Deriving dossiers from context

Pejus Das
(pejusdas@cse.buffalo.edu)

CSE 740
December 16, 2004

Abstract

Have you ever come across a word in a passage you don't have a clue about? We often do. Well, we try to look up the dictionary for its meaning. But at times this is not only frustrating but time consuming; one word definition references another making it difficult to understand the passage. Most of the times a dictionary can be avoided. Wondering how? Use CVA or Contextual Vocabulary Acquisition. This report demonstrates an example how CVA was used to infer the meaning of an unknown word from context using SNePS, the knowledge representing/reasoning system. This report describes the project that was worked on in the CVA seminar offered by Dr. Rapaport at University of Buffalo, in fall 2004.

The CVA Project

The CVA research project is conducted by Prof. William J. Rapaport (Department of Computer Science and Engineering, and Center for Cognitive Science) and Prof. Michael W. Kibby (Department of Learning and Instruction, and Center for Literacy and Reading Instruction). The project aims to develop a computational theory to derive unknown word meanings from context and use the experience thus gained to develop a curriculum for students to make them aware of CVA techniques. The current algorithms in CVA are implemented using SNePS, a knowledge representation/reasoning system and Cassie its computerized cognitive agent.

For more information on the CVA project kindly refer to the CVA homepage at http://www.cse.buffalo.edu/~rapaport/cva.html

SNePS and Cassie

SNePS stands for Semantic Network Processing System. It is a propositional semantic network build by Dr. Stuart Shapiro, (Department of Computer Science, SUNY at Buffalo) and his research group. SNePS can be used for representing and reasoning in natural language. A SNePS network consists of nodes and arcs. Each node is a proposition which is interconnected by arcs. Nodes represent concepts, and arcs represent non-conceptual relations between concepts providing structure to the SNePS network. Cassie is SNePS' computerized cognitive agent. In our project, after we represent the passage and other rules of inference in SNePS, we will ask Cassie what the unknown word means. Following is an example of representing the sentence "Elephants are gray." in SNePS,

SNePSUL code:

(show (assert object (build lex elephant) property grey))

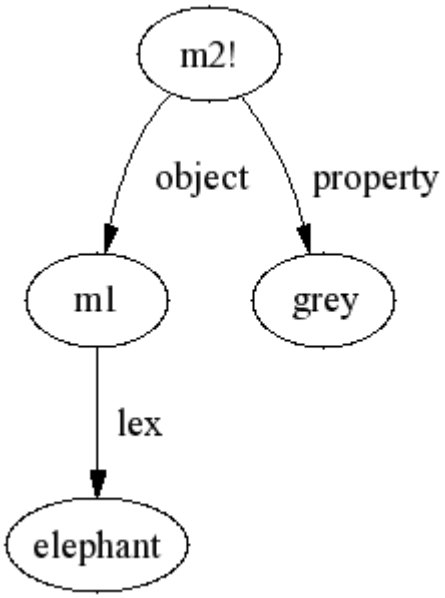

Figure 1: SNePS Network for the sentence “Elephants are gray”

For more information on SNePS, Kindly refer to the SNePS homepage at http://www.cse.buffalo.edu/sneps/.

The Unknown Word

The passage selected for the purpose of the project was, "The Archives of the medical department of Lourdes are filled with dossiers that detail well-authenticated cases of what are termed miraculous healings." This passage was taken from the book "The Power of Your Subconscious Mind" by Dr. Joseph Murphy. Guess what the unknown word is….. dossiers.

Protocols

Subjects were approached and given the passage and were asked to infer the meaning of the unknown word. The word dossier was replaced by a different word, "smizers" (I wonder if that word exists in the Dictionary). Subjects read the passage and were asked to think aloud while they arrive at the meaning of the unknown word. This was documented and used while representing the passage in SNePS. After reading the passage subjects came up with answers like "smizers are documents", "smizers provide details about medical healing", "they are objects in medical departments". Going by what the subjects inferred from the passage, it (the passage) was represented in SNePS in such a way that Cassie would infer the meaning of dossiers as documents, with possible actions of providing more information about well authenticated cases and as things that belong to a medical departments. Kindly refer Appendix A for transcripts of the protocol.

Approach

The sentence was divided into smaller parts. These smaller parts, of the sentence, were represented in SNePS. Based on the verbal protocols some background knowledge and rules of inference were also represented as Cassie's knowledge base. This was used by Cassie to come up with a definition of the unknown word. The Collins Cobuild English Dictionary was used to look up the meaning of other possible words in the passage. For example, what does it mean to be filled with? The dictionary defines to be filled with as to contain. This was used to build up the background knowledge required for Cassie.

Case Frames

For building the network the CVA standard case frames for nouns at http://www.cse.buffalo.edu/~rapaport/CVA/CaseFrames/case-frames/ were used.

Background Knowledge and Inference rules

Following background knowledge and rules of inference were included in the representation.

1. The following is an inference rule. It asserts that if x is filled with y then x contains y. This rule is used by Cassie to infer that Archives contain documents from part of the passage "Archives are filled with dossiers".

```
(describe (assert forall ($x $y)
        ant(build object1 *x rel (build lex filled-with) = filled-with-concept object2 *y)
        cq (build object1 *x rel (build lex contains) = contains-concept object2 *y)))
```

2. The following is an Inference rule asserting that if y contains x and y contains z and z has the property unknown; then z is the subclass of superclass x. This may not seem to be logically correct but it is used subconsciously by the subjects participating in the think aloud protocols. This rule is used by Cassie to infer that dossiers are some kind of documents. From our passage archives contain dossiers and from our rule above archives contain documents and dossiers have the property unknown (this will be added later) thus this rule will fire and tell Cassie that dossiers are some kind of documents.

```
(describe (assert forall ($x $y $z)
        &ant (build object1 *y rel (build lex contains) = contains-concept object2 *x)
        &ant (build object1 *y rel *contains-concept object2 *z)
        &ant (build object *z property (build lex unknown) = unknown-concept)
        cq (build superclass *x subclass *z)))
```

3. The following rule asserts the background knowledge that archives contain documents. This rule adds to Cassie's knowledge base that archives contain documents. This is the background knowledge used by our subjects to infer the meaning of dossier. This helps Cassie to understand what archives are.

```
(describe (assert object1 *archives rel *contains-concept object2 (build lex documents)))
```

