Contextual Vocabulary Acquisition

“Kolper”

Timaporn Chotvinijchai

April 30, 2004

CSE 740: Seminar: Contextual Vocabulary Acquisition

Abstract

This paper is about work in Contextual Vocabulary Acquisition project in my seminar. The project is to get the meaning of unknown word in the passage. The passage used here is “When you are used to a broad view, it is quite depressing when you come to live in a room with one or two **kolpers** fronting on a courtyard”. The unknown word here is “kolper”. It is a Dutch-sounding word. In fact, Kolper is a window that transmits little light because of something in front of it. This kind of window is quiet common in the Netherlands. The goal meaning for kolper in this sentence is “window”. This work accomplished the goal. The immediate step for this project is to add temporal representation and improve the exist presentation. For future work, the definition for this sentence should be used for the next sentences in the series.

Introduction

Contextual Vocabulary Acquisition research project is to find algorithm for getting the definition of unknown word in the passage from information in the context. It is implemented by using SNePS as the knowledge base and inference engine. After testing successfully in SNePS, they will be taught to grade school students. The feedback from students will be collect for using in developing the further algorithm. Consequently, the benefits from this project are: to improve the language processing systems and to help in the education of vocabulary.

My task here was to select a target word and a context to work on, analyze the context, create a SNePS representation for that context, use the noun algorithm for extracting the meaning of that word, and improve the representation based on the result from running the algorithm.

Target word and context

My target word for this project is “kolper”. The selected passage is “When you are used to a broad view, it is quite depressing when you come to live in a room with one or two **kolpers** fronting on a courtyard”. This sentence is the first sentence in the series. The series has five sentences. Each sentence use kolper in different ways. For example, the first three sentence give the brief concept and example of kolper while the last two sentences give the counter-example.

Kolper is a Dutch-sounding word. In fact, it is a window that transmits little light because of something in front of it. This kind of window is quiet common in the Netherlands. The target definition of kolper in this sentence is “window”.

Analysis of the context

To analyze the sentence, I did some human surveys. The subject for this survey is native English speakers. The purpose is to ask what and why they think about kolper.

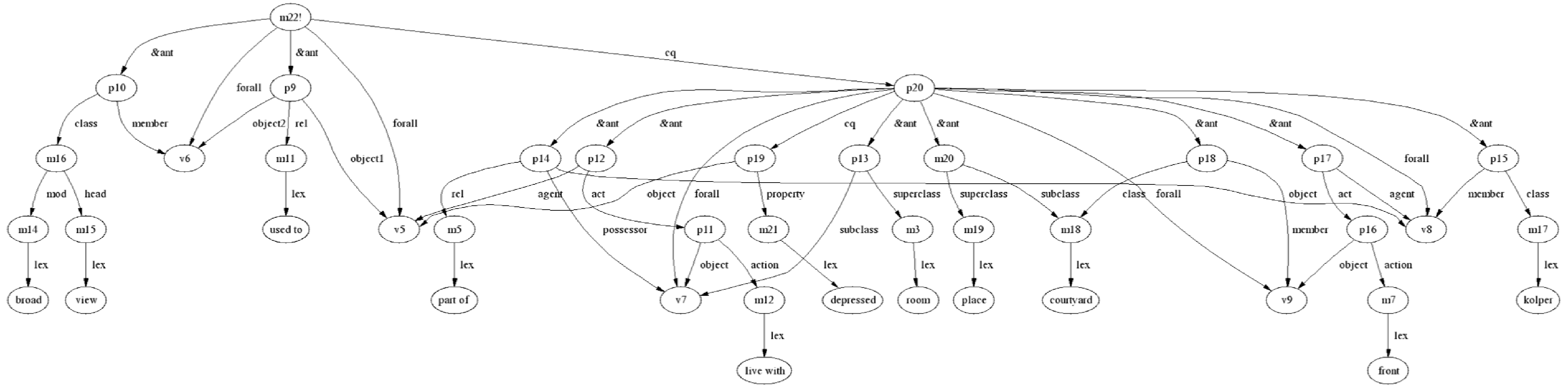
First, the subjects will be asked to read the sentence aloud. Then, they will answer what they think kolper will be. Moreover, they have to explain why they think like that.

