# Contextual Vocabulary Acquisition
# An Attempt: To derive the meaning of word "Impasse" from the context

**CSE720: Seminar on Contextual Vocabulary Acquisition**
**FALL 2004**

**Rashmi Mudiyanur**
**[rm64@buffalo.edu]**

## Abstract

This paper presents a methodology attempted to derive the meaning of the word "Impasse" using the **C**ontextual **V**ocabulary **A**cquisition Concept.

As part of this project, I provide sufficient background knowledge to Cassie so that she is able to derive the meaning of the word occurring in the context. The rules and the context with the new word are represented through SNePS and Cassie is asked to infer meaning.

## Introduction

What is Contextual Vocabulary Acquisition?

**C**ontextual **V**ocabulary **A**cquisition is the active, deliberate acquisition of word meanings from text by reasoning from contextual cues, background knowledge, and hypotheses developed from prior encounters with the word, but without external sources of help such as dictionaries or people. The ultimate goal is not merely to improve vocabulary acquisition, but also to increase students' reading comprehension of science, technology, engineering, and mathematics (STEM) texts, thereby leading to increased learning, by using a "miniature" (but real) example of the scientific method, viz., CVA. Definition Algorithms have been developed to infer the meanings of unknown nouns, verbs and adjectives. Refer to CVA Web Site for more information.

The word that is being used for this experiment is “Impasse”. The context I have chosen is: All chances for agreement were now gone, and compromise would now be impossible; in short, an **impasse** had been reached. This sentence is a typical example of sentences having direct description word in them. The word *in short*, should tell the reader that the consequent of this word is logically equivalent to whatever precedes this word. Also, the context is a very good example of what we call the cloze passages, which is defined as passages containing words with missing meanings, not to be mistaken to passages containing words with missing concepts. To model human cognition, this is a very good example and this motivated met to choose this word. I represented all the information that Cassie should know about ‘chances for agreement’, ‘compromise’ and ‘disagreement’ concepts, what we call background knowledge along with the actual context.

Apparently, I was unable to get the results of this experiment. The experiment result is to get the meaning of the word ‘Impasse’ as Disagreement. However, the work still remains to be incomplete and certain modifications will need to be done to get this working. I have described the case frames that I have used and the background knowledge that I provide in the future sections of this paper.

## Implementation

This task involved identifying the right background knowledge that has to be provided to Cassie so that she will be able to use the same in inferring the meaning of the difficult word. The next step is to choose the right case frames from the set of standard case frames already in SNePS. With the help of the information obtained from the above two processes, we need to represent the prior knowledge in SNePS. Representing the context, applying the noun algorithm and asking cassie as to define the word constitutes the next steps to be followed.

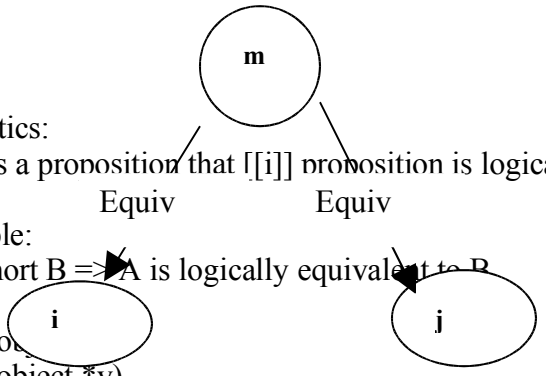
Verbal Protocols

The main intent behind performing verbal protocol is to be able to prudently select the background knowledge that needs to be provided to cassie. Since, CVA is modeled on the basis of human cognition, we experimented this with a couple of subjects. Each subject was given the context with the word ‘Impasse’ replaced with ‘Ontive’ as Impasse is too common a word to be called *Unknown.* They were asked to think aloud while they were using the inferring process. Fortunately, all the subjects were able to infer the meaning as *disagreement*. Based on what prior world knowledge they used, I was able to decide on what knowledge to provide to the system as background.

Case Frames

In addition to the standard case frames in SNePS and the ones mentioned in the CVA website, I used the equiv-equiv relationship that represents the logical equivalence of two objects.

Syntax:




Semantics:
[[m]] is a proposition that [[i]] proposition is logically equivalent to [[j]] proposition/
Equiv Equiv
Example:
A in short B => A is logically equivalent to B
(build o
(build object *y)
(build equiv *x equiv *y)

## Background Knowledge

Based on the information that I obtained from my subjects who participated in the verbal protocols, I was able to arrive at the following rules as background knowledge.

Since, the first sentence says that all chances for agreement were gone, subjects understood that this means that disagreement has been reached.

RULE 1: *If all chances for agreement are gone, then Disagreement has been reached.*

Subjects understood that if compromise is not possible then, agreement is not possible.

