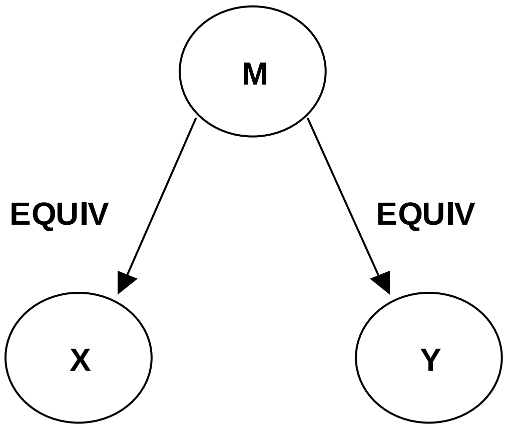
Appendix A: Syntax and Semantics for non standard case frames



[[M]] is the proposition which tell us that [[x]] and [[y]] implies each other.

Appendix B: Script of the running demo

```
==============================================================
Starting image `/util/acl62/composer'
  with no arguments
  in directory `/home/eegrad/clollett/cse740/'
  on machine `localhost'.

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): :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.
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.

   4/29/2004 7:43:38

* (demo "mykolperv18ng.demo")

File /home/eegrad/clollett/cse740/mykolperv18ng.demo is now the source
of input.


 CPU time : 0.01

* ;
=======================================================================
; FILENAME: WORD.demo
; DATE:           DATE
; PROGRAMMER:     YOUR_NAME

;; NOTE TO PROGRAMMER:  GLOBALLY REPLACE "WORD" BY YOUR WORD,
;;                  March 22, 2004
;;                  CARLOS LOLLETT
;; 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 "mykolperv2.demo" :av)
;
; Make sure all necessary files are in the current working directory
; or else use full path names.
;
=======================================================================

; Turn off inference tracing.
; This is optional; if tracing is desired, then delete this.
^(
--> setq snip:*infertrace* nil)
nil



 CPU time : 0.00

*
; Load the appropriate definition algorithm:
;; UNCOMMENT THE ONE YOU *DO* WANT
;; AND DELETE THE OTHER!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
^(
--> load "/projects/rapaport/CVA/STN2/defun_noun.cl")
; Loading /projects/rapaport/CVA/STN2/defun_noun.cl
t



 CPU time : 0.18

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

; Clear the SNePS network:
(resetnet)

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:
 ^(
--> 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 : 0.02

*

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 : 0.01

*
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 : 0.01
```

```
*

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


 CPU time : 0.02

* (define-path class (compose class (kstar (compose subclass- !
superclass))))
class implied by the path (compose class
                           (kstar (compose subclass- ! superclass)))
class- implied by the path (compose (kstar (compose superclass- !
subclass))
                            class-)


 CPU time : 0.00

*
; loading visualization tool
;^(load "show")
; KOLPER
; SENTENCE 2:
; ==========================
; "He virtually always study in the library, as at home he had to work
by artificial light all day because of those kolpers"
; Giving a equivalent meaning but more suitable sentence for
translation

; ' He virtually always study in the library,

; because

; at home, all day, he had to work by artificial light

; because

; those kolpers '


; Two goals(based on van Daalen-Kapteijns and Elshout-Mohr, 1981):

; Goal 1:
; ========
; Get a reformulation of the sentence to have a initial direct
definition
; of kolper
; "kolpers in a house mean having artificial light on all day"

; Goal 2:
; ========
; Inference from the background knowledge
; "kolpers transmit little light"
; However, this inference is done under the assumption of kolper being
a kind of window
; Therefore, it could not be achieve completely
; Informal surveys showed that kolper meaning based only on this
sentence was considered a abstract object.
; e.g "It is something that stop natural illumination", "It is an
object that cannot receive sunlight"
; means that there are at least two possible interpretations
; Active: kolper is actively stopping natural light
; Pasive: kolper is the reason that natural light shouldn't enter to
the room. e.g. Gremlims
; Currently I am working in the Active interpretation

; BACKGROUND KNOWLEDGE:
; =====================
; (put annotated SNePSUL code of your background knowledge here)


; Establishing the condition that library and home are subclasses of
indoors
(describe (assert subclass (build lex "library") superclass (build lex
"indoors")))

(m3! (subclass (m1 (lex library))) (superclass (m2 (lex indoors))))

(m3!)

 CPU time : 0.00

* (describe (assert subclass (build lex "home") superclass (build lex
"indoors")))

(m5! (subclass (m4 (lex home))) (superclass (m2 (lex indoors))))

(m5!)

 CPU time : 0.01

*






; Another Rule-based inference in order to establish a equivalence
relationship
; 'If somebody is human and someplace is an indoor then To say
somebody(human) is in someplace and it is required to use artificial
light all-day is equivalent to say that someplace  must be a bad
natural iluminated place



( describe (
      assert forall ($place $person)
      &ant (build member *place class (build lex "indoors"))
      &ant (build member *person class (build lex "human"))
      cq (build
            equiv ( build
                  object1 (build
                        agent *person
                        act ( build
                              action ( build
                                    object (build
                                          object (build lex "work" )
                                          location *place
                                    )
                                    property ( build lex "all_day")
                              )
                        )
                  )
                  rel (build lex "require")
                  object2 (build lex "artificial light")
            )
            equiv (build object *place property (build lex "badlight")
            )
      )
))

(m12! (forall v2 v1)
 (&ant (p2 (class (m6 (lex human))) (member v2))
  (p1 (class (m2 (lex indoors))) (member v1)))
 (cq
  (p9
   (equiv (p8 (object v1) (property (m11 (lex badlight))))
    (p7
     (object1
      (p6
       (act (p5
             (action
              (p4 (object (p3 (location v1) (object (m7 (lex work)))))
               (property (m8 (lex all_day)))))))
       (agent v2)))
     (object2 (m10 (lex artificial light))) (rel (m9 (lex
require))))))))

(m12!)

 CPU time : 0.01