The results from all subjects are similar to each other. All people think that kolper is a window. They also use the similar reason to explain why. First, the reason is the sentence discuss about broad view. Then, kolper is part of a room and also fronting a courtyard. They said when kolper front a courtyard, kolper must make view of that courtyard. Therefore, the part of a room that makes a view is window. In summary, they conclude that kolper is a window.

From this survey, I use the answer from the subjects to build the background knowledge used in the project. This knowledge will give the system enough information to get the definition of unknown word in the sentence. There is some important knowledge. First, window makes a view. Second, window is part of a room. Finally, when window front something, it will make view of that things.

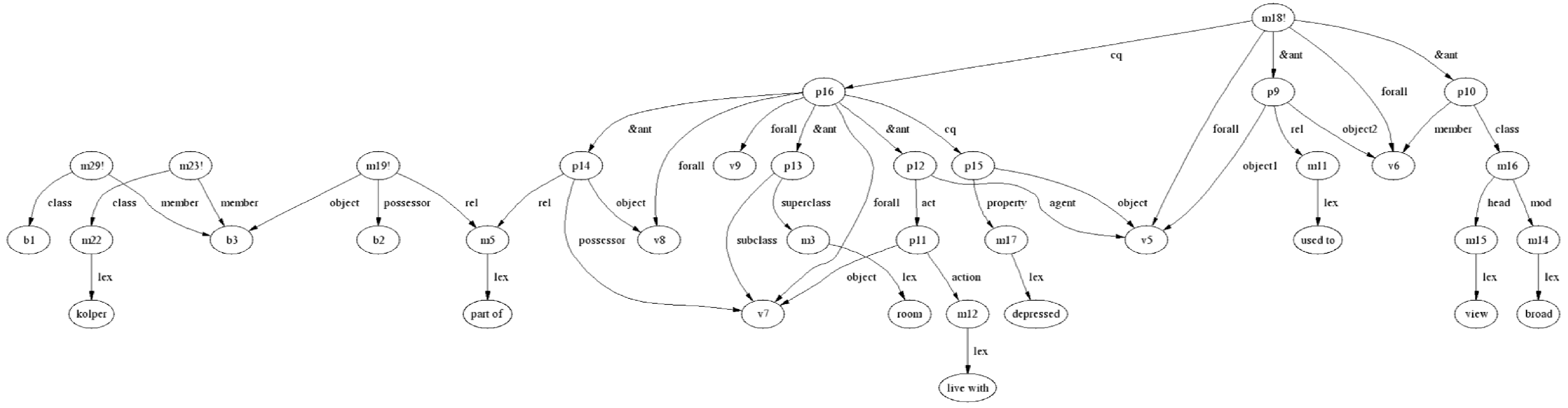
Representing the context

I use SNePS (Semantic Network Processing System) to represent the background knowledge and the sentence. The sentence has pattern "When P, then: when Q then R". At first, I use one rule-based case frame to represent this sentence. The simplify sentence is "If you used to V and V is broad view, then if you live with W, W is subclass of room, kolper is part of W, kolper front courtyard then you are depressed". The following is the SNePS diagram that represents the simplified version of the sentence.



“If you used to V and V is broad view, then if you live with W, W is subclass of room, kolper is part of W, kolper front courtyard then you are depressed”

Anyway, when I used this representation, there was a problem. The system did not infer to any background knowledge or rules. The result will be nil. So I change the representation from one rule-based sentence to one rule-based sentence and few normal sentences. The modified rule-based is “If you used to V and V is broad view, then if you live with W, W is subclass of room, X is part of W then you are depressed”. The normal sentences are “Kolper is part of a room”, “Kolper front a courtyard”, and “Courtyard is a place”. When I used the latter presentation, the system can infer that kolper is a “window” which has possible action “make view courtyard” and “front courtyard”. It also infers that kolper is part of a room.



“If you used to V and V is broad view, then if you live with W, W is subclass of room, X is part of W then you are depressed”,

“Kolper is part of a room”,

“Kolper front a courtyard”,

“Courtyard is a place"

Immediate Next Steps to Take

If I have more time, I will put the temporal representation to represent the phrase "for long time" in the rule "if people used to something, then people live with that thing for long time". The case frame that will be used in this case should be

```
(build stime #time1 etime #time2 duration #dur)
```

The meaning is "for start time #time1 and end time #time2, the duration is #dur". I will substitute "for long time" for #dur.

Another thing is to improve the exist representation to be more clear and better represent the sentence.