4. The following asserts the background knowledge that if x details y then x gives information about y. In our passage dossiers detail well authenticated cases. Thus using this rule, Cassie will be able to infer that dossiers give information about well authenticated cases.

```
(describe (assert forall ($x $y)
                ant (build agent *x act (build action (build lex detail) = detail-concept object *y))
                cq  (build agent *x act (build action (build lex provides\ information\ about) object *y))))
```

SNePS Representation of the Passage

The passage was divided into smaller parts for representing it in SNePS. Following are SNePSUL commands to build the SNePS network from the passage,

1. We tell Cassie that that there is something called Medical Departments. These medical departments can be later referenced by our code using either *med-dept or *med-dept-concept

```
(show (add member #med-dept class (build lex Medical\ Department) = med-dept-concept))
```

2. Here we tell Cassie that there is something called Lourdes.

```
(show (add object #lourdes proper-name lourdes))
```

3. Now we tell Cassie that the medical departments belong to Lourdes. We have no background knowledge of what Lourdes means. It is not necessary for inferring the unknown word meaning. Here we use the references for medical department and Lourdes defined above in rule 1 and 2.

```
(show (add object *med-dept rel *med-dept-concept possessor *lourdes))
```

3. Here we tell Cassie that there is something (dossier) called Dossiers. Thus, we are introducing the unknown word to Cassie.

(show (add subclass #dossiers superclass (build lex dossiers) = dossiers-concept))

4. The following tells Cassie that dossiers have the property unknown. This property in conjunction with inference rule 2 helps Cassie to infer that dossiers are a subclass of documents.

(show (add object *dossiers-concept property *unknown-concept))

5. Here we tell Cassie that the archives are filled with dossiers.

(show (add object1 *archives rel *filled-with-concept object2 *dossiers-concept))

6. Now we tell Cassie that the archives belong to the medical departments.

(show (add object *archives rel *archives-concept possessor *med-dept))

7. Here we introduce the concept of well authenticated cases

(show (add subclass #well-auth-cases superclass (build lex well-auth-cases) = well-auth-cases-concept))

8. Here we tell Cassie that that dossiers detail well-authenticated cases

(show (add agent *dossiers act (build action *detail-concept object *well-auth-cases)))

9. Here we tell Cassie that well authenticated cases are a subclass of cases. This rule can be omitted as it does not contribute to the inference process.

(show (add subclass *well-auth-cases superclass (build lex cases) = cases-concept))

Output

Now we ask Cassie what the unknown word “dossiers” appearing in the passage means.

^(defineNoun "dossiers")

Cassie replies with the following,

Definition of dossiers:

Class Inclusions: documents,

Possible Actions: provides information about b5, detail b5,

Possible Properties:

nil

We see that Cassie understands that dossiers are a type of documents with possible actions of providing information about b5.

"b5" is a node in the SNePS network and it represents the concept of well authenticated cases. We confirm this by asking Cassie to describe well authenticated cases. Cassie replies b5.

* (describe *well-auth-cases)

(b5)

Thus Cassie was able to infer the meaning of "dossiers" from context.

Modifications to the Noun Algorithm

Using the passage and inference rules 1, 2, 3 and passage (4, 5) Cassie was able to infer that dossiers are type of documents. To make Cassie infer that dossiers detail or provide information about well authenticated cases something more had to be done.

This information was present in Cassie's knowledge base (passage: 8). However the noun algorithm was overlooking this. The noun algorithm identifies the agent-act-action-object case frame to look up possible actions the unknown word performs. If the agent in the case frame is the member of the class of the noun under consideration then it outputs the action performed in the action part performed on the object in the object part of the case frame. In our code we have a representation (passage: 8) with the agent-act-action-object case frame, the agent being a subclass of the superclass of dossiers and the action being to detail well authenticated cases. However the noun algorithm fails to identify the subclass superclass case frame with

the agent-act-action-object case frame. Hence the noun algorithm was not able to perceive that dossiers have a possible action of detailing well authenticated cases. So the noun algorithm was modified to incorporate this. Following is a snippet of the noun algorithm and the modifications are highlighted in bold font.

```
;;;-------------------------------------------------------------------------
;;;
;;;          function: act-object-noun
;;;                                                  created: stn 2002
;;;-------------------------------------------------------------------------
(defun act-object-noun (noun)
   "Finds actions performed by at least one member of the category 'noun' and
     the objects that those actions are performed on."
  (let (actions)
    (setf actions #3! ((find (compose action- act- ! agent subclass- ! superclass lex) ~noun)))
    (mapcar #'(lambda (act) (obj-act-indiv noun act)) actions)))

;;;-------------------------------------------------------------------------
;;;
;;;          function: obj-act-indiv
;;;                                                  created: stn 2002
;;;-------------------------------------------------------------------------
(defun obj-act-indiv (noun action)
  "Finds the objects that 'noun' performs 'action' on."
  (let (objects)
    (setf objects #3! ((find (compose object- act- ! agent subclass- ! superclass lex) ~noun
                                    (compose object- action) ~action)))
    (if (null objects) action
      (mapcar #'(lambda (obj) (list action obj)) objects))))
```

Short Term Further Work:

At onset, it was decided that given the passage Cassie would make the following inferences,

1. Dossiers are some kind of documents
2. Have possible actions of providing more information about well authenticated cases and
3. As things that belong to a medical departments

Cassie is able to infer 1 and 2 above. However inference 3 is not being inferred by Cassie. The information is provided in the passage. The Passage (6) adds to Cassie's knowledge that the archives can be found in medical departments. From this and other rules, it can be inferred that dossiers are some kind of medical

documents. But due to shortage of time this could not be complete. However given a few more days, it would have been implemented successfully.

Long Term Further Work

We have modified the noun algorithm to incorporate the superclass – subclass case frame. We have done this by replacing the member - class case frame with subclass – superclass case frame. Hence now Cassie won't be able to recognize the member – class case frame when deriving possible actions for the unknown noun. Thus our Long term assignment would be to modify the noun algorithm to incorporate both case frames in the noun algorithm.