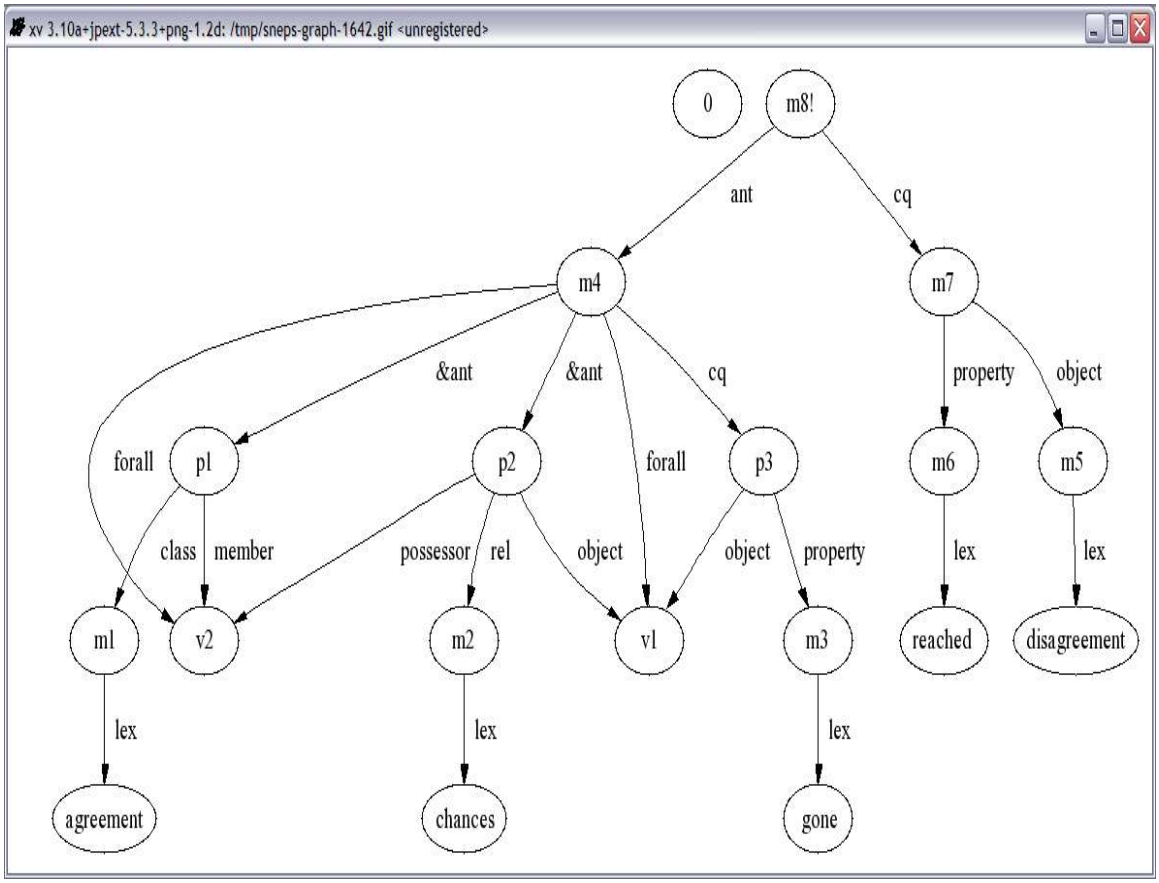
RULE 2: *If compromise is impossible, then Disagreement has been reached.*

We also introduce a new rule to say that if X is reached, Y is reached and X is Unknown, then Y is X – this is to tell Cassie to derive the meaning of the unknown word.

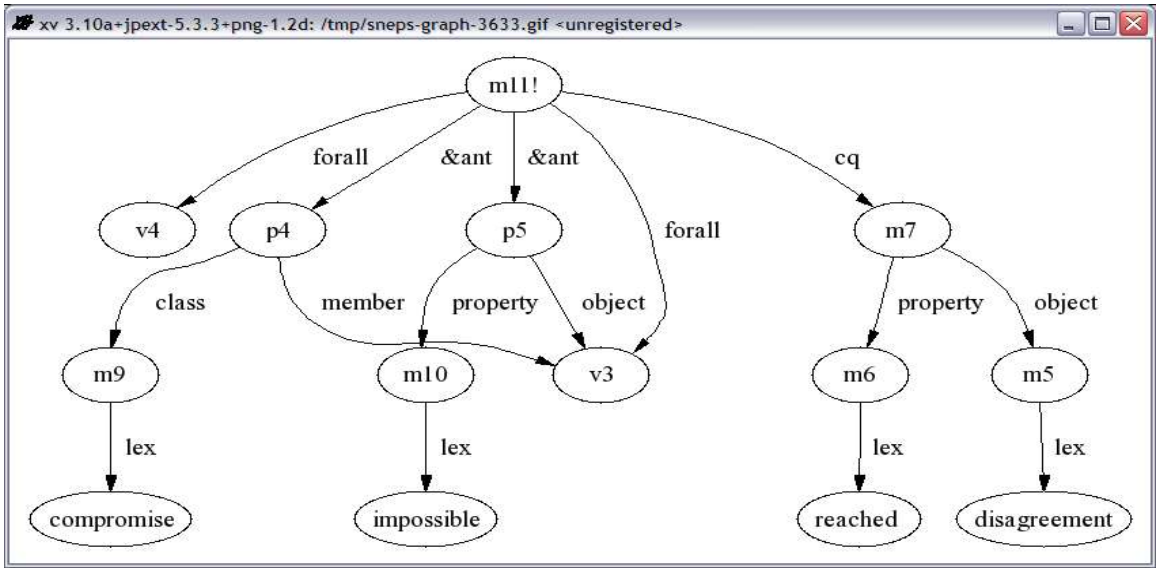
RULE 3: *If X is reached and Y is reached and Y is unknown, then Y is X.*