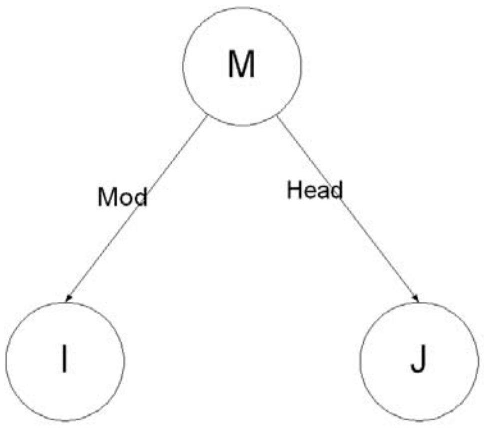
Future Work

As mention above, the sentence used in this project is the first sentence in the series. So the future work for this project is to use definition or information of kolper found in this sentence as the background knowledge for the next sentence. The next sentence will use the definition that kolper is a window and add more information from context in that sentence. Finally, system will get more information when more sentences are presented.

APPENDIX

i. New SNePS case frames created

In addition to the standard SNePS case frames, I defined one more case frame in the context. The syntax and semantics are described below

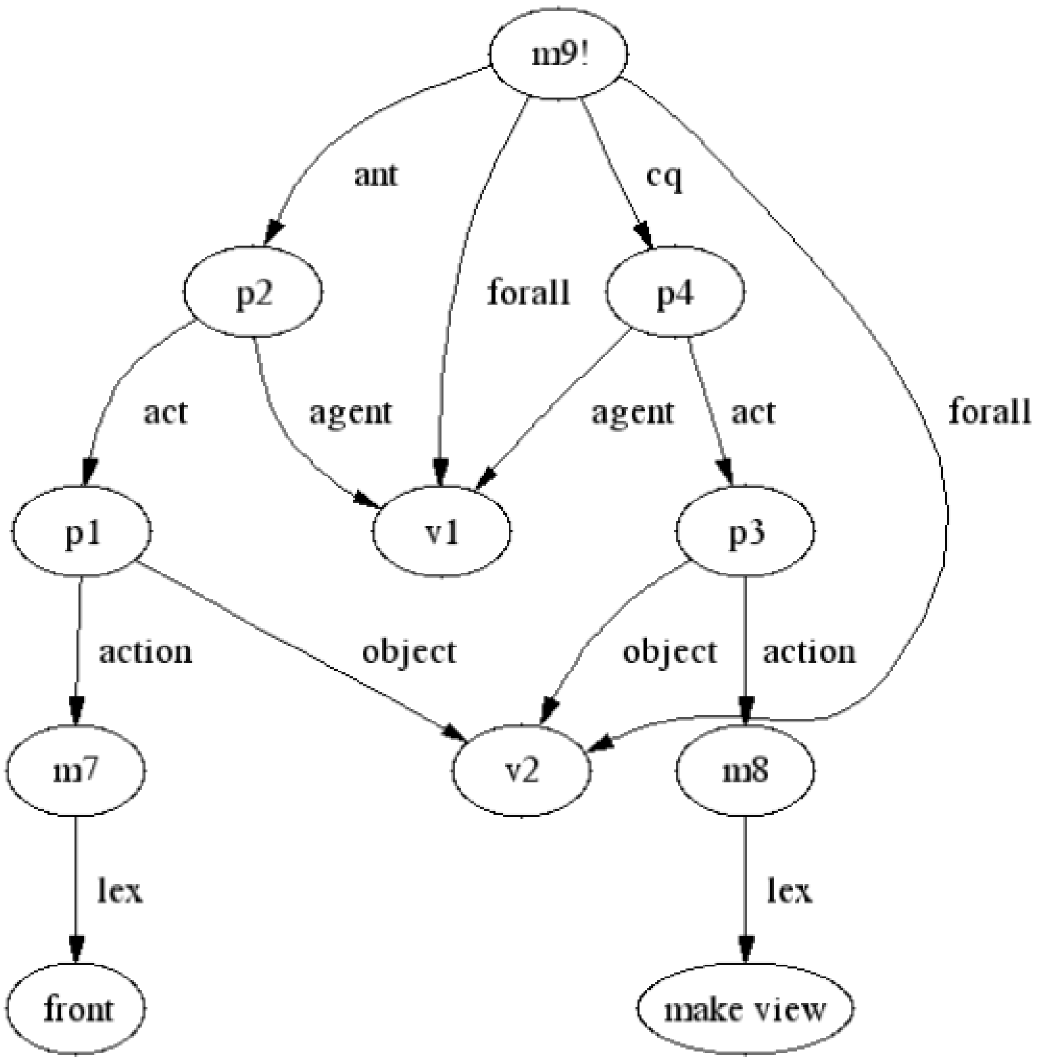


Syntax : [[(build mod i head j)]]

