lex migrate)=MIGRATES)
                    )
  )
)

(m20! (forall v6)
 (ant (p10 (object v6) (property (m13 (lex migratory)))))
 (cq (p11 (act (m19 (action (m18 (lex migrate))))) (agent v6))))
(m15! (class (m14 (head (m3 (lex bird))) (mod (m13)))) (member b1))

(m20! m15!)

 CPU time : 0.01

* ;-------------------------------------------------------------
; BG IX. If x is the first of y then x is an indicator that
;        y has arrived
;-------------------------------------------------------------
(describe (build lex arrive)=ARRIVAL)


 CPU time : 0.00

* (describe (add forall ($x $y)
ant (build object1 *x
                           rel (build lex first)=FIRSTOF
   object2 *y
    )
cq (build object1 *x
                          rel (build lex indicates)
  object2 (build agent *y
         act (build action *ARRIVAL)
  )

                    )
          )
)

(m27! (forall v8 v7)
 (ant (p16 (object1 v7) (object2 v8) (rel (m24 (lex first)))))
 (cq
  (p18 (object1 v7)
   (object2 (p17 (act (m26 (action (m23 (lex arrive))))) (agent v8)))
   (rel (m25 (lex indicates))))))

(m27!)

 CPU time : 0.00

* ;-------------------------------------------------------------
; BG X, XI and XII. Defining the "first" of a class
```

```
;-------------------------------------------------------
(describe (add forall ($x $y)
ant (build object1 *x rel *FIRSTOF object2 *y)
cq (build member *x class *y)
          )
)

(m28! (forall v10 v9)
 (ant (p19 (object1 v9) (object2 v10) (rel (m24 (lex first)))))
 (cq (p20 (class v10) (member v9))))

(m28!)

 CPU time : 0.01

*
(describe (add forall ($x $y $z)
&ant (build object1 *x rel *FIRSTOF object2 *y)
&ant (build object1 *z rel *FIRSTOF object2 *y)
cq (build equiv *x equiv *z)
  )
)

(m29! (forall v13 v12 v11)
 (&ant (p22 (object1 v13) (object2 v12) (rel (m24 (lex first))))
  (p21 (object1 v11) (object2 v12) (rel (m24))))
 (cq (p23 (equiv v13 v11))))

(m29!)

 CPU time : 0.01

*
(describe (add forall ($x $y $z)
&ant (build object1 *x rel *FIRSTOF object2 *y)
&ant (build min 0 max 0 arg(build equiv *x equiv *z))
cq (build min 0 max 0 arg(build object1 *z rel *FIRSTOF object2 *y))
  )
)

(m30! (forall v16 v15 v14)
 (&ant (p26 (min 0) (max 0) (arg (p25 (equiv v16 v14))))
  (p24 (object1 v14) (object2 v15) (rel (m24 (lex first)))))
 (cq
  (p28 (min 0) (max 0)
   (arg (p27 (object1 v16) (object2 v15) (rel (m24)))))))

(m30!)

 CPU time : 0.05

*
;-------------------------------------------------------
; BG XIII. Defining "other"
;-------------------------------------------------------
(describe (add forall ($x $y $z)
&ant (build member *x class *y)
&ant (build member *z class (build mod (build lex other) head *y))
cq (build min 0 max 0 arg(build equiv *x equiv *z))
          )
)
```

```
(m32! (forall v19 v18 v17)
 (&ant
  (p31 (class (p30 (head v18) (mod (m31 (lex other))))) (member v19))
  (p29 (class v18) (member v17)))
 (cq (p33 (min 0) (max 0) (arg (p32 (equiv v19 v17))))))
(m15! (class (m14 (head (m3 (lex bird))) (mod (m13 (lex migratory)))))
 (member b1))
(m11! (class (m7 (lex season))) (member b3))
(m8! (class (m7)) (member b2))
(m4! (class (m3)) (member b1))
(m2! (class (m1 (lex AmericanRobin))) (member b1))

(m32! m15! m11! m8! m4! m2!)