Appendix A: Think Aloud Protocols:

The passage was, "The Archives of the medical department of Lourdes are filled with smizers that detail well-authenticated cases of what are termed miraculous healings." Note that the word dossiers was replaced by smizers. This is what the subjects had to say,

=========
Subject I
=========

Smizers are related to archives (after reading first part of the sentence).
Something contained in archives.

Contains info about unusual cases.
Smizers are some chronicles.
Used to take record of cases.
Best guess - parts of the archives.

They are files or folders or compartments that contain unusual info
(unusual because miraculous healings are possible but unusual).

These cases have already happened i.e. genuine cases – clue from well authenticated.

All archives are reliable and well-authenticated. In this case
well-authenticated is used because miraculous healings are hard to believe.

Possible synonyms :- Chronicles, articles

It could mean people. Smizers sound like misers and misers are people.

---> Smizers are something contained in archives. (Like chronicles). They contain information on unusual cases.

==========
Subject II
==========

Written papers because it is a medical department. It is concerned with cases.

"Archive" word suggests data stored.

Filled with some object, so it could be some paper.

Some other object in medical department viz. skeleton of brain, specimens like in biology lab.

Synonyms

Specimen
Files (case files)

---> Smizers are papers or case files.

==========
Subject III
==========

Lourdes is a place where a miracle occured in 1980. Mother Mary appeared and healed.....

Filled with document or records because archive is the clue. It has to be hard copy evidence.

Detail well-authenticated cases so points to only documents or records.

Synonym :- records

---> Smizers are documents/records.

Possible answers:

dossiers are documents, cases or records.
they are found in medical departments.
they describe well-authenticated cases. (detail = describe)

Appendix B: Script file of sample run:

```
Script started on Fri Dec 10 17:39:56 2004
%mlisp International Allegro CL Enterprise Edition
6.2 [Solaris] (Sep 23, 2004 10:59)
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): :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:1a 2004/08/26 23:05:27] loaded.
Type `(sneps)' or `(snepslog)' to get started.
cl-user(2): (sneps)


   Welcome to SNePS-2.6 [PL:1a 2004/08/26 23:05:27]

Copyright (C) 1984--2004 by Research Foundation of
State University of New York. SNePS comes with ABSOLUTELY NO WARRANTY!
```

```
Type `(copyright)' for detailed copyright information.
Type `(demo)' for a list of example applications.

   12/10/2004 17:40:34

* (demo "dossiers.demo")

File /home/csgrad/pejusdas/public_html/2004fall/cse740/dossiers.demo is now the source of input.


 CPU time : 0.01

* ; ============================================================================
; FILENAME:     dossiers.demo
; DATE:             November 16 2004
; PROGRAMMER:          Pejus Das

;; 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 "dossier.demo" :av)
;
; Make sure all necessary files are in the current working directory
; or else use full path names. The necessary files are "dossier.demo", "defun_noun.cl"
; ============================================================================
;

; Turn off inference tracing.
; This is optional; if tracing is desired, then delete this.

^(
--> setq snip:*infertrace* nil)
nil



 CPU time : 0.00

*

; Load the appropriate definition algorithm:
; the noun algorithm has been modified to include the subclass superclass case frame to
; deduce the meaning of unknown words

^(
--> load "~/public_html/2004fall/cse740/defun_noun.cl")
; Loading
;    /home/csgrad/pejusdas/public_html/2004fall/cse740/defun_noun.cl
t
```

```
 CPU time : 0.27

*

; Clear the SNePS network:

(resetnet t)

Net reset


 CPU time : 0.01

*

; OPTIONAL:
; UNCOMMENT THE FOLLOWING CODE TO TURN FULL FORWARD INFERENCING ON:
; full forward inferencing is turned on. Hence the code has been uncommented.


;enter the "snip" package:
^(
--> in-package snip)
#<The snip package>



 CPU time : 0.00

*
;turn on full forward inferencing:
^(
--> defun broadcast-one-report (represent)
  (let (anysent)
    (do.chset (ch *OUTGOING-CHANNELS* anysent)
              (when (isopen.ch ch)
        (setq anysent
                    (or (try-to-send-report represent ch)
                        anysent)))))
  nil)
broadcast-one-report



 CPU time : 0.00

*
;re-enter the "sneps" package:
^(
--> in-package sneps)
#<The sneps package>



 CPU time : 0.00