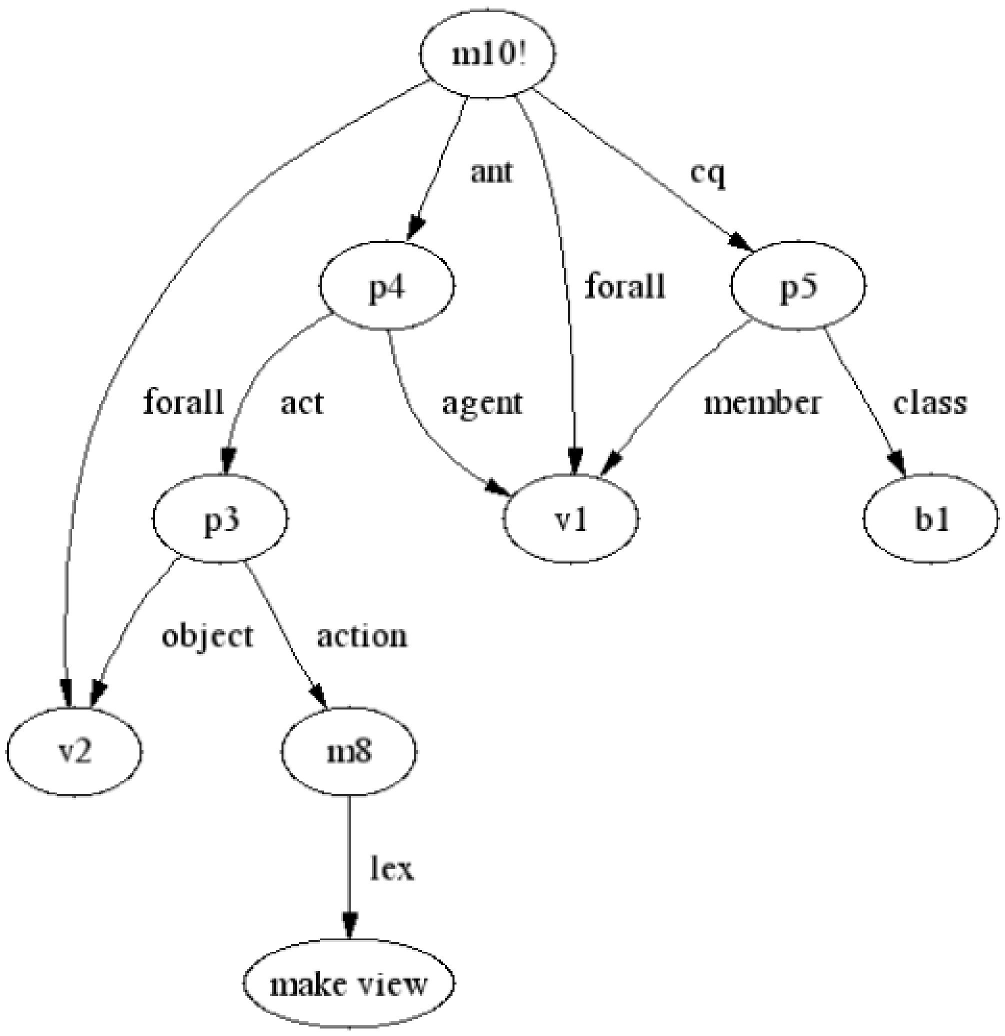
Semantics : [[m]] is the proposition such that an individual [[J]] modified by [[I]]