 CPU time : 0.07

*

; CASSIE READS THE PASSAGE:
; =========================
;-----------------------------------------------------------
; I. THE AMERICAN ROBIN IS CALLED THE "HARBINGER OF SPRING"
;-----------------------------------------------------------

(describe (add object *AmericanRobin
                  rel (build lex harbinger) = HARBINGERS
                  possessor *Spring) = ARCHOS)

(m36! (class (m34 (lex harbinger))) (member b1))
(m35! (object b1) (possessor b2) (rel (m34)))

(m36! m35!)

 CPU time : 0.01

*

;-----------------------------------------------
; II. BECAUSE OF ITS EARLY NORTHWARD MIGRATION
;-----------------------------------------------
(describe (add cause (add agent *AmericanRobin
                                act (build action *MIGRATES) = MIGRATION
                      ) = ENM
                  effect *ARCHOS
          )
)

(m38!
 (cause (m37! (act (m19 (action (m18 (lex migrate))))) (agent b1)))
 (effect (m36! (class (m34 (lex harbinger))) (member b1))
  (m35! (object b1) (possessor b2) (rel (m34)))))

(m38!)

 CPU time : 0.01

*
(describe (add object *MIGRATION property (build lex early)))

(m40! (object (m19 (action (m18 (lex migrate)))))
 (property (m39 (lex early))))
```

```
(m40!)

 CPU time : 0.01

* (describe (add action *MIGRATES from #STARTLOC to #ENDLOC))

(m41! (action (m18 (lex migrate))) (from b4) (to b5))

(m41!)

 CPU time : 0.00

* (describe (add object1 *ENDLOC rel (build lex north) object2 *STARTLOC))

(m43! (object1 b5) (object2 b4) (rel (m42 (lex north))))

(m43!)

 CPU time : 0.01

*
;---------------------------------------------------------------
; III. WHICH BRINGS ITS ARRIVAL BEFORE MOST OTHER MIGRATORY BIRDS
;---------------------------------------------------------------
(describe (add before #ARRIVETIME1 after #ARRIVETIME2))

(m44! (after b7) (before b6))

(m44!)

 CPU time : 0.01

* (describe (add cause *ENM
                  effect (add agent *AmericanRobin
                               act (build action *ARRIVAL)
                               time *ARRIVETIME1
                         )
          )
)

(m47!
 (cause (m37! (act (m19 (action (m18 (lex migrate))))) (agent b1)))
 (effect (m46! (act (m26 (action (m23 (lex arrive))))) (agent b1))
  (m45! (act (m26)) (agent b1) (time b6))))

(m47!)

 CPU time : 0.01

*
(describe (add agent (build mod (build lex other)
                                head *MIGRATORYBIRDS
                     )

               act (build action *ARRIVAL)
               time *ARRIVETIME2
          )
)

(m50! (act (m26 (action (m23 (lex arrive))))))
```

```
 (agent
  (m48 (head (m14 (head (m3 (lex bird))) (mod (m13 (lex migratory)))))
   (mod (m31 (lex other))))))
(m49! (act (m26)) (agent (m48)) (time b7))

(m50! m49!)

 CPU time : 0.01

*

;-------------------------------------------------------------------------
; IV. IT IS USUALLY THE FIRST OF THE SUMMER BIRDS TO BE NOTED BY HUMANS
;-------------------------------------------------------------------------
(describe (add object1 *AmericanRobin
                  rel *FIRSTOF
                  object2 #SummerBirdsNotedByHumans
          )
)

(m54! (class b8) (member b1))
(m53! (object1 b1)
 (object2 (m52 (act (m26 (action (m23 (lex arrive))))) (agent b8)))
 (rel (m25 (lex indicates))))
(m51! (object1 b1) (object2 b8) (rel (m24 (lex first))))

(m54! m53! m51!)

 CPU time : 0.01

* (describe (add subclass *SummerBirdsNotedByHumans
                  superclass (build mod *Summer head *BIRDCLASS)
          )
)

(m56! (subclass b8) (superclass (m55 (head (m3 (lex bird))) (mod b3))))

(m56!)