## SNePS Representation of Background Knowledge

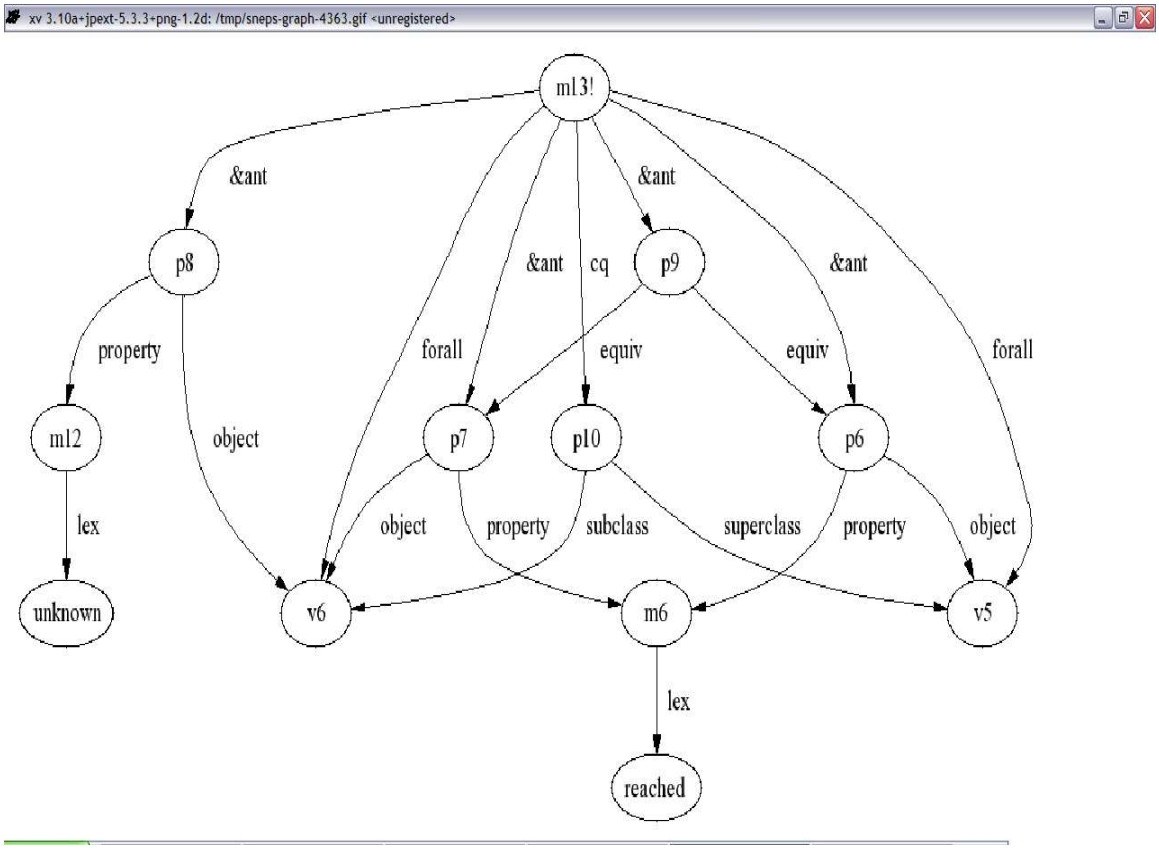
After cogent selection, analysis and a couple of attempts, the background knowledge was represented as follows:




RULE 1: *If all chances for agreement are gone, then Disagreement has been reached.*



RULE 2: *If compromise is impossible, then Disagreement has been reached.*



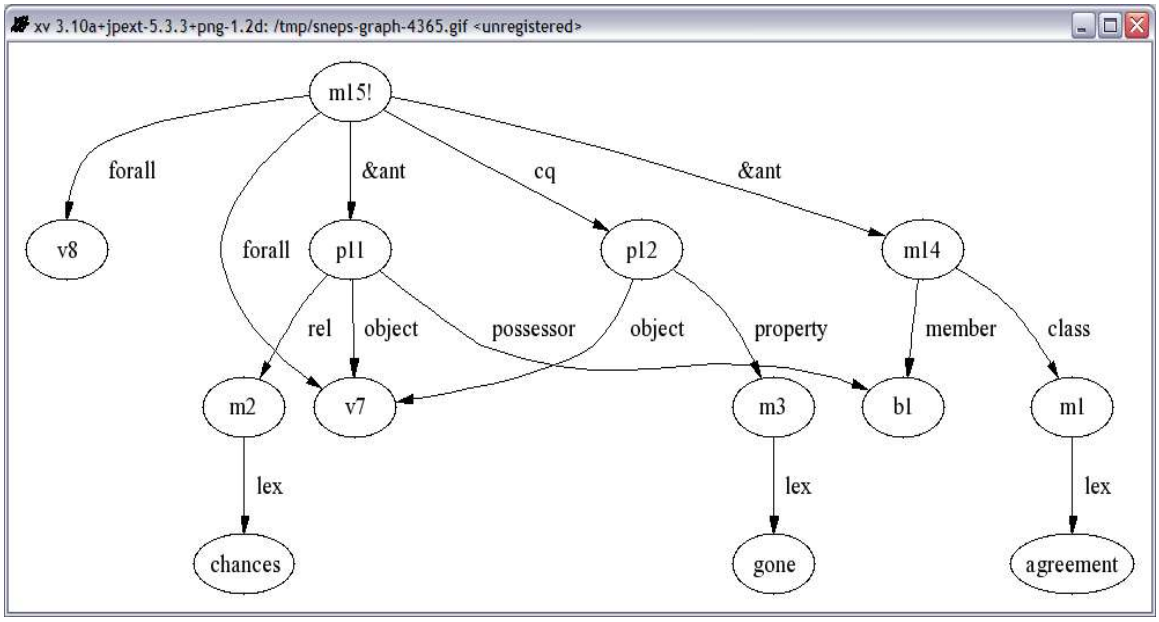

RULE 3: *If X is reached and Y is reached and Y is unknown, then Y is X.*

## The Context

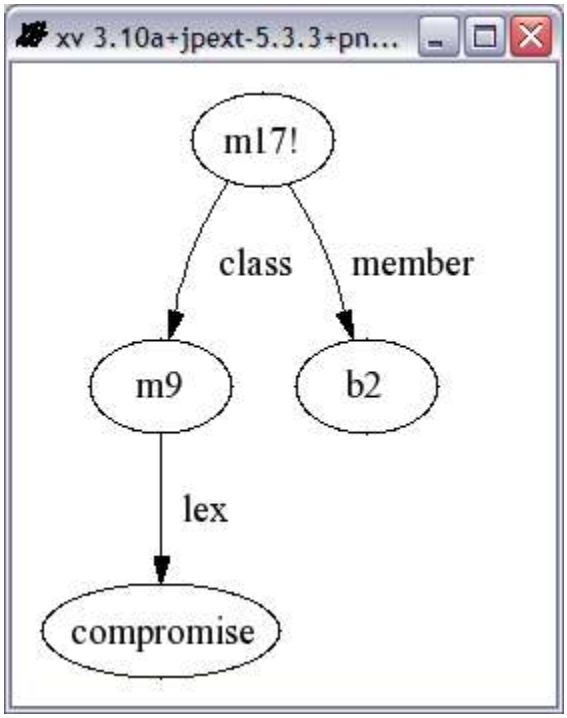
The context can be split into 3 statements. Each statement is connected by and or a symbol that implies ‘and’. Hence, we represent each statement separately and connect them using the thresh rule.