ii. SNePS diagrams

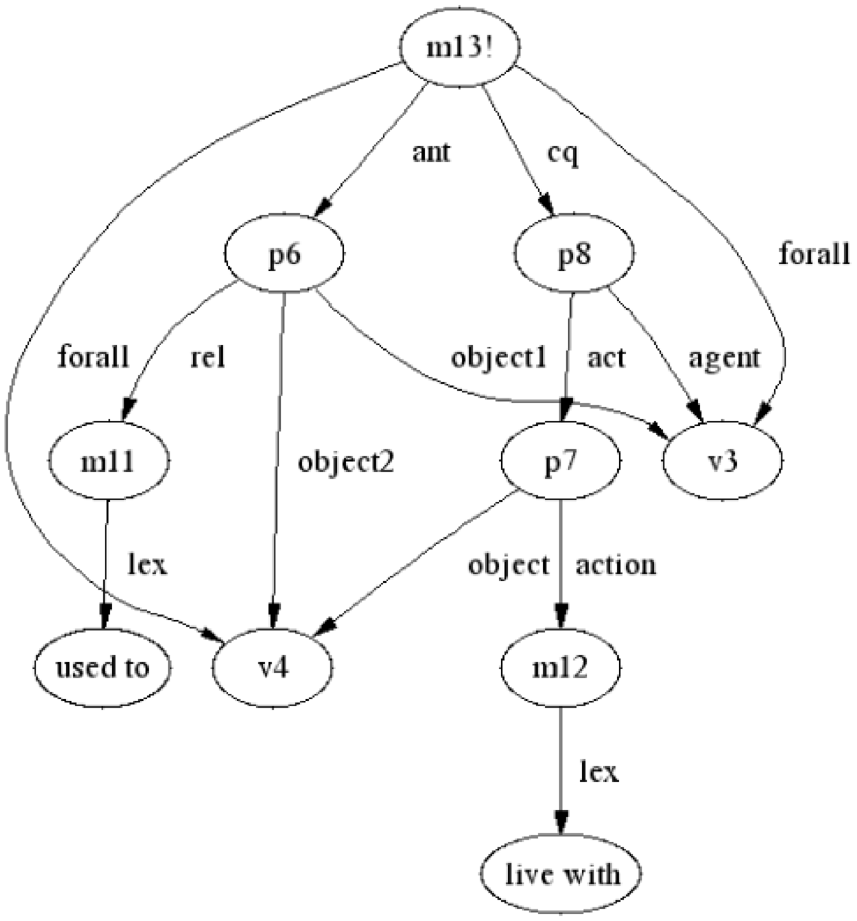
Here are some diagrams from the SNePS representation for background knowledge. The diagram of sentence are represented above



This diagram represents the rule “if part of room front some place, then it make view of that place”. Part of room is represented by v1 and place is represented by v2. I used case frame agent/act/action/object for this rule. Agent is v1 or “part of room”. Action is “front” and “make view”. Finally, object is v4 or “place”.



This diagram represents "if part of room make view of place, then it is window". This rule is extended from the previous so the variables represent the same thing. That is part of room is represented by v1 and place is represented by v2. I used case frame agent/act/action/object for the first part in the rule “part of room make view of place”. Agent is v1 or “part of room”. Action is “make view”. Object is v4 or “place”.
Then, I used case frame member/class to represent “it is window”. Member is v1 or “part of room” and class is b1 that is a window.



This diagram represents "if people used to something then people live with that thing for long time". This rule represents different things so the variable here is different from above. Here, v3 represents people and v4 represents thing. As I mention in immediate step that if I have time I will put the temporal representation for this rule. Now I just ignore the phrase “for long time” in this diagram. The case frame used for the first part in this diagram is object1/rel/object2 for “people used to something”. Object1 is people, object2 is thing, and rel is “used to”. The case frame used in the second part is agent/ac/action/object. Agent is v3 or “people”; action is “live with”; and object is v4 or “thing”.

iii. Annotated Demo

```
Script started on Fri Apr 30 09:39:02 2004
csedb: No such file or directory.
pollux {~/CSE740} > composer

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
[2] cl-user(3): :ld /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.
[2] cl-user(4): (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.

   4/30/2004 9:39:44

* (demo "kolper.demo" :av)

File /home/csgrad/tc35/CSE740/kolper.demo is now the source of input.

  The demo will pause between commands, at that time press
  RETURN to continue, or ? to see a list of available commands


 CPU time : 0.01

* ;
=======================================================================
; FILENAME:     kolper.demo
; DATE:         April 30, 2004
; PROGRAMMER:   Timaporn Chotvinijchai

;; this template version:       template.demo.2003.11.17.txt

; 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 "kolper.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.
^(
--- pause ---

--> setq snip:*infertrace* nil)
--- pause ---
nil



 CPU time : 0.00

*
; Load the appropriate definition algorithm:
;; UNCOMMENT THE ONE YOU *DO* WANT
;; AND DELETE THE OTHER!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 ^(
--- pause ---

--> load "/projects/rapaport/CVA/STN2/defun_noun.cl")
--- pause ---
;   Loading /projects/rapaport/CVA/STN2/defun_noun.cl
t



 CPU time : 0.19

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

; load show utility
^
--- pause ---

--> (load "show")
--- pause ---
;   Loading /home/csgrad/tc35/CSE740/show.lisp
t



 CPU time : 0.00

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