*
```

```
; A ruled based to extend the function relationship to equivalent
concepts
(describe (assert forall ($myv1 $myv2 $myk)
              &ant (build equiv *myv1 equiv *myv2)
              &ant (build object1 *myk rel (build lex "cause") object2
*myv1)
```

```
              cq (build object1 *myk rel (build lex "cause") object2
*myv2)
      )
)

(m14! (forall v5 v4 v3)
 (&ant (p11 (object1 v5) (object2 v3) (rel (m13 (lex cause))))
  (p10 (equiv v4 v3)))
 (cq (p12 (object1 v5) (object2 v4) (rel (m13)))))

(m14!)

 CPU time : 0.00

*


; Rule-based inference. "if a place is a bad natural illuminated place
then the light is not entering in that place
(describe (assert forall $place2
                  ant (build object *place2 property (build lex
"badlight"))
                  cq (build object *place2 property (build lex "ligh_
not_entering_there"))
            )

)
(m16! (forall v6) (ant (p13 (object v6) (property (m11 (lex
badlight)))))
 (cq (p14 (object v6) (property (m15 (lex ligh_
not_entering_there))))))

(m16!)

 CPU time : 0.00

*



;(describe (assert forall ($st1 $st2) ant(build object1 *st1 rel (build
lex "cause") object2 *st2 ) cq (build mode (build lex "presumably")
object (build object1 *st1 rel (build lex "function") object2 *st2))))


; Establishing the a probable relation between causality and funtion
(describe (assert forall ($st1 $st2)
                  ant(build object1 *st1 rel (build lex "cause")
object2 *st2 )
                  cq (build object1 *st1 rel (build lex "function")
object2 *st2)
            )
      )

(m18! (forall v8 v7)
 (ant (p15 (object1 v7) (object2 v8) (rel (m13 (lex cause)))))
 (cq (p16 (object1 v7) (object2 v8) (rel (m17 (lex function))))))
```

```
(m18!)

 CPU time : 0.00

*




; CASSIE READS THE PASSAGE:
; =========================
; (put annotated SNePSUL code of the passage here)

;Original Sentence:
;"He virtually always studied in the library,as at home he had to work
by artificial light all day because of those kolpers"
;Revised Sentence:
;He virtually always studied in the library, because at home, all day,
he had to work by artificial light because of those kolpers
;Three sections connected by cause-:
; 1) He virtually always studied in the library

; 2) at home, all day, he had to work by artificial light
; 3) those kolpers

; 1) He virtually always studied in the library
; he is human
(describe (add member #henry class (build lex "human")))
(m19! (class (m6 (lex human))) (member b1))

(m19!)

 CPU time : 0.00

*
; location is a library
(describe (add member #mylibrary class (build lex "library")))

(m28!
 (equiv
  (m27
   (object1
    (m26
     (act (m25
           (action
            (m24 (object (m23 (location b2) (object (m7 (lex work)))))
             (property (m8 (lex all_day)))))))
     (agent b1)))
   (object2 (m10 (lex artificial light))) (rel (m9 (lex require))))
  (m22 (object b2) (property (m11 (lex badlight))))))
(m21! (class (m2 (lex indoors))) (member b2))
(m20! (class (m1 (lex library))) (member b2))

(m28! m21! m20!)

 CPU time : 0.01
```

```
* ; He virtually always studied in the library
(describe (add

      object (build
            object (build
                  agent *henry
                  act (build
                        action (build
                              object (build
                                    object (build lex study)
                                    location *mylibrary
                              )
                              property (build lex "virtually always")
                        )
                  )
            )

      )
      ))

(m36!
 (object
  (m35
   (object
    (m34
     (act (m33
           (action
            (m32 (object (m30 (location b2) (object (m29 (lex
study)))))
             (property (m31 (lex virtually always)))))))
     (agent b1))))))

(m36!)

 CPU time : 0.01

*

; Causality relation between 1 and the other two sentence is omitted at
this stage

; 2) at home, all day, he had to work by artificial light
; myhome is a home
(describe (add member #myhome class (build lex "home")))

(m45!
 (equiv
  (m44
   (object1
    (m43
     (act (m42
           (action
            (m41 (object (m40 (location b3) (object (m7 (lex work)))))
             (property (m8 (lex all_day)))))))
     (agent b1)))
   (object2 (m10 (lex artificial light))) (rel (m9 (lex require))))
  (m39 (object b3) (property (m11 (lex badlight))))))
```

```
(m38! (class (m2 (lex indoors))) (member b3))
(m37! (class (m4 (lex home))) (member b3))

(m45! m38! m37!)

 CPU time : 0.02

*
;at home, all day, he had to work by artificial light
(describe (add
            object1 (build
                  agent *henry
                  act ( build
                        action ( build
                              object ( build
                                    object (build lex "work" )
                                    location *myhome
                              )
                              property (build lex "all_day")
                        )
                  )
            )
            rel (build lex "require")
            object2 (build lex "artificial light")

      ))

(m44!
 (object1
  (m43
   (act (m42
         (action
          (m41 (object (m40 (location b3) (object (m7 (lex work)))))
           (property (m8 (lex all_day)))))))
   (agent b1)))
 (object2 (m10 (lex artificial light))) (rel (m9 (lex require))))

(m44!)

 CPU time : 0.00

*
```

```
; 3) those kolpers
(describe (add member #mykolper class (build lex "kolper")))

(m47! (class (m46 (lex kolper))) (member b4))

(m47!)
```

```
 CPU time : 0.01

*


; Causal relation from 3(cause) and 2(consequence) using function
relation, to test noun algorithm(besides function there is no other
relation used in noun algorithm)
(describe (add
      object1 *mykolper
      rel (build lex "cause")
      object2 (build
            object1 (build
                  agent *henry
                  act ( build
                        action ( build
                              object ( build
                                    object (build lex "work" )
                                    location *myhome
                              )
                              property (build lex "all_day")
                        )
                  )
            )
            rel (build lex "require")
            object2 (build lex "artificial light")
      )
))

(m49! (object1 b4) (object2 (m39 (object b3) (property (m11 (lex
badlight)))))
 (rel (m13 (lex cause))))
(m48! (object1 b4)
 (object2
  (m44!
   (object1
    (m43
     (act (m42
           (action
            (m41 (object (m40 (location b3) (object (m7 (lex work)))))
             (property (m8 (lex all_day)))))))
     (agent b1)))
   (object2 (m10 (lex artificial light))) (rel (m9 (lex require)))))
 (rel (m13)))

(m49! m48!)

 CPU time : 0.07

*



; big causal relation from 1 to 2 excluded at this stage
(describe (add
            object1 ( build
                  object1 (build
                        agent *henry
                        act ( build
                              action ( build
                                    object ( build
                                          object (build lex "work" )
                                          location *myhome
                                    )
                                    property (build lex "all_day")
                              )
                        )
                  )
                  rel (build lex "require")
                  object2 (build lex "artificial light")

            )
            rel (build lex "cause")
            object2 (build
                  object (build
                        object (build
                              agent *henry
                              act (build
                                    action (build
                                          object (build
                                                object (build lex
study)
                                                location *mylibrary
                                          )
                                          property (build lex
"virtually always")
                                    )
                              )
                        )
                  )
            )
))

(m51!
 (object1
  (m44!
   (object1
    (m43
     (act (m42
           (action
            (m41 (object (m40 (location b3) (object (m7 (lex work)))))
             (property (m8 (lex all_day)))))))
     (agent b1)))
   (object2 (m10 (lex artificial light))) (rel (m9 (lex require)))))
 (object2
  (m36!
   (object
    (m35
     (object
      (m34
       (act (m33
             (action
              (m32 (object (m30 (location b2) (object (m29 (lex
study)))))
```

```
               (property (m31 (lex virtually always)))))))
       (agent b1)))))))
 (rel (m17 (lex function))))
(m50! (object1 (m44!)) (object2 (m36!)) (rel (m13 (lex cause))))
(m19! (class (m6 (lex human))) (member b1))

(m51! m50! m19!)

 CPU time : 0.03

*
;(show m51)

;========== From sentence 1 =============

; "Kolper is a window"
(describe (add subclass (build lex "kolper") superclass (build lex
"window")))

(m58! (subclass (m46 (lex kolper))) (superclass (m57 (lex window))))

(m58!)

 CPU time : 0.01

*




; Deductions
; ===================
;
; This information wasn't inferred from passage information
(describe (deduce object1 *mykolper rel (build lex "function") object2
$j))

(m60! (object1 b4) (object2 (m39 (object b3) (property (m11 (lex
badlight)))))
 (rel (m17 (lex function))))
(m59! (object1 b4)
 (object2
  (m44!
   (object1
    (m43
     (act (m42
           (action
            (m41 (object (m40 (location b3) (object (m7 (lex work)))))
             (property (m8 (lex all_day)))))))
     (agent b1)))
   (object2 (m10 (lex artificial light))) (rel (m9 (lex require)))))
 (rel (m17)))

(m60! m59!)

 CPU time : 0.03
```

```
*

; Ask Cassie what "KOLPER" means:
;; UNCOMMENT THE ONE YOU *DO* WANT
;; AND DELETE THE OTHER!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
^(
--> defineNoun "kolper")
 Definition of kolper:
 Class Inclusions: window,
 Possible Properties: function m36, function m39, function m44, cause
m36, cause m39, cause m44,
nil



 CPU time : 0.05

*


; ^(defineVerb "WORD")

;(show *nodes)

End of /home/eegrad/clollett/cse740/mykolperv18ng.demo demonstration.


 CPU time : 0.53

* (describe m36 m39 m44)

(m44!
 (object1
  (m43
   (act (m42
         (action
          (m41 (object (m40 (location b3) (object (m7 (lex work)))))
           (property (m8 (lex all_day)))))))
   (agent b1)))
 (object2 (m10 (lex artificial light))) (rel (m9 (lex require))))
(m39 (object b3) (property (m11 (lex badlight))))
(m36!
 (object
  (m35
   (object
    (m34
     (act (m33
           (action
            (m32 (object (m30 (location b2) (object (m29 (lex
study)))))
             (property (m31 (lex virtually always)))))))
     (agent b1))))))

(m44! m39 m36!)
```