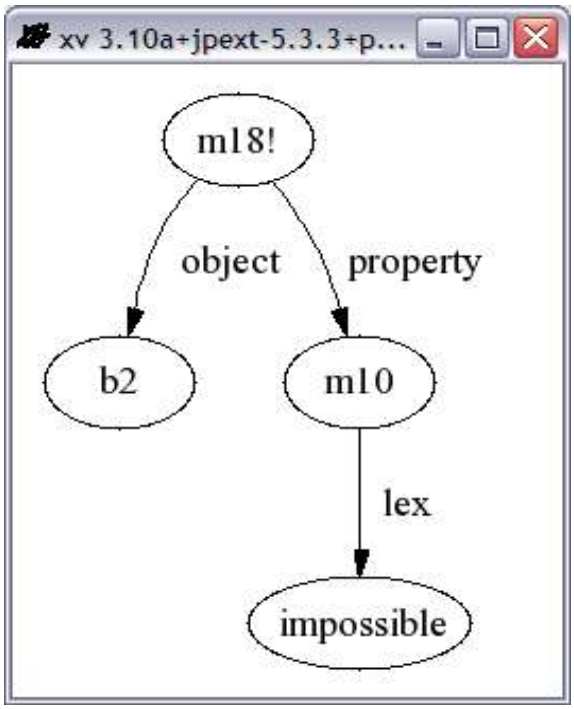
## SNePS Representation of the Context

*All chances for agreement are not gone*

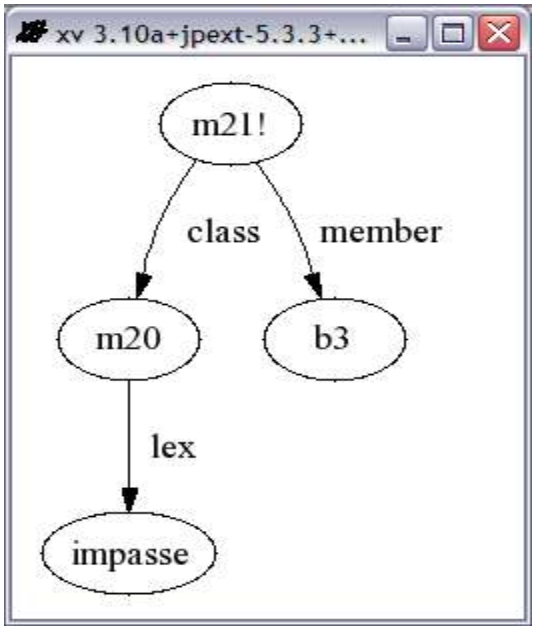


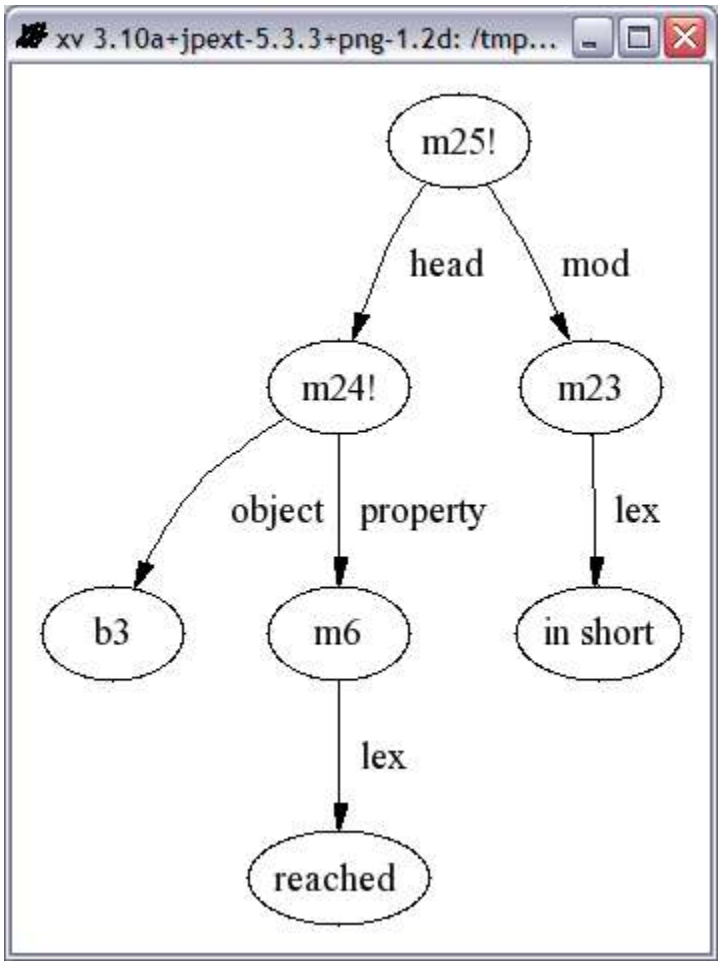

*Compromise is now impossible*



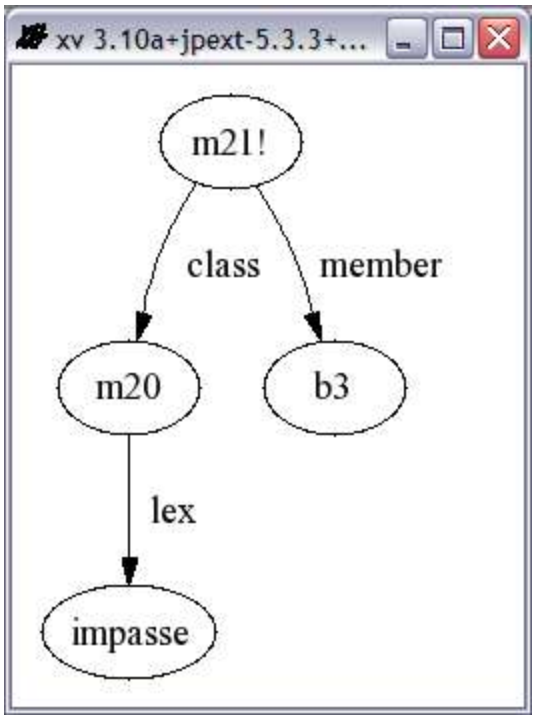



*In short, Impasse has been reached.*

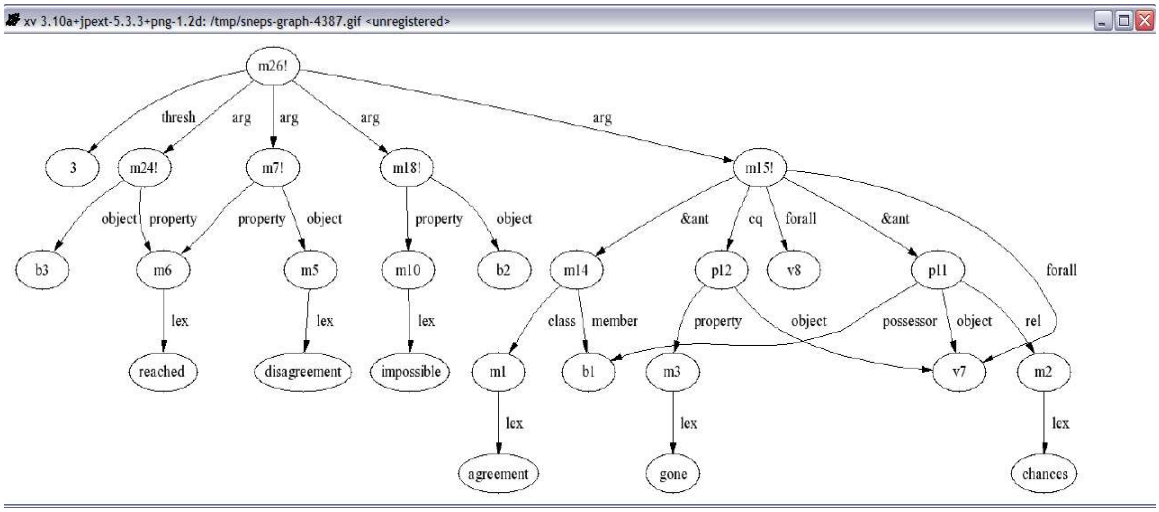





*Impasse is Unknown*




Final Representation




Noun Algorithm

Since Impasse is a noun in English language, we use the noun algorithm to derive the meaning of this word.

## Results

Appendix I has the entire code of this project including the background knowledge and the passage representation. Appendix II has a sample run of the demo program. I have tried with three different case frames to get the output. I am including the demo run with all the three different trials.

The following is the output of Cassie when asked to define the word 'Impasse':

```
--> defineNoun "impasse")
--- pause --
 Definition of impasse:
 Possible Properties: unknown, reached,
nil
```

The above result does not say much about the *unknown* word. However, if we keenly observe the final representation of the passage, we understand that cassie is deducing that disagreement has been reached. Perhaps, it is unable to correlate disagreement to impasse even after giving the rule that if x is reached and y is reached, then x is y.

## Immediate Steps

I have recognized the following steps that need to be immediately followed to get this word working:

1. Figure out why cassie is not correlating Impasse with Disagreement even after giving the RULE 3.
2. After Step 1 is successfully implemented, we need to see if cassie gives Disagreement for Class Inclusions of Impasse.

## Future Work

As you can see there are quite a large amount of research that can go into this programming project. Some of the long terms tasks that I suggest are:

1. One can try and implement the 'In short' phrase using the right case frame so that thresh case frame which is superfluous can be eliminated
2. One can modify the noun algorithm to make cassie look for equiv-equiv caseframes as well.

## References

1. William J. Rapaport, "Contextual Vocabulary Acquisition: from algorithm to

curriculum", http://www.cse.buffalo.edu/~rapaport/CVA/cva.html

2. "SNeRG, The SNePS Research Group", http://www.cse.buffalo.edu/sneps

3. "SNePS 2.6.1 User's Manual" [Postscript], Chs. 4, 7.3.

4. Dulin, Kenneth L. (1970), "Using Context Clues in Word Recognition and Comprehension", *The Reading Teacher* 23(5):440-445, 469.

# Appendix I

```
; ============================================================================
; FILENAME:     Impasse.demo
; DATE:         11-15-2004
; PROGRAMMER:       Rashmi Mudiyanur



;; 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 "Impasse.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)

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

; Clear the SNePS network:
(resetnet)

; OPTIONAL:
; UNCOMMENT THE FOLLOWING CODE TO TURN FULL FORWARD INFERENCING ON:
;
 ;enter the "snip" package:
^(in-package snip)
;
; ;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)
;
;re-enter the "sneps" package:
^(in-package sneps)

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

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

;defining the relations not found in the rels
(define isa mod head)

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


;RULE1: If all chances for agreement are gone, then Disagreement has been reached

(describe (assert
        ant (build forall ($x $A)
                &ant (build member *A
                        class (build lex "agreement") )
                &ant (build possessor *A
                        rel (build lex "chances")
                        object *x)
                cq (build object *x
                        property (build lex "gone")))
        cq (build object (build lex "disagreement")
                property (build lex "reached") ) ) )


;RULE2: If compromise is impossible then Disagreement has been reached.

(describe (assert forall ($x $y)
        &ant (build member *x
                class (build lex "compromise") )
        &ant (build object *x
             property (build lex "impossible") )
        cq (build object (build lex "disagreement")
           property (build lex "reached") ) ) )


;RULE3: If X is reached and Y is reached and Y is unknown, then Y is X.

(describe (assert forall ($x $y)
        &ant (build object *x
             property (build lex "reached") ) = r1
        &ant (build object *y
             property (build lex "reached") ) = r2
        &ant (build object *y
             property (build lex "unknown") )
        &ant (build equiv *r1 equiv *r2)
        cq (build subclass *y superclass *x) ) )



; CASSIE READS THE PASSAGE:
; ========================
; (put annotated SNePSUL code of the passage here)
 (describe (add forall ($x $A)
        &ant (build member #A class (build lex "agreement") )
        &ant (build possessor *A
             rel (build lex "chances")
```

```
                    object *x)
          cq (build object *x
              property (build lex "gone") ) ) = arg1)



(describe (add member #c class (build lex "compromise") ) )



(describe (add object *c
                        property (build lex "impossible") ) = arg2)



(describe (add member #I class (build lex "impasse") ) )



(describe (add object *I property (build lex "unknown") ) )



(describe (add
                mod (build lex "in short")
                head (add object *I
                        property (build lex "reached") ) = arg3) )



(describe (assert thresh 3
                        arg *arg1
                        arg *arg2
                        arg *arg3) )



; Ask Cassie what "Impasse" means:

; UNCOMMENT THE ONE YOU *DO* WANT
; AND DELETE THE OTHER!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 ^(defineNoun "impasse")
```