 CPU time : 0.00

* (describe (add subclass *SummerBirdsNotedByHumans
                  superclass (build mod (build lex notedBy) head (build lex
humans))          )
)
(m60! (subclass b8)
 (superclass (m59 (head (m58 (lex humans))) (mod (m57 (lex notedBy))))))

(m60!)

 CPU time : 0.01

*


; Ask Cassie what "harbinger" means:
^(
--> defineNoun "harbinger")
 Definition of harbinger:
 Possible Class Inclusions: AmericanRobin, bird, m14, b8,
 Possible Actions: arrive, migrate,
```

```
 Possible Properties: indicates m52, first b8,
 Possessive: season,
nil



 CPU time : 0.61

*

End of /home/csgrad/ag33/cse663/harbinger.demo demonstration.


 CPU time : 1.20

* (lisp)
"End of SNePS"
cl-user(3): (exit)
; Exiting Lisp
pollux {~/cse663} > exit

exit

script done on Tue Dec 09 23:03:35 2003
```

Appendix C: SNePS Network Diagrams

Note that the following diagrams represent only the background rules and the passage. Items which CASSIE infers from the passage are not included

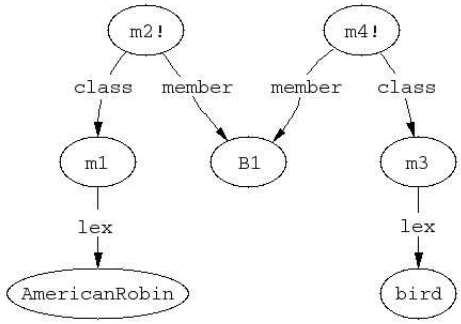

Figure 2: "The American Robin is a bird"

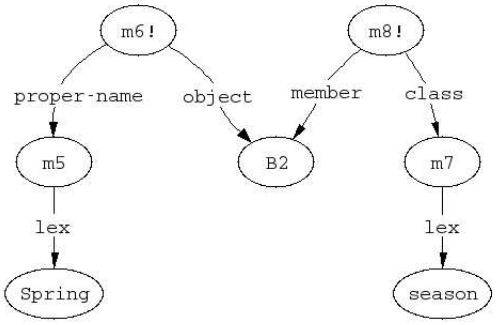

Figure 3: "Spring is a season"

here.

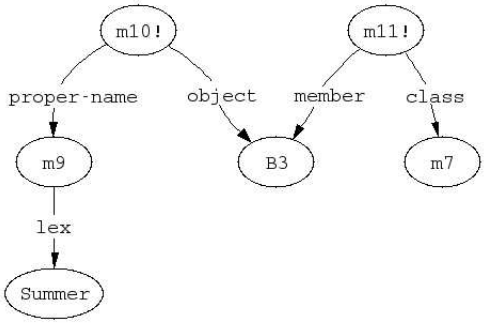

Figure 4: "Summer is a season"

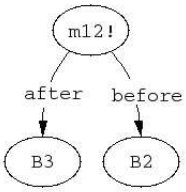

Figure 5: "Spring is before Summer (or Summer is after Spring)"

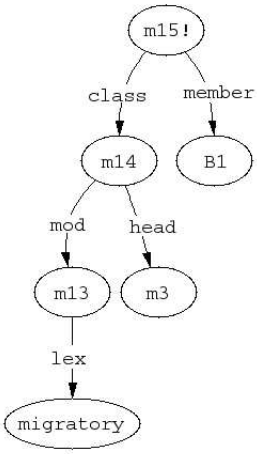

Figure 6: "The American Robin is a migratory bird"