CPU time : 0.00

Appendix C: SNePS network Diagrams

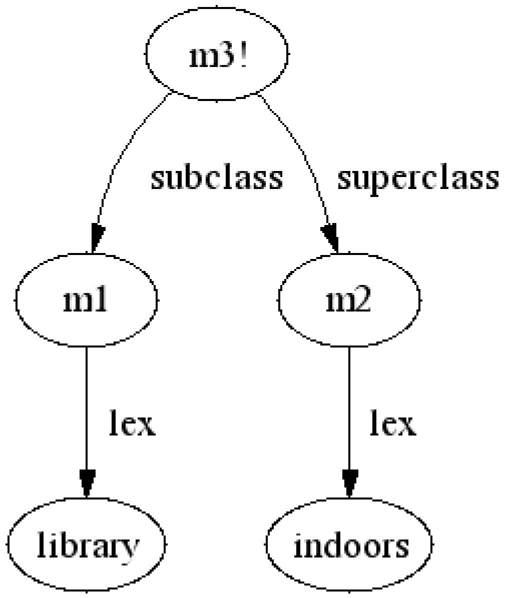


Fig 1. Library is a subclass of the superclass indoors

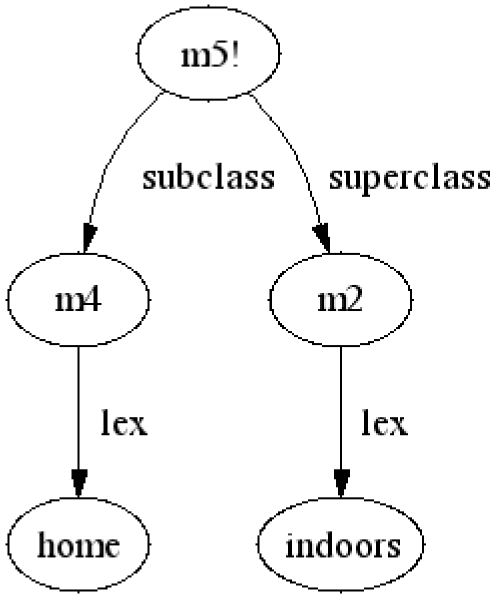


Fig 2. Home is a subclass of the superclass indoors

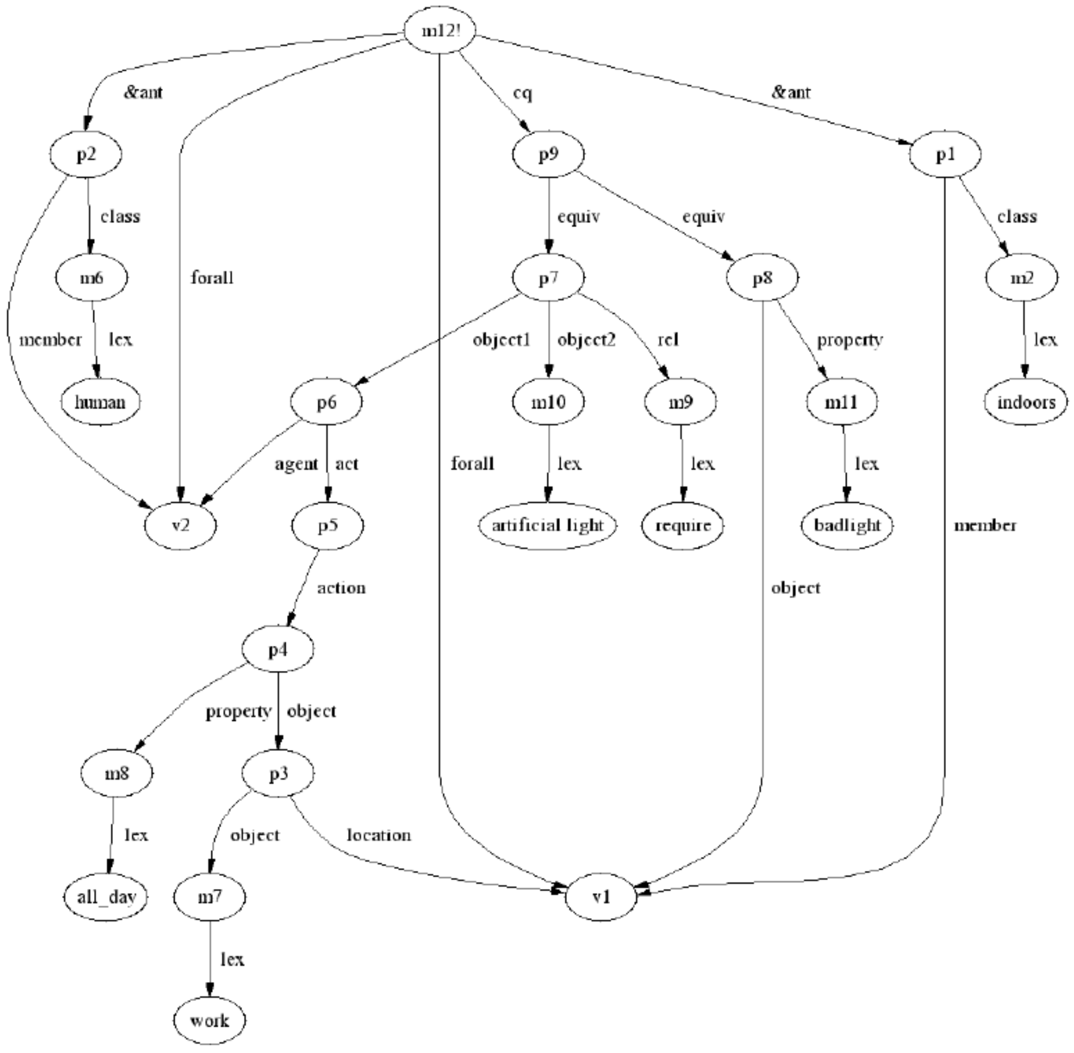


Fig 3. For any indoor place v1 and human v2, v2 work at v1 all day require articial light is equivalent to say that v1 has bad natural illumination