## Appendix II

I am including three runs of this demo. Each is explained below:

1. Equiv-Equiv run – I am using the equiv-equiv case frame to represent the ISA relationship.
2. Assert run – I am trying to assert all the three statements together at the end, while I am representing the context
3. Add run – I tried to add all the three statements together at the end, while I am representing the context.

Equiv-Equiv Run

```
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.

   12/18/2004 17:28:24

* (demo "Impasse1.demo")
File /home/csgrad/rm64/cse720/Impasse1.demo is now the source of input.
```

```
CPU time : 0.00

*
; ============================================================================
; FILENAME:       Impasse.demo
; DATE:                   11-15-2004
; PROGRAMMER:         Rashmi Mudiyanur
```

```
;; 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 "Impasse.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 : 0.17

* ; ^(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 : 0.01

*

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


 CPU time : 0.01

*
; 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

*
;defining the relations not found in the rels
(define isa mod head)

(isa mod head)

 CPU time : 0.01
```

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


;RULE1: If all chances for agreement are gone, then Disagreement has been reached

(describe (assert
                        ant (build forall ($x $A)
                                &ant (build member *A
                                        class (build lex "agreement") )
                                &ant (build possessor *A
                                        rel (build lex "chances")
                                        object *x)
                                cq (build object *x
                                        property (build lex "gone")))
                        cq (build object (build lex "disagreement")
                                property (build lex "reached") ) ) )

(m8!
 (ant
  (m4 (forall v2 v1)
   (&ant (p2 (object v1) (possessor v2) (rel (m2 (lex chances))))
    (p1 (class (m1 (lex agreement))) (member v2)))
   (cq (p3 (object v1) (property (m3 (lex gone)))))))
 (cq (m7 (object (m5 (lex disagreement))) (property (m6 (lex reached))))))

(m8!)

 CPU time : 0.00

*

;RULE2: If compromise is impossible then Disagreement has been reached.

(describe (assert forall ($x $y)
                        &ant (build member *x
                                class (build lex "compromise") )
                        &ant (build object *x
                  property (build lex "impossible") )
                        cq (build object (build lex "disagreement")
                 property (build lex "reached") ) ) )

(m11! (forall v4 v3)
 (&ant (p5 (object v3) (property (m10 (lex impossible))))
  (p4 (class (m9 (lex compromise))) (member v3)))
 (cq (m7 (object (m5 (lex disagreement))) (property (m6 (lex reached))))))

(m11!)

 CPU time : 0.01

*

;RULE3: If X is reached and Y is reached and Y is unknown, then Y is X.
```

```
(describe (assert forall ($x $y)
                    &ant (build object *x
                property (build lex "reached") ) = r1
                    &ant (build object *y
                property (build lex "reached") ) = r2
                    &ant (build object *y
                property (build lex "unknown") )
                    &ant (build equiv *r1 equiv *r2)
                    cq (build subclass *y superclass *x) ) )

(m13! (forall v6 v5)
 (&ant
  (p9
   (equiv (p7 (object v6) (property (m6 (lex reached))))
    (p6 (object v5) (property (m6)))))
  (p8 (object v6) (property (m12 (lex unknown)))) (p7) (p6))
 (cq (p10 (subclass v6) (superclass v5))))

(m13!)

 CPU time : 0.00

*


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


(describe (add forall ($x $A)
                    &ant (build member #A class (build lex "agreement") )
                    &ant (build possessor *A
              rel (build lex "chances")
                                object *x)
                    cq (build object *x
            property (build lex "gone") ) ) = arg1)

(m15! (forall v8 v7)
 (&ant (p11 (object v7) (possessor b1) (rel (m2 (lex chances))))
  (m14 (class (m1 (lex agreement))) (member b1)))
 (cq (p12 (object v7) (property (m3 (lex gone))))))

(m15!)

 CPU time : 0.01

*


(describe (add member #c class (build lex "compromise") ) )

(m17! (class (m9 (lex compromise))) (member b2))

(m17!)

 CPU time : 0.02
```

```
*


(describe (add object *c
                                property (build lex "impossible") ) = arg2)

(m18! (object b2) (property (m10 (lex impossible))))
(m7! (object (m5 (lex disagreement))) (property (m6 (lex reached))))

(m18! m7!)

 CPU time : 0.01

*


(describe (add member #I class (build lex "impasse") ) )

(m21! (class (m20 (lex impasse))) (member b3))

(m21!)

 CPU time : 0.01

*


(describe (add object *I property (build lex "unknown") ) )

(m22! (object b3) (property (m12 (lex unknown))))

(m22!)