*
```

```
; load all pre-defined relations:
(intext "/projects/rapaport/CVA/STN2/demos/rels")
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 : 0.01

*

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


 CPU time : 0.02

*
; load all 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.


 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 : 0.00

*

; BACKGROUND KNOWLEDGE:
; =====================
; this section describes the background knowledge necessary to infer the meaning of the unknown word.

; the following is a rule asserting that if x is filled with y then x contains y.

(describe (assert forall ($x $y)
            ant(build object1 *x rel (build lex filled-with) = filled-with-concept object2 *y)
            cq (build object1 *x rel (build lex contains) = contains-concept object2 *y)))

(m3! (forall v2 v1)
 (ant (p1 (object1 v1) (object2 v2) (rel (m1 (lex filled-with)))))
 (cq (p2 (object1 v1) (object2 v2) (rel (m2 (lex contains))))))

(m3!)

 CPU time : 0.01

*

; the following is a rule asserting that if y contains x and y contains z and z has the property unknown
; then z is subclass of superclass x. one of those weird rules. :).

(describe (assert forall ($x $y $z)
            &ant (build object1 *y rel (build lex contains) = contains-concept object2 *x)
            &ant (build object1 *y rel *contains-concept object2 *z)
            &ant (build object *z property (build lex unknown) = unknown-concept)
            cq (build superclass *x subclass *z)))

(m5! (forall v5 v4 v3)
```

```
 (&ant (p5 (object v5) (property (m4 (lex unknown))))
  (p4 (object1 v4) (object2 v5) (rel (m2 (lex contains))))
  (p3 (object1 v4) (object2 v3) (rel (m2))))
 (cq (p6 (subclass v5) (superclass v3))))

(m5!)

 CPU time : 0.00

*

; this asserts that there is something (archive) called as Archives
(describe (add subclass #archives superclass (build lex archives) = archives-concept))

(m7! (subclass b1) (superclass (m6 (lex archives))))

(m7!)

 CPU time : 0.01

*

; the following rule asserts the background knowledge that archives
; contain documents. This references "archives" defined above
(describe (assert object1 *archives rel *contains-concept object2 (build lex documents)))

(m9! (object1 b1) (object2 (m8 (lex documents)))
 (rel (m2 (lex contains))))

(m9!)

 CPU time : 0.00

*

;asserts the background knowledge that if x details y then x gives information about y.

(describe (assert forall ($x $y)
                        ant (build agent *x act (build action (build lex detail) = detail-concept object *y))
                        cq  (build agent *x act (build action (build lex provides\ information\ about) object *y))))

(m12! (forall v7 v6)
 (ant
  (p8 (act (p7 (action (m10 (lex detail))) (object v7))) (agent v6)))
 (cq
  (p10
   (act (p9 (action (m11 (lex provides information about)))
        (object v7)))
   (agent v6))))

(m12!)

 CPU time : 0.00

*

; CASSIE READS THE PASSAGE:
```

```
; ========================
; the following adds the passage in which the unknown word appears to cassie's knowledge base

; the following adds that there is something (med-dept) called Medical Department
(add member #med-dept class (build lex Medical\ Department) = med-dept-concept)

(m14!)

 CPU time : 0.01

*
; the following adds that there is something (lourdes) called Lourdes
(add object #lourdes proper-name lourdes)

(m15!)

 CPU time : 0.00

*
; the following rule adds "the medical department of lourdes"
(describe (add object *med-dept rel *med-dept-concept possessor *lourdes))

(m16! (object b2) (possessor b3) (rel (m13 (lex Medical Department))))

(m16!)

 CPU time : 0.01

*
; the following adds that there is something (dossier) called Dossiers
(describe (add subclass #dossiers superclass (build lex dossiers) = dossiers-concept))

(m18! (subclass b4) (superclass (m17 (lex dossiers))))

(m18!)

 CPU time : 0.00

*
; adds that dossier have property unknown
(describe (add object *dossiers-concept property *unknown-concept))

(m19! (object (m17 (lex dossiers))) (property (m4 (lex unknown))))
(m9! (object1 b1) (object2 (m8 (lex documents)))
 (rel (m2 (lex contains))))

(m19! m9!)

 CPU time : 0.02

*
; adds dossiers as member of class of dossiers
;(describe (add member *dossiers class (build lex dossier) = dossier-concept))


; the rule adds that the archive are filled with dossiers
(describe (add object1 *archives rel *filled-with-concept object2 *dossiers-concept))
```

```
(m22! (subclass (m17 (lex dossiers))) (superclass (m8 (lex documents))))
(m21! (object1 b1) (object2 (m17)) (rel (m2 (lex contains))))
(m20! (object1 b1) (object2 (m17)) (rel (m1 (lex filled-with))))
(m9! (object1 b1) (object2 (m8)) (rel (m2)))

(m22! m21! m20! m9!)

 CPU time : 0.03

*
; the rule adds that the archives belong to the medical departments
(describe (add object *archives rel *archives-concept possessor *med-dept))

(m23! (object b1) (possessor b2) (rel (m6 (lex archives))))

(m23!)

 CPU time : 0.01

*
; the rule adds that there is something (#well-auth-cases) called well-auth-cases
(describe (add subclass #well-auth-cases superclass (build lex well-auth-cases) = well-auth-cases-concept))

(m25! (subclass b5) (superclass (m24 (lex well-auth-cases))))

(m25!)

 CPU time : 0.00

*
; the rule adds that dossiers detail well-authenticated cases
(describe (add agent *dossiers act (build action *detail-concept object *well-auth-cases)))

(m29!
 (act (m28 (action (m11 (lex provides information about)))
       (object b5)))
 (agent b4))
(m27! (act (m26 (action (m10 (lex detail))) (object b5))) (agent b4))

(m29! m27!)

 CPU time : 0.01

*
; the rule adds that well authenticated cases are a subclass of cases
;(describe (add subclass *well-auth-cases superclass (build lex cases) = cases-concept))

; describe the entire network
;(describe *nodes)


; Ask Cassie what "WORD" means:
;============================

^(
--> defineNoun "dossiers")
```

Definition of dossiers:
Class Inclusions: documents,
Possible Actions: provides information about b5, detail b5,
Possible Properties:
nil

CPU time : 0.11

*

End of /home/csgrad/pejusdas/public_html/2004fall/cse740/dossiers.demo demonstration.

CPU time : 0.55

* (describe *well-auth-cases)

(b5)

(b5)

CPU time : 0.00

* (lisp)
"End of SNePS"
cl-user(3): :exit
; Exiting
%exit exit

script done on Fri Dec 10 17:41:05 2004

Appendix C: SNePS network diagrams

Following diagrams represent the internal representation of the background rules and passage.