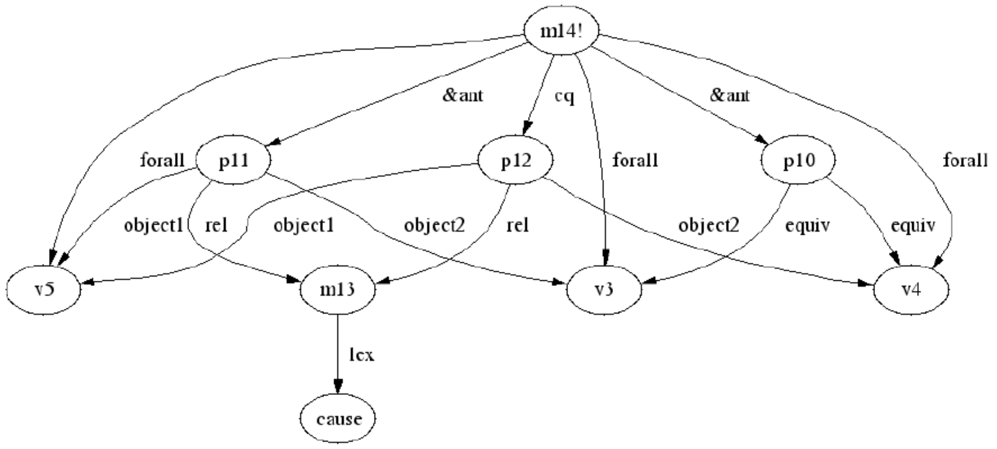


Fig 4. For any objects v3,v4,v5 if v5 cause v3 and v3 and v4 are equivalent then v5 cause v4

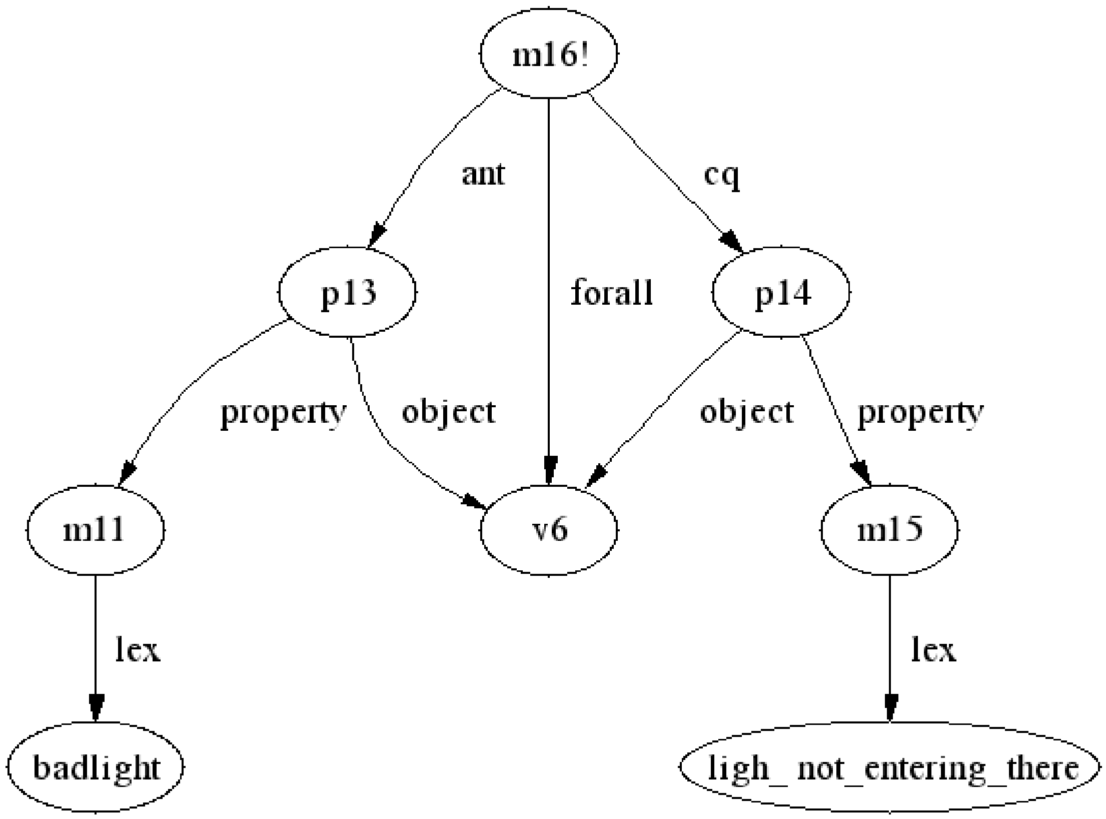


Fig 5. if a place has bad natural illumination the light is not entering to that place.