 CPU time : 0.00

*


(describe (add
                        mod (build lex "in short")
                        head (add object *I
                                property (build lex "reached") ) = arg3) )

(m25! (head (m24! (object b3) (property (m6 (lex reached)))))
 (mod (m23 (lex in short))))

(m25!)

 CPU time : 0.01

*


(describe (assert thresh 3
                                arg *arg1
                                arg *arg2
                                arg *arg3) )
```

```
(m26! (thresh 3)
 (arg (m24! (object b3) (property (m6 (lex reached))))
  (m18! (object b2) (property (m10 (lex impossible))))
  (m15! (forall v8 v7)
   (&ant (p11 (object v7) (possessor b1) (rel (m2 (lex chances))))
    (m14 (class (m1 (lex agreement))) (member b1)))
   (cq (p12 (object v7) (property (m3 (lex gone))))))
  (m7! (object (m5 (lex disagreement))) (property (m6)))))

(m26!)

 CPU time : 0.00

*


; Ask Cassie what "Impasse" means:

; UNCOMMENT THE ONE YOU *DO* WANT
; AND DELETE THE OTHER!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 ^(
--> defineNoun "impasse")
 Definition of impasse:
 Possible Properties: unknown, reached,
nil



 CPU time : 0.05

*

End of /home/csgrad/rm64/cse720/Impasse1.demo demonstration.


 CPU time : 0.34

*
```

## Assert run

```
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.

   12/18/2004 17:55:37

* (demo "Impasse1.demo")

File /home/csgrad/rm64/cse720/Impasse1.demo is now the source of input.


 CPU time : 0.00

*
; ============================================================================
; FILENAME:       Impasse.demo
; DATE:                   11-15-2004
; PROGRAMMER:     Rashmi Mudiyanur



;; 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 "Impasse.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 : 0.17

* ; ^(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 : 0.01

*

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


 CPU time : 0.01

*
; 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 : 0.01

*
;defining the relations not found in the rels
(define isa mod head)

(isa mod head)

 CPU time : 0.00

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


;RULE1: If all chances for agreement are gone, then Disagreement has been reached

(describe (assert
                        ant (build forall ($x $A)
                                &ant (build member *A
                                             class (build lex "agreement") )
                                &ant (build possessor *A
                                             rel (build lex "chances")
                                             object *x)
                                cq (build object *x
                                          property (build lex "gone")))
                        cq (build object (build lex "disagreement")
                               property (build lex "reached") ) ) )

(m8!
 (ant
  (m4 (forall v2 v1)
   (&ant (p2 (object v1) (possessor v2) (rel (m2 (lex chances))))
    (p1 (class (m1 (lex agreement))) (member v2)))
   (cq (p3 (object v1) (property (m3 (lex gone)))))))
 (cq (m7 (object (m5 (lex disagreement))) (property (m6 (lex reached))))))
```

```
(m8!)

 CPU time : 0.00

*

;RULE2: If compromise is impossible then Disagreement has been reached.

(describe (assert forall ($x $y)
                    &ant (build member *x
                              class (build lex "compromise") )
                    &ant (build object *x
                property (build lex "impossible") )
                    cq (build object (build lex "disagreement")
              property (build lex "reached") ) ) )

(m11! (forall v4 v3)
 (&ant (p5 (object v3) (property (m10 (lex impossible))))
  (p4 (class (m9 (lex compromise))) (member v3)))
 (cq (m7 (object (m5 (lex disagreement))) (property (m6 (lex reached))))))

(m11!)

 CPU time : 0.01

*

;RULE3: If X is reached and Y is reached and Y is unknown, then Y is X.

(describe (assert forall ($x $y)
                    &ant (build object *x
                property (build lex "reached") ) = r1
                    &ant (build object *y
                property (build lex "reached") ) = r2
                    &ant (build object *y
                property (build lex "unknown") )
                    &ant (build thresh 2
                              arg *r1 arg *r2)
                    cq (build subclass *y superclass *x) ) )

(m13! (forall v6 v5)
 (&ant
  (p9 (thresh 2)
   (arg (p7 (object v6) (property (m6 (lex reached))))
    (p6 (object v5) (property (m6)))))
  (p8 (object v6) (property (m12 (lex unknown)))) (p7) (p6))
 (cq (p10 (subclass v6) (superclass v5))))

(m13!)

 CPU time : 0.00

*


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

```
; (put annotated SNePSUL code of the passage here)


(describe (add forall ($x $A)
                    &ant (build member #A class (build lex "agreement") )
                    &ant (build possessor *A
                rel (build lex "chances")
                                object *x)
                    cq (build object *x
              property (build lex "gone") ) ) = arg1)

(m15! (forall v8 v7)
 (&ant (p11 (object v7) (possessor b1) (rel (m2 (lex chances))))
  (m14 (class (m1 (lex agreement))) (member b1)))
 (cq (p12 (object v7) (property (m3 (lex gone))))))

(m15!)

 CPU time : 0.01

*


(describe (add member #c class (build lex "compromise") ) )

(m17! (class (m9 (lex compromise))) (member b2))

(m17!)

 CPU time : 0.03

*


(describe (add object *c
                                property (build lex "impossible") ) = arg2)

(m18! (object b2) (property (m10 (lex impossible))))
(m7! (object (m5 (lex disagreement))) (property (m6 (lex reached))))

(m18! m7!)

 CPU time : 0.02

*


(describe (add member #I class (build lex "impasse") ) )

(m21! (class (m20 (lex impasse))) (member b3))

(m21!)

 CPU time : 0.00