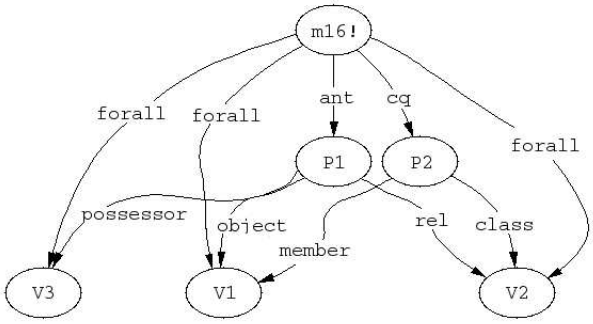

Figure 7: "If x is z's y then x is a y"

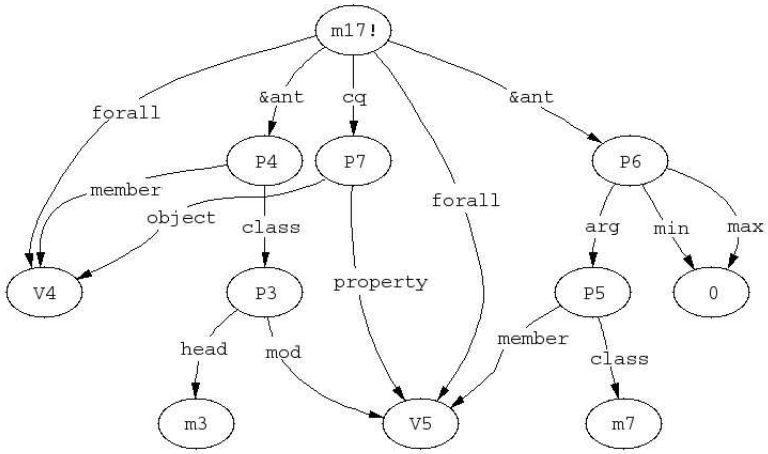

Figure 8: "If x is a y bird and y is not a season then x is a y"

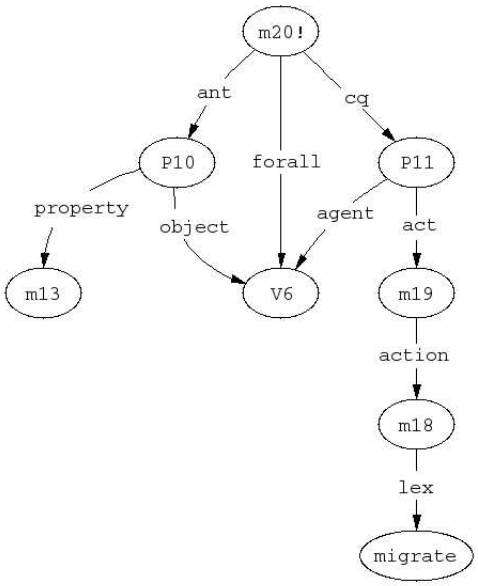

Figure 9: "If x is migratory then x migrates"

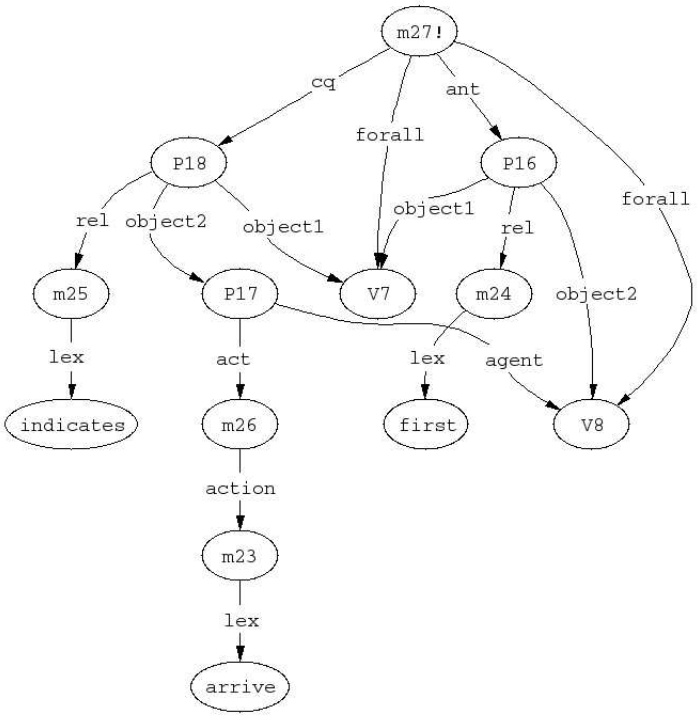

Figure 10: "If x is the first of y then x is an indicator that y has arrived"

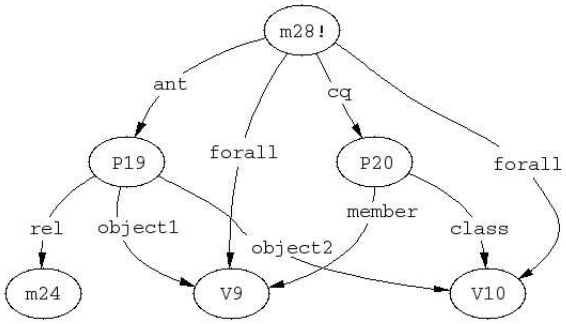

Figure 11: "If x is the first of y then x is a y"

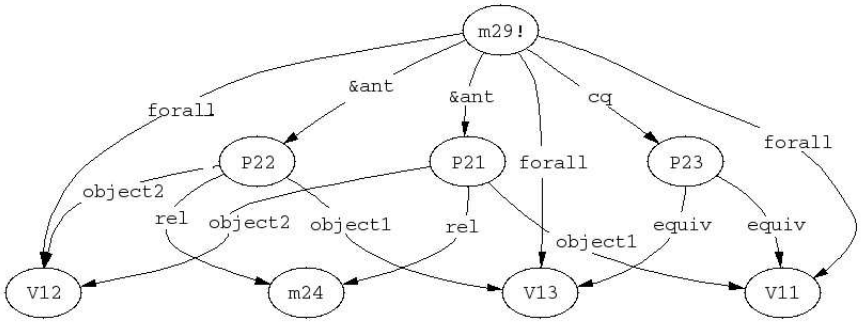

Figure 12: "If x is the first of y and z is the first of y then x and z are equivalent"

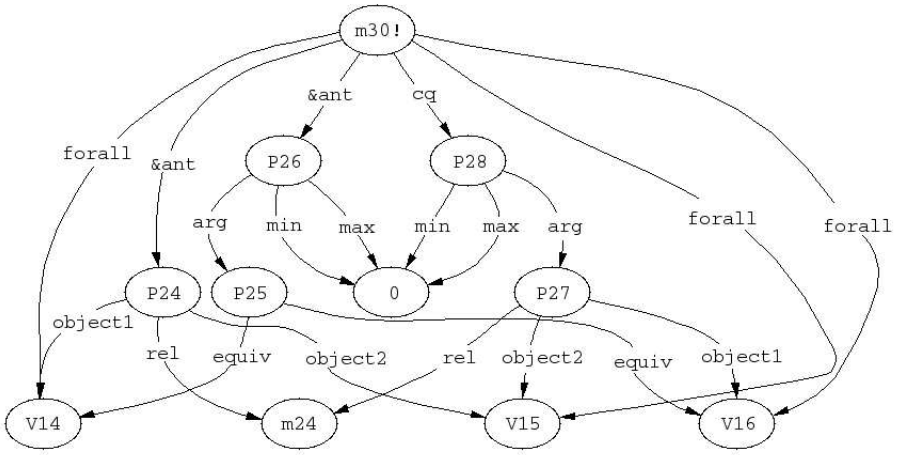