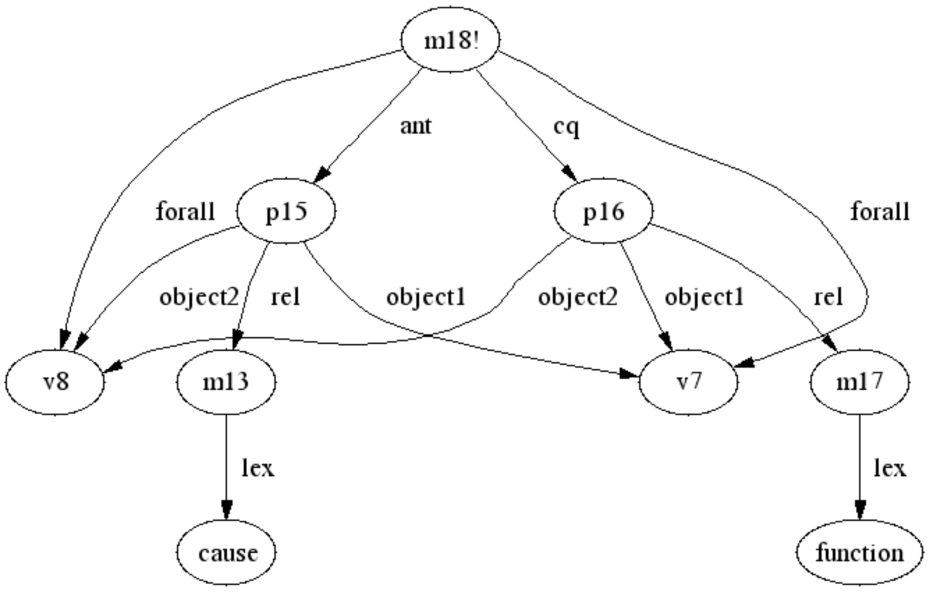


Fig 6. If v7 cause v8 v7's function is v8

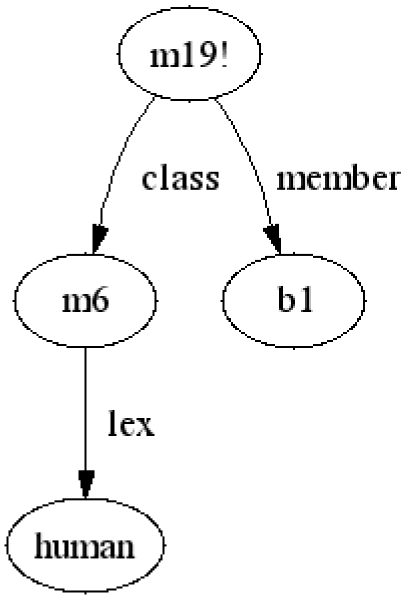


Fig 7. b1 is human

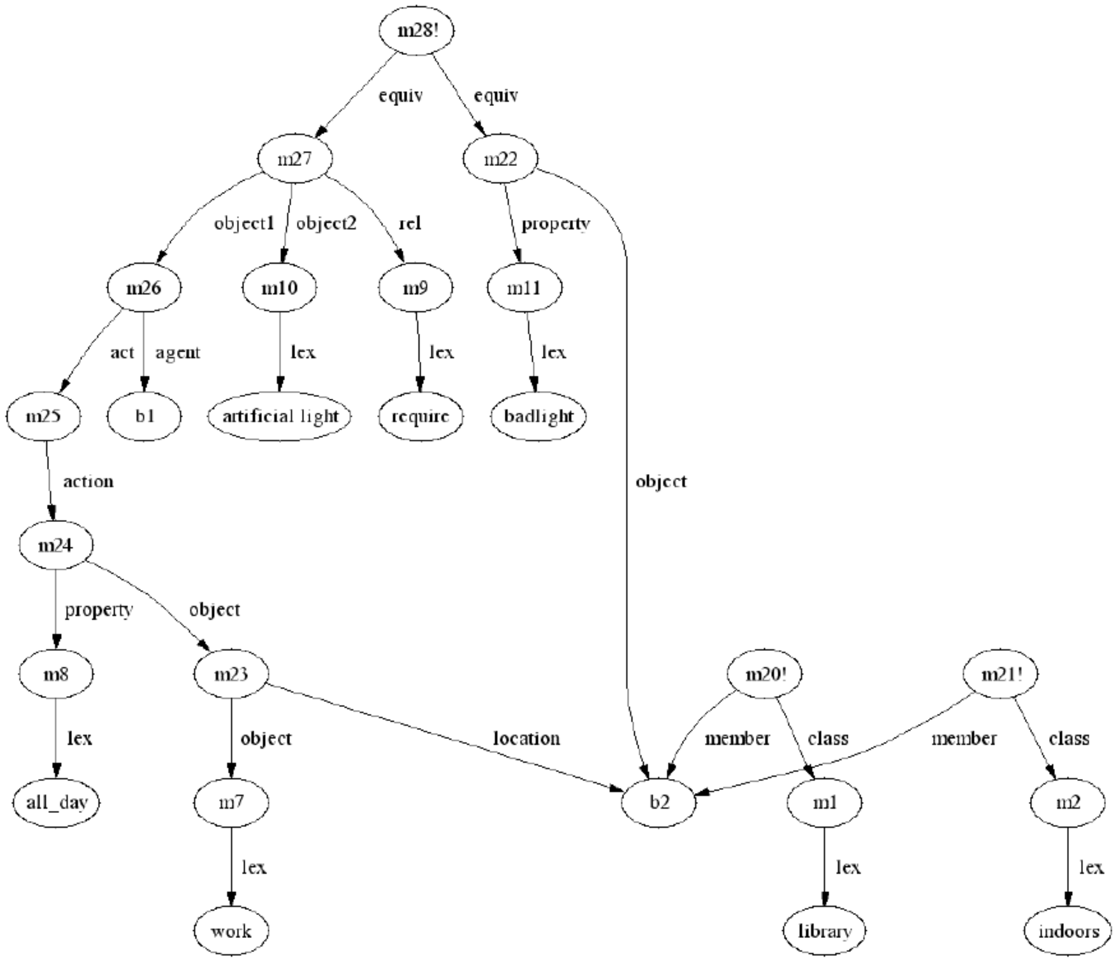


Fig 8. (left part) b2 is a library and then an indoors