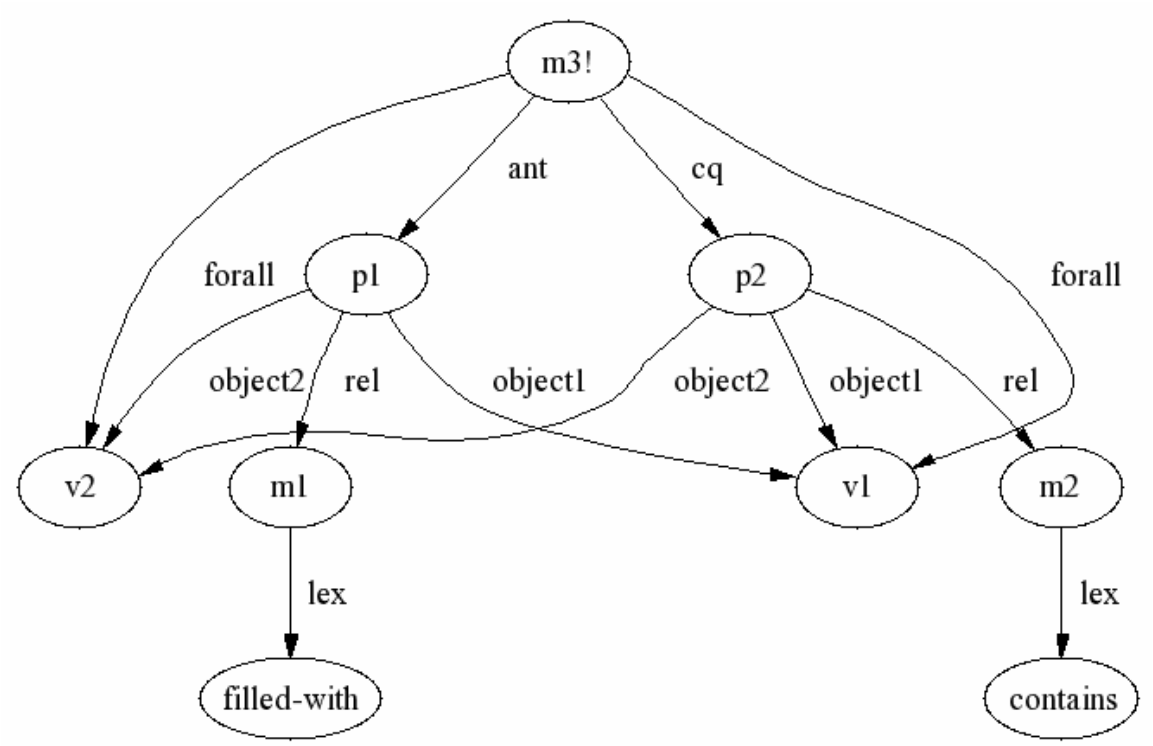

Figure 2: If x is filled with y then x contains y

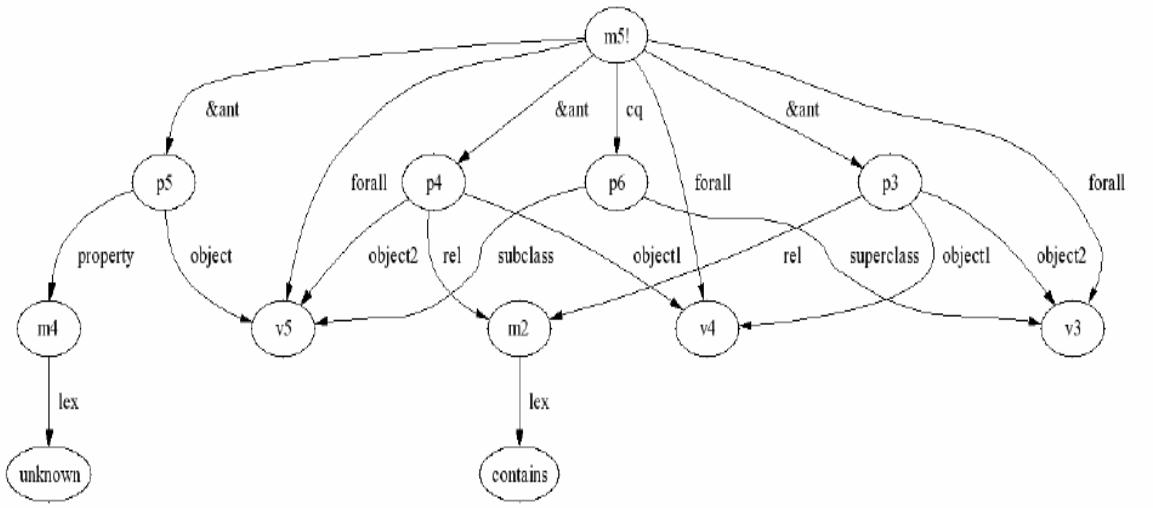

Figure 3: If y contains x and y contains z and z has the property unknown then z is subclass of super class x