```
--- pause ---

Net reset - Relations and paths are still defined


 CPU time : 0.01

*
; OPTIONAL:
; UNCOMMENT THE FOLLOWING CODE TO TURN FULL FORWARD INFERENCING ON:
;
; ;enter the "snip" package:
 ^(
--- pause ---

--> in-package snip)
--- pause ---
#<The snip package>



 CPU time : 0.00

* ;
; ;turn on full forward inferencing:
 ^(
--- pause ---

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



 CPU time : 0.00

*
; ;re-enter the "sneps" package:
 ^(
--- pause ---

--> in-package sneps)
--- pause ---
#<The sneps package>



 CPU time : 0.00

*
```

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


 CPU time : 0.00

*

(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)

 CPU time : 0.05

*

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


 CPU time : 0.05

* (define mod head)
--- pause ---

(mod head)

 CPU time : 0.00

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


 CPU time : 0.01

*

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

;; #window is a window
(describe (assert member #window class (build lex "window")))
--- pause ---

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

(m2!)

 CPU time : 0.01

*
;; #room is a room
(describe (assert member #room class (build lex "room")))
--- pause ---

(m4! (class (m3 (lex room))) (member b2))

(m4!)
```

```
 CPU time : 0.00

*
;; window is part of a room
(describe (assert object *window
                  rel (build lex "part of")
                  possessor *room))
--- pause ---

(m6! (object b1) (possessor b2) (rel (m5 (lex part of))))

(m6!)

 CPU time : 0.00

*
;; if part of room  front some place, then it make view of that place
(describe (assert forall ($por $place)
              ant (build agent *por
                         act (build action (build lex "front")
                                    object *place))
              cq  (build agent *por
                         act (build action (build lex "make view")
                                    object *place))))
--- pause ---

(m9! (forall v2 v1)
 (ant (p2 (act (p1 (action (m7 (lex front))) (object v2))) (agent v1)))
 (cq
  (p4 (act (p3 (action (m8 (lex make view))) (object v2))) (agent
v1))))

(m9!)

 CPU time : 0.00

*

;; if part of room make view of place, then it is window
(describe (assert forall (*por *place)
                ant (build agent *por
                          act (build action (build lex "make view")
                                     object *place))
                cq  (build member *por class *window)))
--- pause ---

(m10! (forall v2 v1)
 (ant
  (p4 (act (p3 (action (m8 (lex make view))) (object v2))) (agent v1)))
 (cq (p5 (class b1) (member v1))))

(m10!)

 CPU time : 0.00