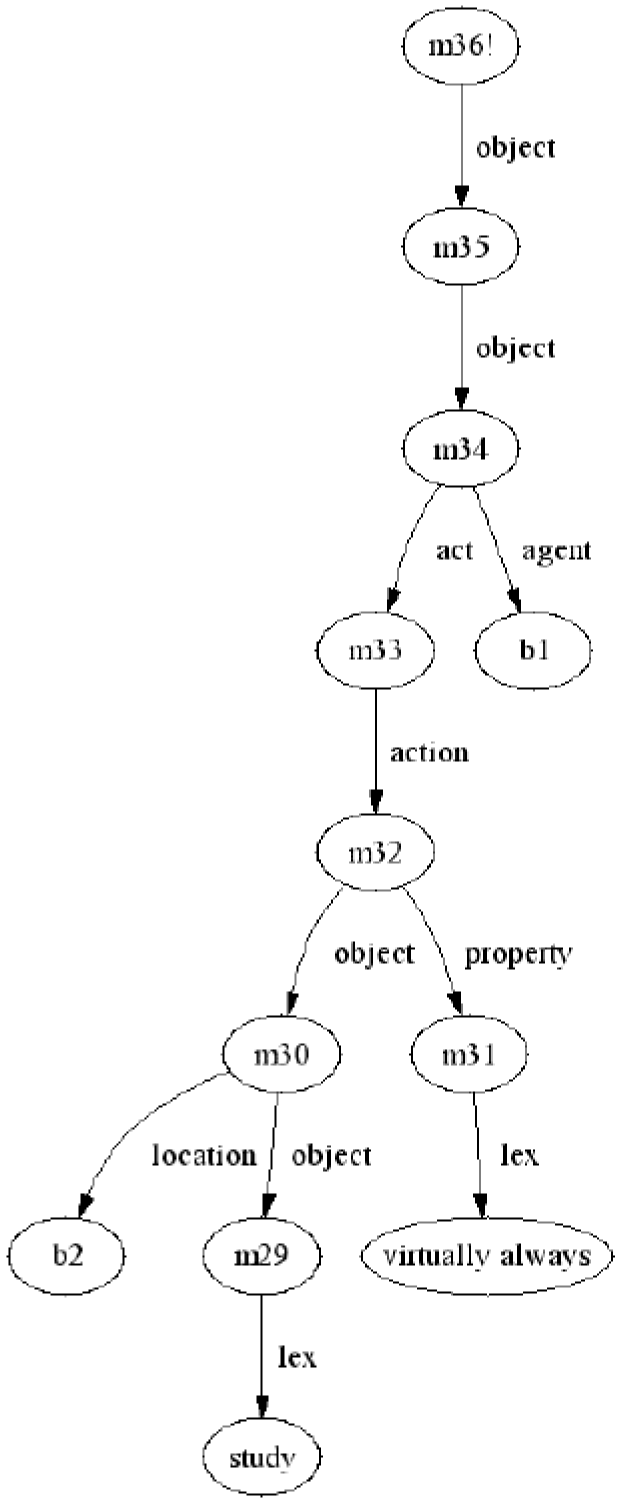


Fig 9. he virtually always study at the library

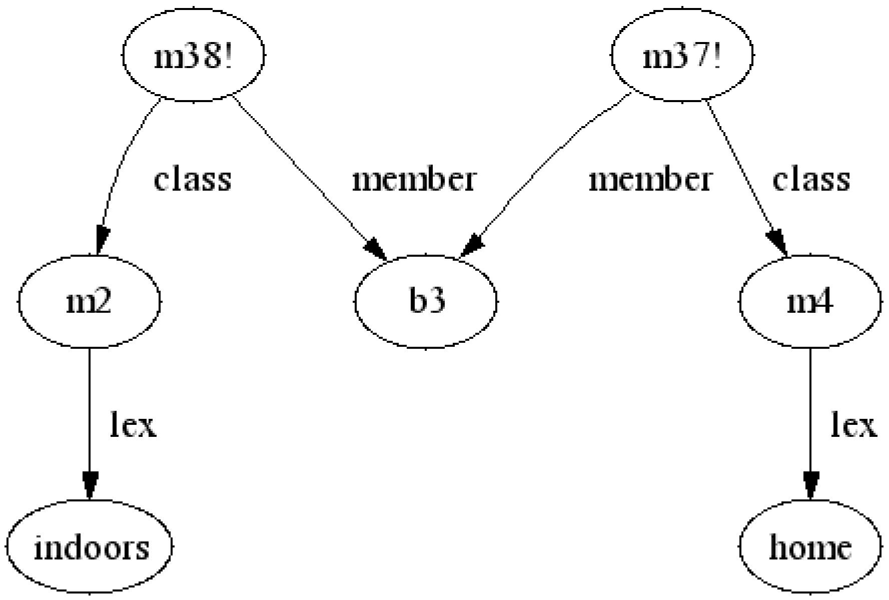


Fig 10. b3 is a home and an indoors

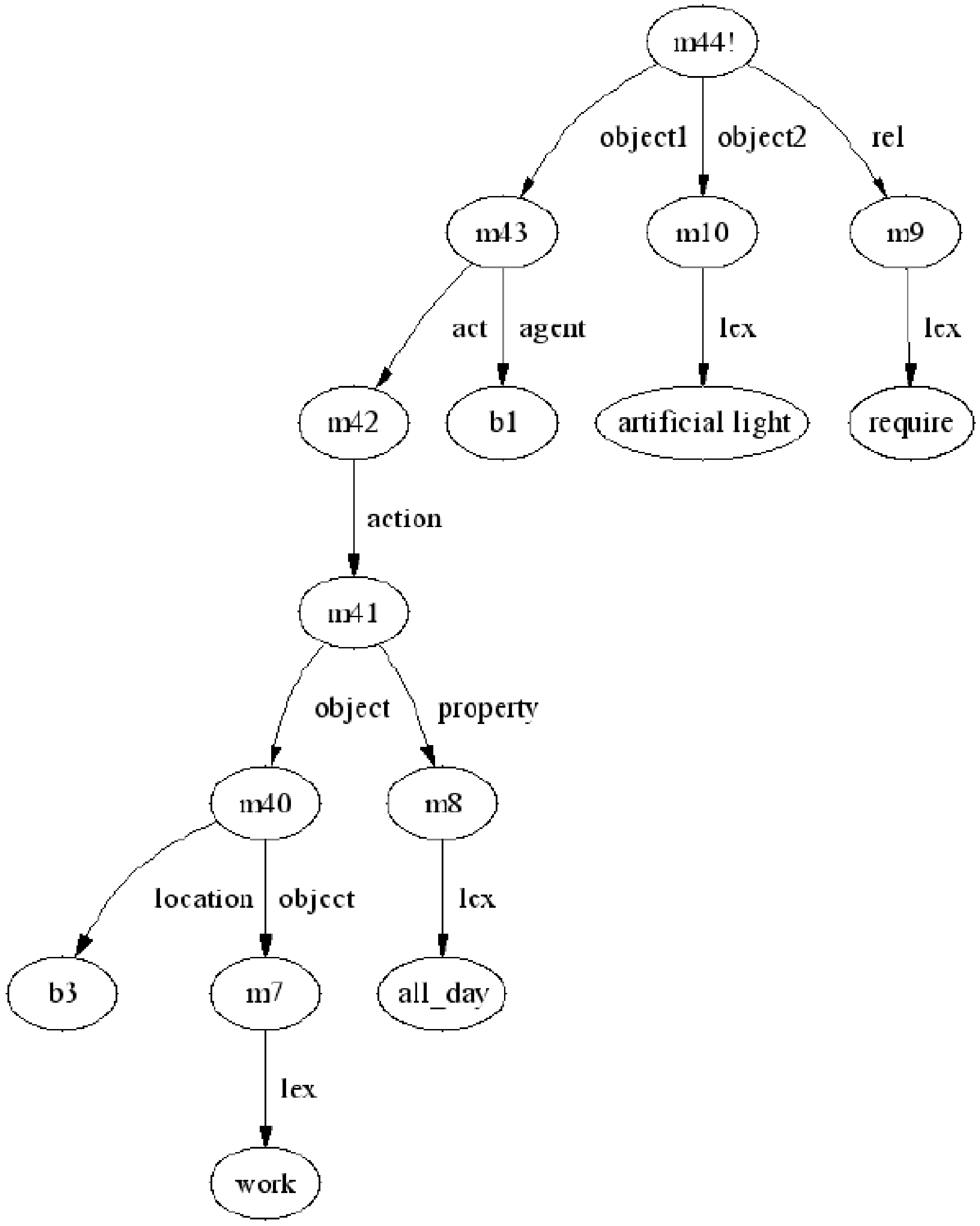


Fig 11. he require artificial like to work at home all day.