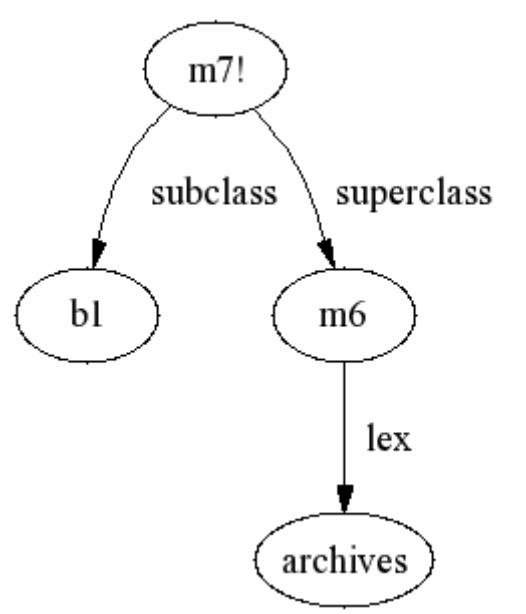

Figure 4: There is something (archive) called as Archives

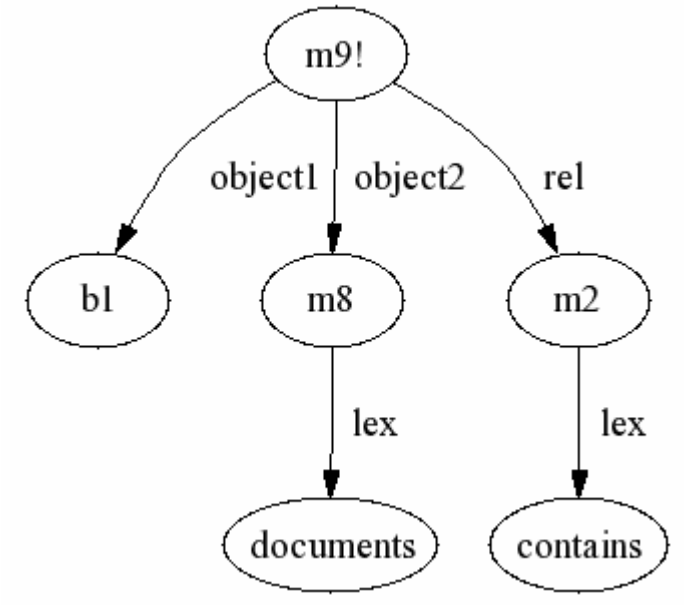

Figure 5: Archives contain documents

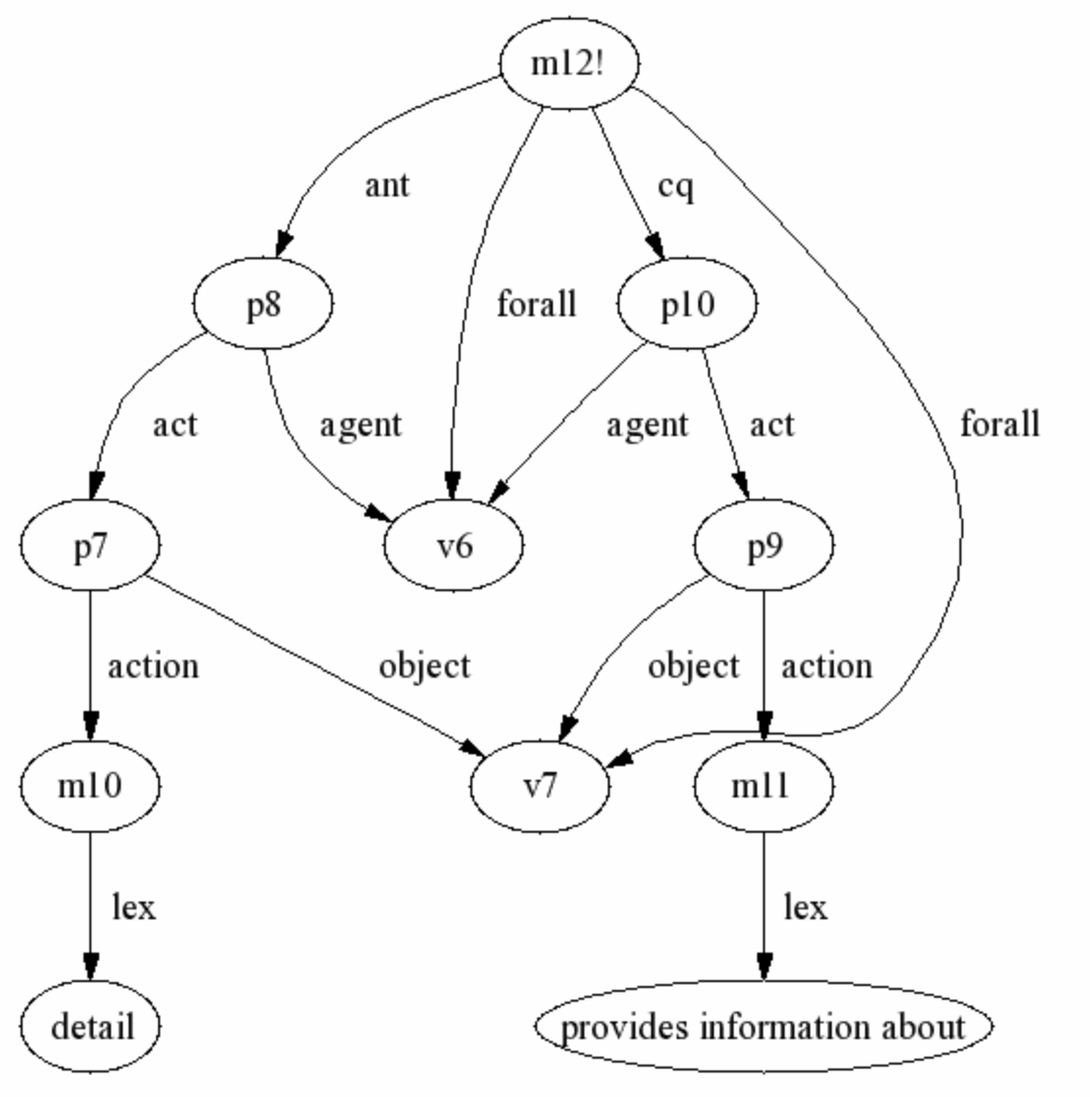

Figure 6: If x details y then x gives information about y

The following is the network representation of the passage

1.