*
```

```
(describe (add object *I property (build lex "unknown") ) )

(m22! (object b3) (property (m12 (lex unknown))))

(m22!)

 CPU time : 0.01

*


(describe (add
                        mod (build lex "in short")
                        head (add object *I
                                property (build lex "reached") ) = arg3) )

(m25! (head (m24! (object b3) (property (m6 (lex reached)))))
 (mod (m23 (lex in short))))

(m25!)

 CPU time : 0.01

*


(describe (assert thresh 3
                                arg *arg1
                                arg *arg2
                                arg *arg3) )

(m26! (thresh 3)
 (arg (m24! (object b3) (property (m6 (lex reached))))
  (m18! (object b2) (property (m10 (lex impossible))))
  (m15! (forall v8 v7)
   (&ant (p11 (object v7) (possessor b1) (rel (m2 (lex chances))))
    (m14 (class (m1 (lex agreement))) (member b1)))
   (cq (p12 (object v7) (property (m3 (lex gone))))))
  (m7! (object (m5 (lex disagreement))) (property (m6)))))

(m26!)

 CPU time : 0.00

*


; Ask Cassie what "Impasse" means:

; UNCOMMENT THE ONE YOU *DO* WANT
; AND DELETE THE OTHER!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 ^(
--> defineNoun "impasse")
 Definition of impasse:
 Possible Properties: unknown, reached,
nil
```

```
 CPU time : 0.05

*

End of /home/csgrad/rm64/cse720/Impasse1.demo demonstration.

 CPU time : 0.34

*
```

```
Add Run

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.

   12/18/2004 18:08:01

* (demo "Impasse1.demo")

File /home/csgrad/rm64/cse720/Impasse1.demo is now the source of input.


 CPU time : 0.00

*
; ============================================================================
; FILENAME:       Impasse.demo
; DATE:                   11-15-2004
; PROGRAMMER:     Rashmi Mudiyanur



;; 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 "Impasse.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 : 0.00

*
; 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 : 0.01

*

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


 CPU time : 0.01

*
; 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

*
;defining the relations not found in the rels
(define isa mod head)

(isa mod head)

 CPU time : 0.00

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


;RULE1: If all chances for agreement are gone, then Disagreement has been reached

(describe (assert
                        ant (build forall ($x $A)
                                &ant (build member *A
                                            class (build lex "agreement") )
                                &ant (build possessor *A
```

```
                                        rel (build lex "chances")
                                        object *x)
                                cq (build object *x
                                        property (build lex "gone")))
                        cq (build object (build lex "disagreement")
                                property (build lex "reached") ) ) )

(m8!
 (ant
  (m4 (forall v2 v1)
   (&ant (p2 (object v1) (possessor v2) (rel (m2 (lex chances))))
    (p1 (class (m1 (lex agreement))) (member v2)))
   (cq (p3 (object v1) (property (m3 (lex gone)))))))
 (cq (m7 (object (m5 (lex disagreement))) (property (m6 (lex reached))))))

(m8!)

 CPU time : 0.00

*

;RULE2: If compromise is impossible then Disagreement has been reached.

(describe (assert forall ($x $y)
                &ant (build member *x
                        class (build lex "compromise") )
                &ant (build object *x
          property (build lex "impossible") )
                cq (build object (build lex "disagreement")
        property (build lex "reached") ) ) )

(m11! (forall v4 v3)
 (&ant (p5 (object v3) (property (m10 (lex impossible))))
  (p4 (class (m9 (lex compromise))) (member v3)))
 (cq (m7 (object (m5 (lex disagreement))) (property (m6 (lex reached))))))

(m11!)

 CPU time : 0.00

*

;RULE3: If X is reached and Y is reached and Y is unknown, then Y is X.

(describe (assert forall ($x $y)
                &ant (build object *x
          property (build lex "reached") ) = r1
                &ant (build object *y
          property (build lex "reached") ) = r2
                &ant (build object *y
          property (build lex "unknown") )
                &ant (build thresh 2
                        arg *r1 arg *r2)
                cq (build subclass *y superclass *x) ) )

(m13! (forall v6 v5)
 (&ant
  (p9 (thresh 2)
```

```
  (arg (p7 (object v6) (property (m6 (lex reached))))
   (p6 (object v5) (property (m6)))))
 (p8 (object v6) (property (m12 (lex unknown)))) (p7) (p6))
 (cq (p10 (subclass v6) (superclass v5))))

(m13!)

 CPU time : 0.01

*


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


(describe (add forall ($x $A)
                    &ant (build member #A class (build lex "agreement") )
                    &ant (build possessor *A
              rel (build lex "chances")
                                object *x)
                    cq (build object *x
            property (build lex "gone") ) ) = arg1)

(m15! (forall v8 v7)
 (&ant (p11 (object v7) (possessor b1) (rel (m2 (lex chances))))
  (m14 (class (m1 (lex agreement))) (member b1)))
 (cq (p12 (object v7) (property (m3 (lex gone))))))

(m15!)

 CPU time : 0.01

*


(describe (add member #c class (build lex "compromise") ) )

(m17! (class (m9 (lex compromise))) (member b2))

(m17!)

 CPU time : 0.02

*


(describe (add object *c
                              property (build lex "impossible") ) = arg2)

(m18! (object b2) (property (m10 (lex impossible))))
(m7! (object (m5 (lex disagreement))) (property (m6 (lex reached))))

(m18! m7!)

 CPU time : 0.02
```

```
*


(describe (add member #I class (build lex "impasse") ) )

(m21! (class (m20 (lex impasse))) (member b3))

(m21!)

 CPU time : 0.01

*


(describe (add object *I property (build lex "unknown") ) )

(m22! (object b3) (property (m12 (lex unknown))))

(m22!)

 CPU time : 0.00

*


(describe (add
                        mod (build lex "in short")
                        head (add object *I
                                property (build lex "reached") ) = arg3) )

(m25! (head (m24! (object b3) (property (m6 (lex reached)))))
 (mod (m23 (lex in short))))

(m25!)

 CPU time : 0.00

*


(describe (add thresh 3
                                arg *arg1
                                arg *arg2
                                arg *arg3) )
Error: Attempt to take the cdr of 0 which is not listp.
  [condition type: simple-error]

Restart actions (select using :continue):
 0: Return to Top Level (an "abort" restart).
 1: Abort entirely from this process.

[changing package from "common-lisp-user" to "snepsul"]
[1] snepsul(3):
```