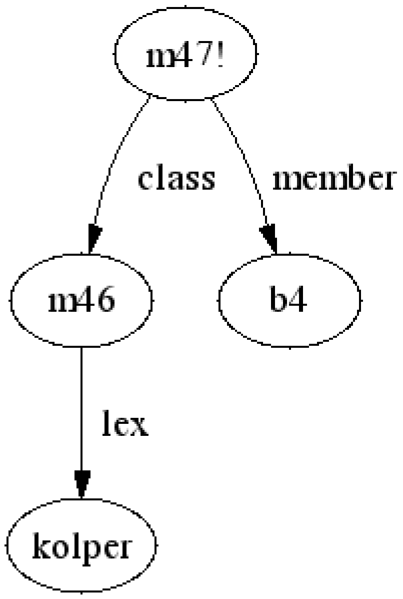


Fig 12. b4 is a kolper

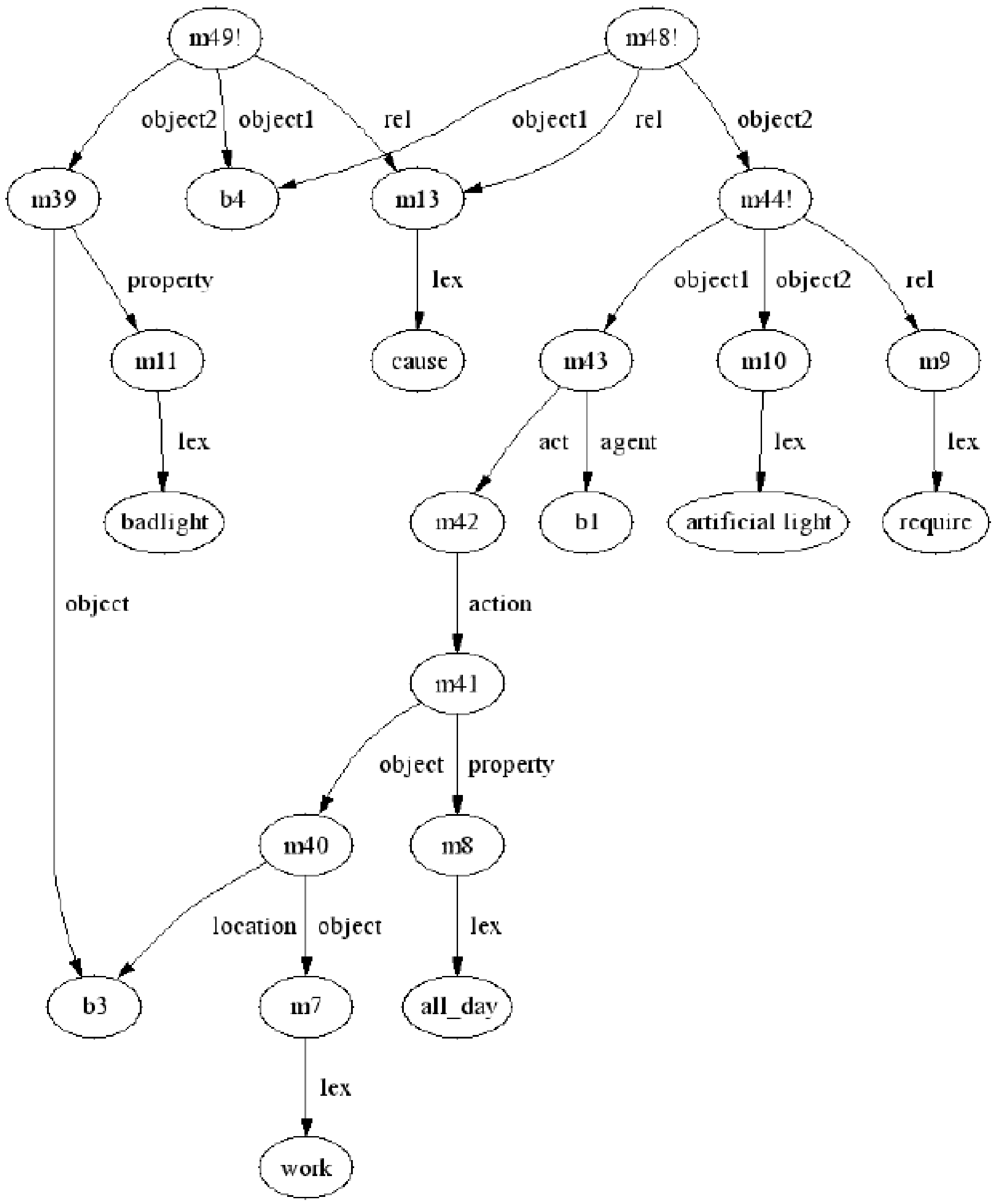


Fig 13. Kolper causes that b1 requires to work by artificial light all day at b4 and b4 having bad natural illumination