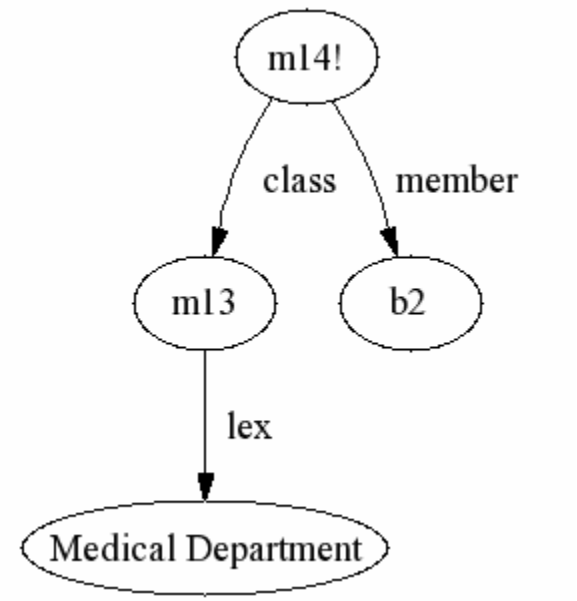

Figure 7: There is something (med-dept) called Medical Departments

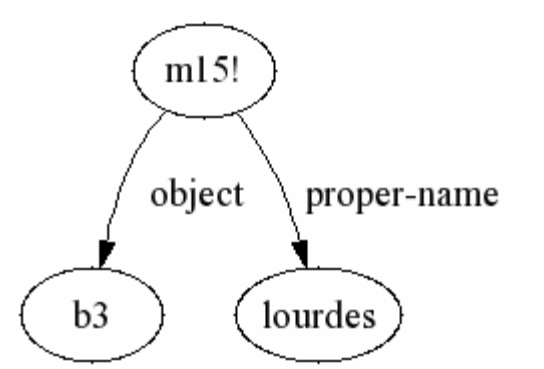

Figure 8: There is something (lourdes) called Lourdes

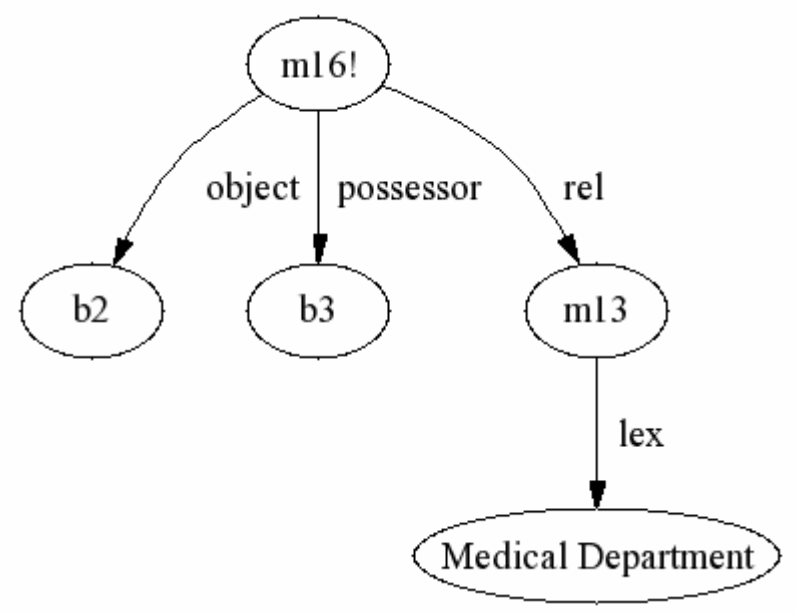

Figure 9: The medical department of Lourdes

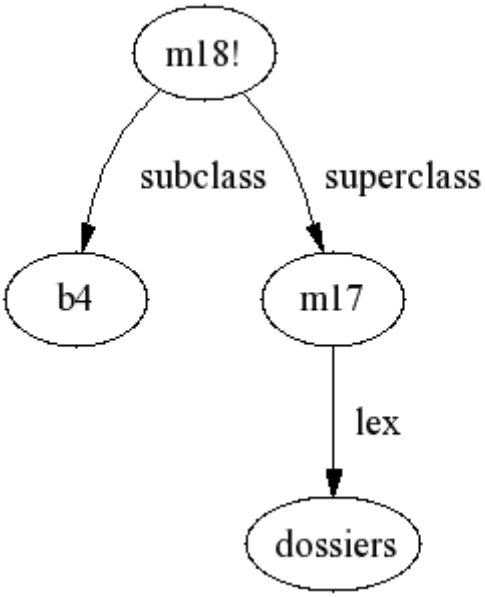

Figure 10: There is something (dossier) called dossiers

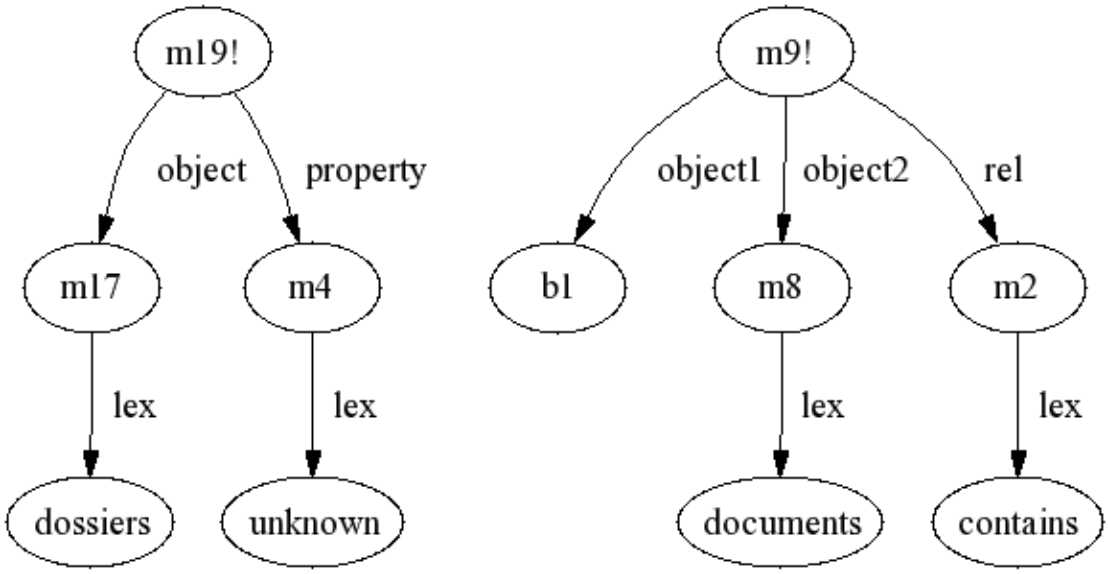

Figure 11: Dossiers have the property unknown

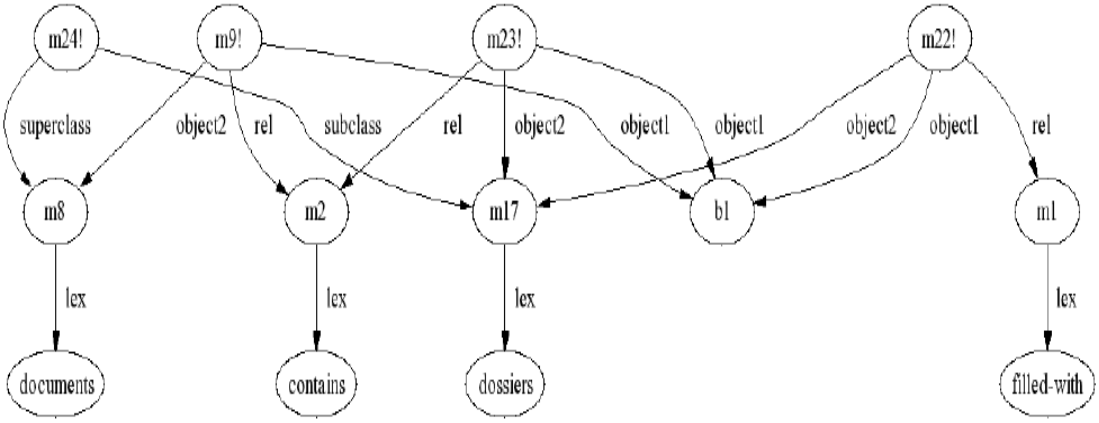

Figure 12: The archives are filled with dossiers