*
```

```
;; if people "used to something" then peolple  "live with that thing
for long time"
;; here is the rule that has to put temporal representation for
representing "for long time"
;; the case frame that will be used here should be "build stime #time1
etime #time2 dur #dur"

(describe (assert forall ($people $thing)
                  ant (build object1 *people
                             rel (build lex "used to")
                             object2 *thing)
                  cq (build  agent *people
                             act (build action (build lex "live with")
                                         object *thing)

)
))
--- pause ---

(m13! (forall v4 v3)
 (ant (p6 (object1 v3) (object2 v4) (rel (m11 (lex used to)))))
 (cq
  (p8 (act (p7 (action (m12 (lex live with))) (object v4)))
   (agent v3))))

(m13!)

 CPU time : 0.00

*



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

;; the sentence is
;; "when you are used to a broad view, it is quiet depressing when you
come to live in a room with one or two
;; kolpers fronting on a courtyard"

;; the comment in the rule was the first version that represent the
sentence directly but it
;; is not infer to any background knowledge. If remove the comment, and
then run the noun algorithm
;; the result will be nil

(describe (add forall ($u $v)
           &ant (build object1 *u
                       rel (build lex "used to")
                       object2 *v)
           &ant (build member *v class (build mod (build lex "broad")
head (build lex "view")))
            cq (build   forall ($w $x $y)
                        &ant (build agent *u
                            act (build action (build lex "live with")
                                              object *w))
                        &ant (build subclass *w superclass (build lex
"room"))
                        &ant (build object *x rel (build lex "part of")
possessor *w)
                ;       &ant (build member *x class (build lex
"kolper"))
                ;       &ant (build agent *x
                 ;                   act (build action (build lex
"front")
                  ;                              object *y))
        ;               &ant (build member *y class (build lex
"courtyard"))
         ;               &ant (build subclass (build lex "courtyard")
superclass (build lex "place"))
                        cq (build object *u
                                  property (build lex "depressed"))

           )
))
--- pause ---

(m18! (forall v6 v5)
 (&ant
  (p10 (class (m16 (head (m15 (lex view))) (mod (m14 (lex broad)))))
   (member v6))
  (p9 (object1 v5) (object2 v6) (rel (m11 (lex used to)))))
 (cq
  (p16 (forall v9 v8 v7)
   (&ant (p14 (object v8) (possessor v7) (rel (m5 (lex part of))))
    (p13 (subclass v7) (superclass (m3 (lex room))))
    (p12 (act (p11 (action (m12 (lex live with))) (object v7)))
     (agent v5)))
   (cq (p15 (object v5) (property (m17 (lex depressed))))))))

(m18!)

 CPU time : 0.04

*

;; this representation are used instead of the comment rule above
;;

;; kolper is part of room
(describe (add object #kolper
                rel (build lex "part of")
                possessor *room))
--- pause ---

(m19! (object b3) (possessor b2) (rel (m5 (lex part of))))

(m19!)

 CPU time : 0.01

*
;; kolper is a kolper
```

```
(describe (add member *kolper class (build lex "kolper")))
--- pause ---

(m23! (class (m22 (lex kolper))) (member b3))

(m23!)

 CPU time : 0.01

*
;; kolper front courtyard
(describe (add agent *kolper
                act (build action (build lex "front")
                           object (build lex "courtyard"))))
--- pause ---

(m29! (class b1) (member b3))
(m28!
 (act (m27 (action (m8 (lex make view)))
       (object (m24 (lex courtyard)))))
 (agent b3))
(m26! (act (m25 (action (m7 (lex front))) (object (m24)))) (agent b3))

(m29! m28! m26!)

 CPU time : 0.01

*
;; courtyard is subclass of place
(describe (add  subclass (build lex "courtyard")
                superclass (build lex "place")))
--- pause ---

(m31! (subclass (m24 (lex courtyard))) (superclass (m30 (lex place))))

(m31!)

 CPU time : 0.01

*

; Ask Cassie what "WORD" means:
;; UNCOMMENT THE ONE YOU *DO* WANT
;; AND DELETE THE OTHER!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

; it should show that kolper also make a view of courtyard from the
rule
; also kolper is part of kroom or kolper is part of room from the
inference

 ^(
--- pause ---

--> defineNoun "kolper")
--- pause ---
 Definition of kolper:
 Possible Class Inclusions: window,
 Possible Actions: make view courtyard, front courtyard,
 Possessive: room part of,
nil



 CPU time : 0.09

* ; ^(defineVerb "WORD")


; it should show that kolper is part of room too ..
; maybe because of the definition of window & room ...

End of /home/csgrad/tc35/CSE740/kolper.demo demonstration.


 CPU time : 0.53

*
(lisp)
"End of SNePS"
[2] cl-user(5): :exit
; Exiting Lisp
pollux {~/CSE740} > exit

exit
```

iv. Online Resources

All work I did on the project can be found at:

http://www.cse.buffalo.edu/~tc35/cva/

- kolper.demo : demo file
- prj_demo : output from running the demo file
- cvareport.doc : this report
- rule.doc : the first version of sentence representation diagram
- mod_rule.doc : the final version of sentence representation diagram
- cvareport.pdf : final version of this report that include diagram in rule.doc and mod_rule.doc

v. References:

- Rapaport, William J., Kibby, Michael W. (2001) “Contextual Vocabulary Acquisition: Development of a Computational Theory and Educational Curriculum”.

- Van Daalen-Kapteijns, M.M., & Elshout-Mohr, M. (1981), "The Acquisition of Word Meanings as a Cognitive Learning Process", *Journal of Verbal Learning and Verbal Behavior* 20: 386-399.