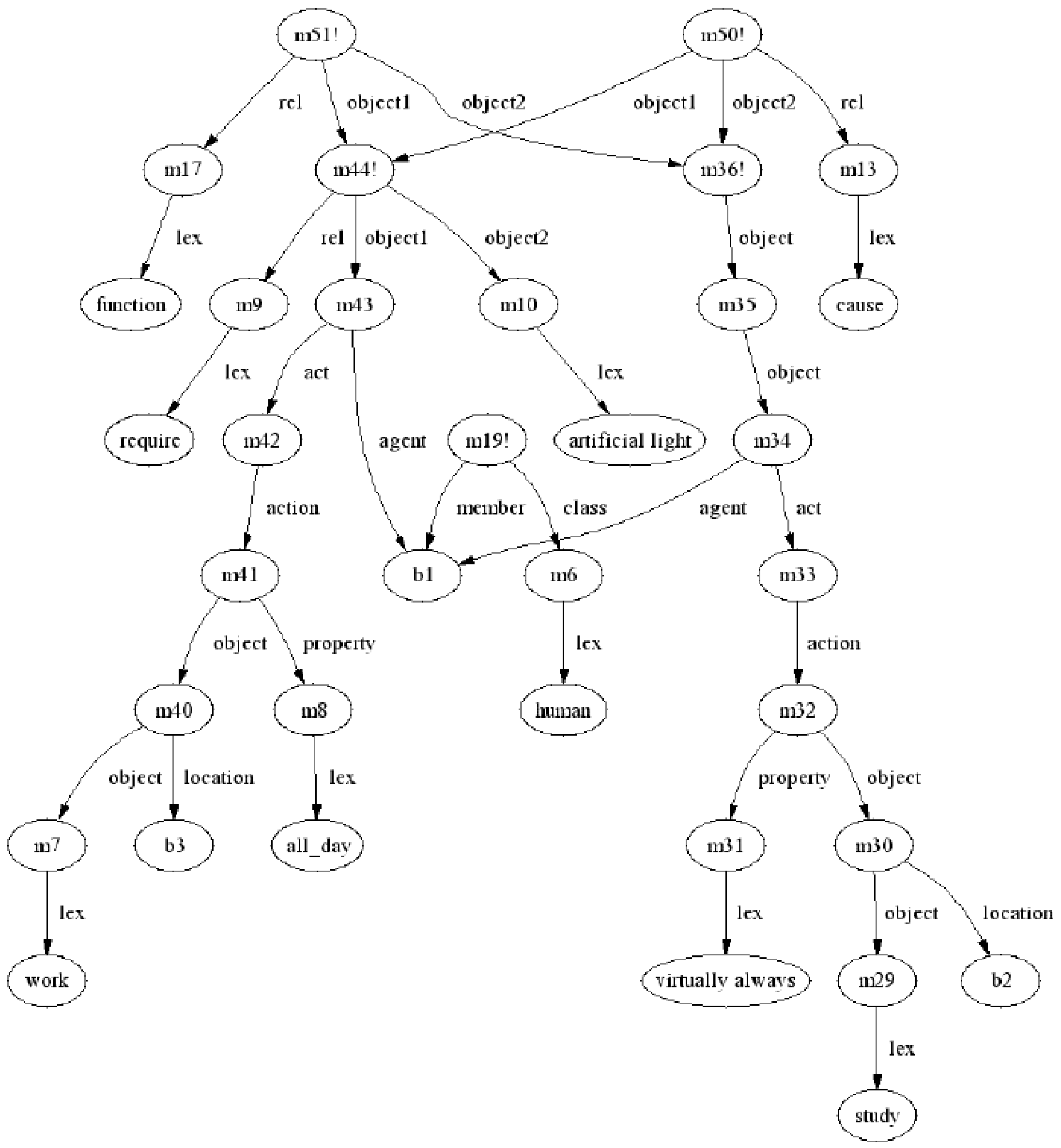


Fig 14. The whole sentence+some inferences

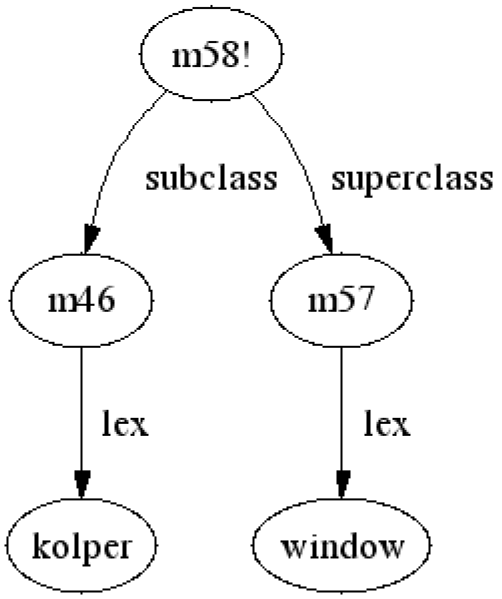


Fig 15. kolper is a window

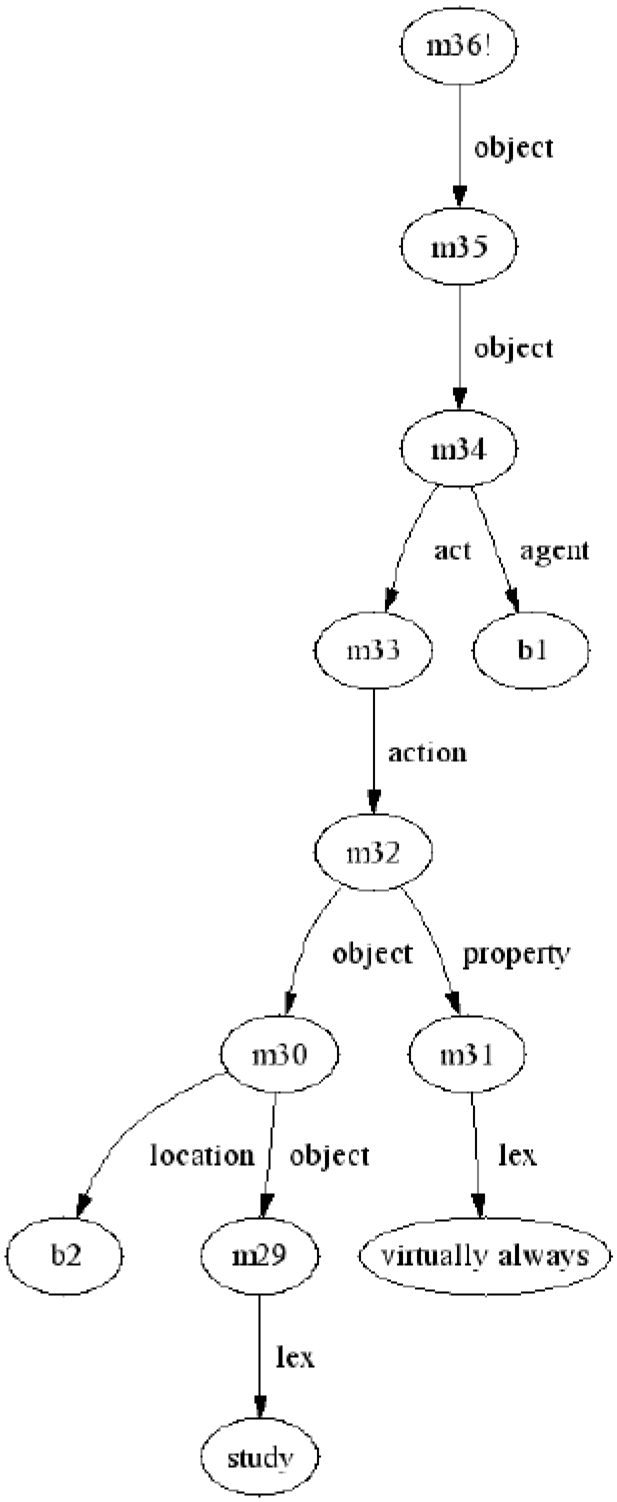


Fig 16. someone virtually always study at the library

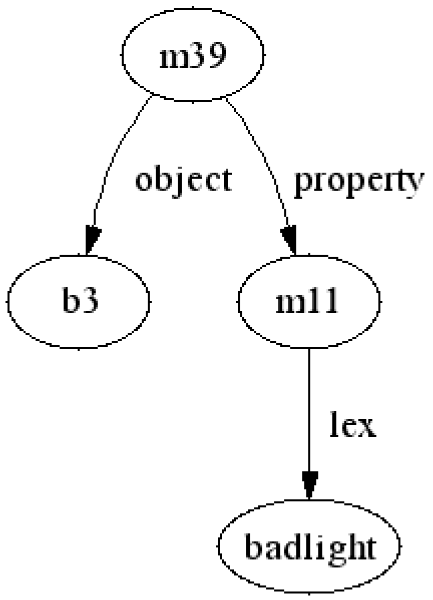


Fig 17. a place has bad natural illumination

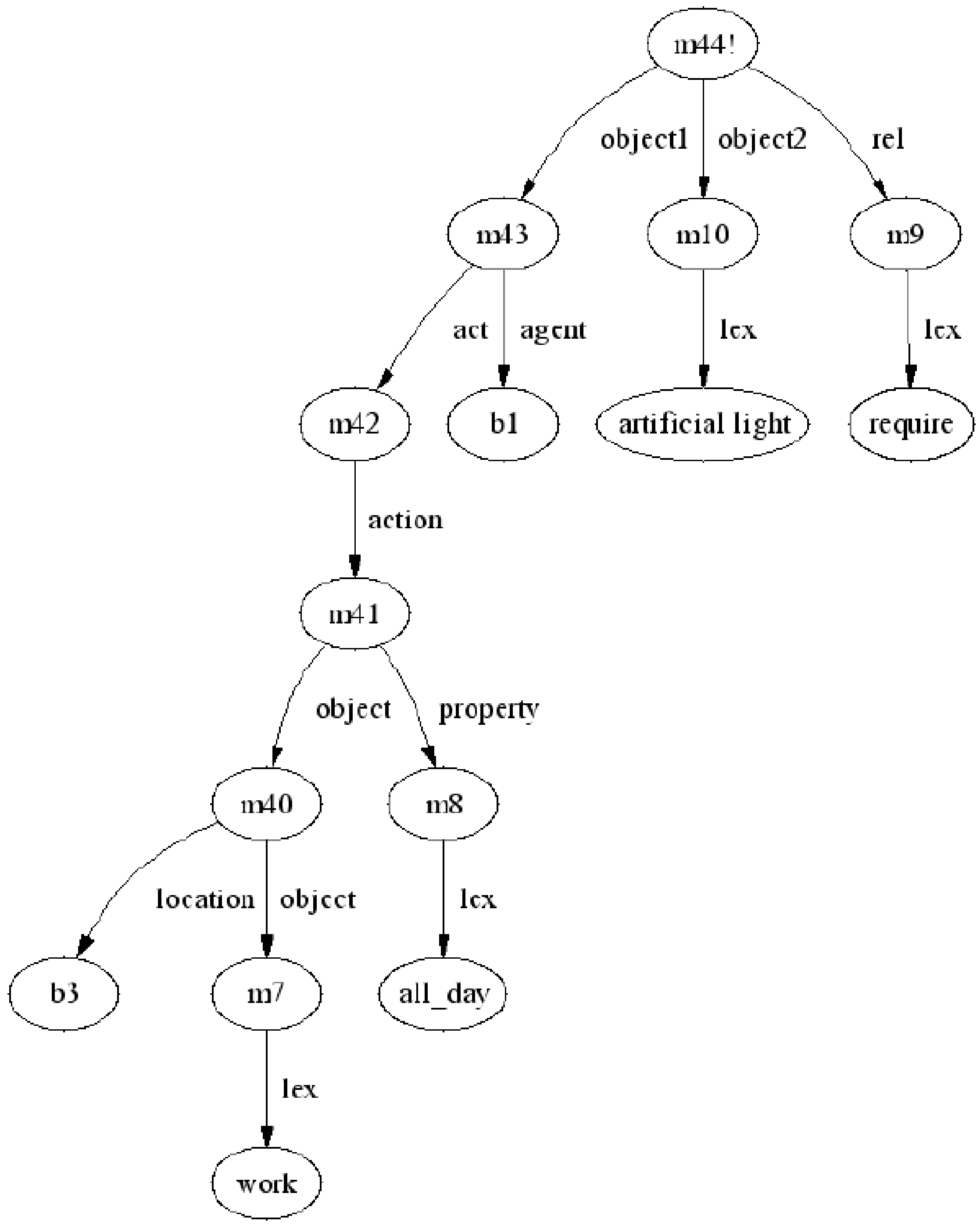


Fig 18. Someone needs to work by artificial light all day at home.