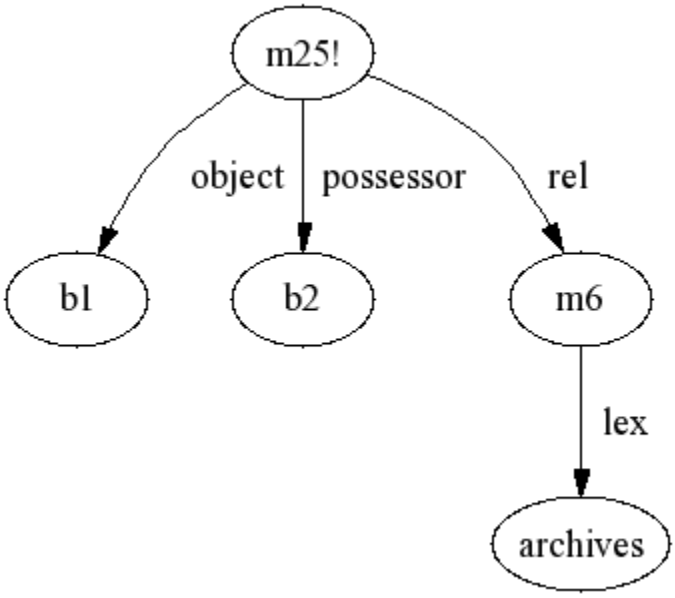

Figure 13: Archives belong to the medical departments

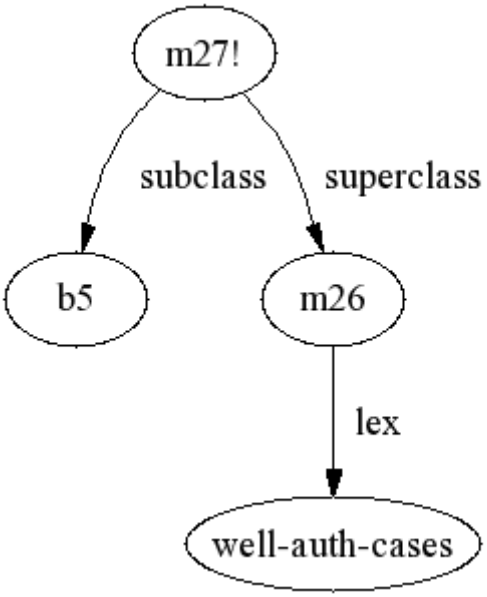

Figure 14: There is something (#well-auth-cases) called well-auth-cases

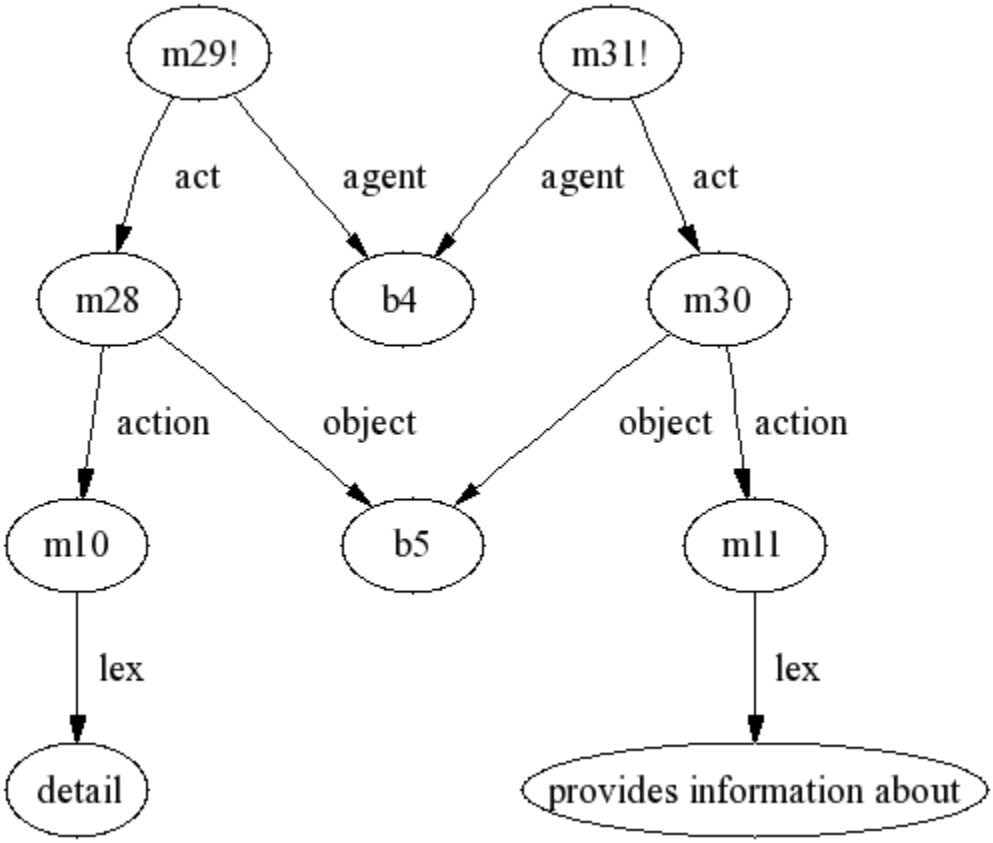

Figure 15: Dossiers detail well-authenticated cases

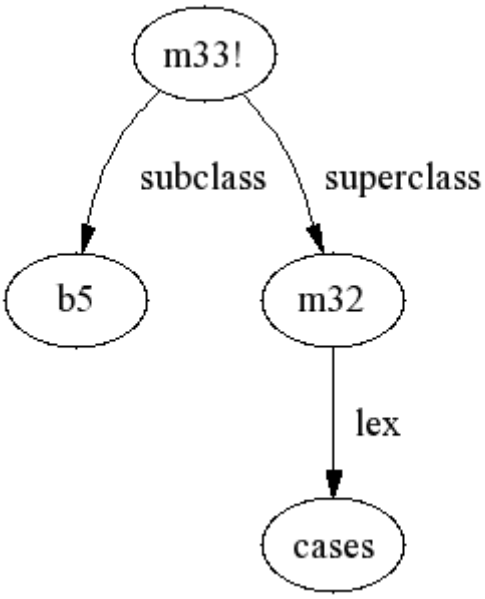

Figure 16: Well authenticated cases are a subclass of cases

References

1. http://www.cse.buffalo.edu/~rapaport/740/F04/syl.html
2. http://www.cse.buffalo.edu/sneps/
3. Martins, João P. (2002), Section on SNePS from draft of forthcoming knowledge representation text