Figure 13: "If x is the first of y and x is not equivalent to z then z is not the first of y"

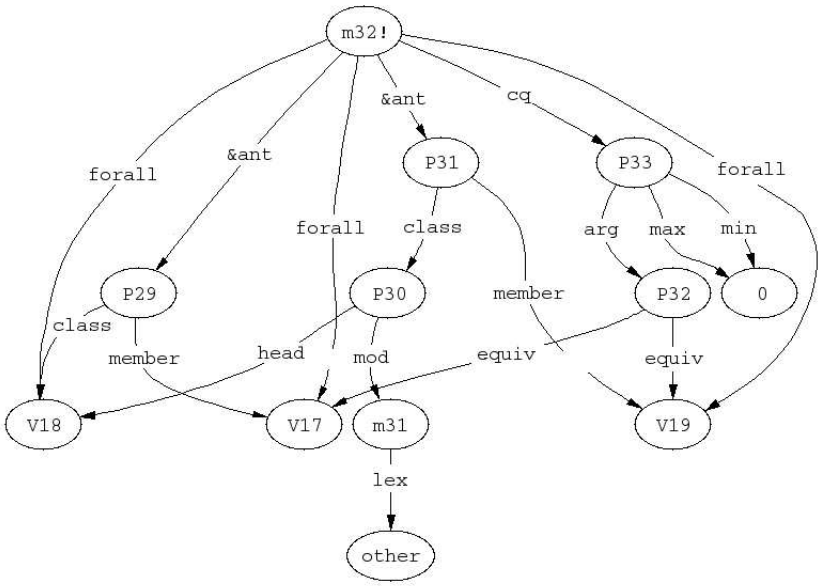

Figure 14: "If x is a y and z is another y then x is not equivalent to z"

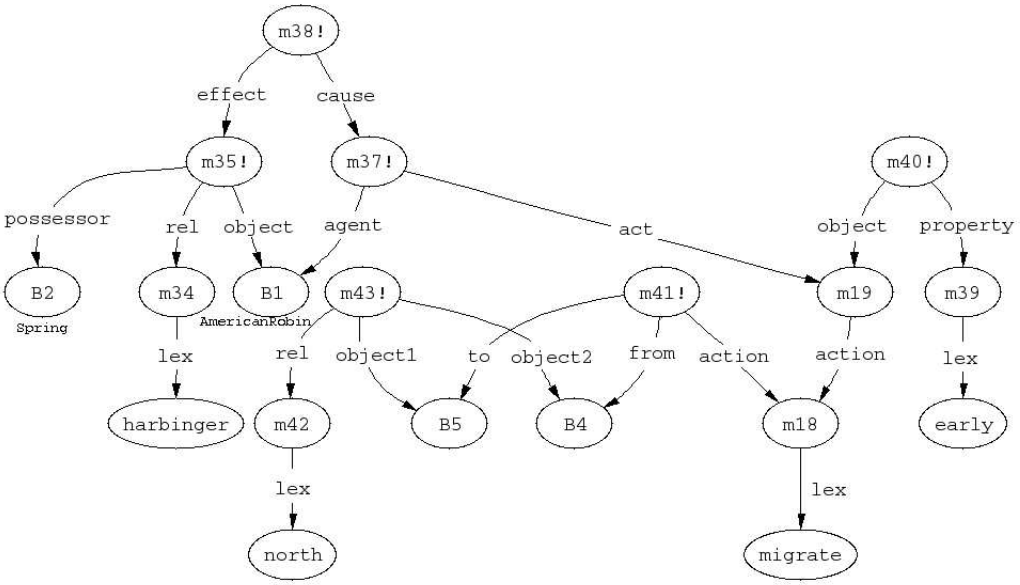

Figure 15: "(I) The American Robin is called the harbinger of Spring (II) because of its early northward migration. . . "

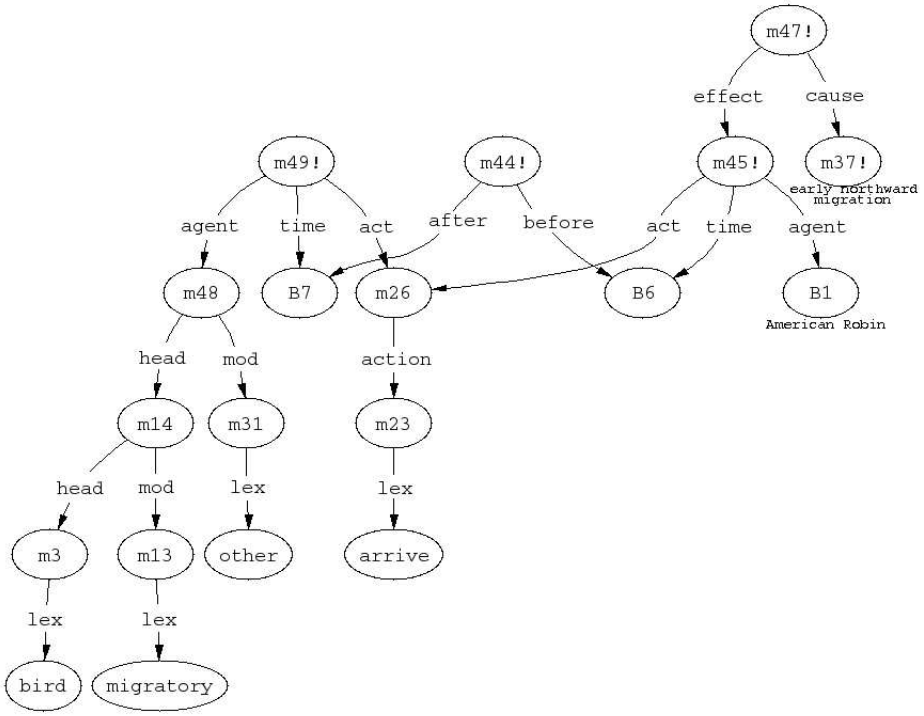

Figure 16: "(III)... which brings its arrival before most other migratory birds..."

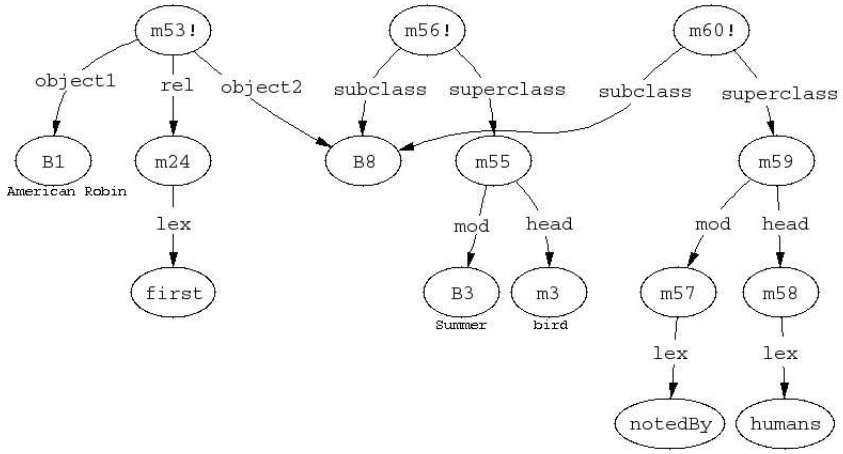

Figure 17: "(IV)... it is usually the first of the summer birds to be noted by humans."

References

1. Napieralski, Scott (2003), "Noun Algorithm Case Frames", [http://www.cse.buffalo.edu/ stn2/cva/case-frames/']

2. Quillian, M. Ross (1967), "Word Concepts: A Theory and Simulation of Some Basic Semantic Capabilities", in Ronald J. Brachman & Hector J. Levesque (eds.), *Readings in Knowledge Representation* (Los Altos, CA: Morgan Kaufmann, 1985): p.98

3. Rapaport, William J., & Ehrlich, Karen (2000), "A Computational Theory of Vocabulary Acquisition", in Lucja M. Iwanska & Stuart C. Shapiro (eds.), *Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language* (Menlo Park, CA/Cambridge, MA: AAAI Press/MIT Press): 5.

4. Shapiro, Stuart C., & Rapaport, William J. (1996), "A Dictionary of SNePS Case Frames", retrieved from [http://www.cse.buffalo.edu/sneps/Manuals/dictionary.pdf].