COMPUTATIONALLY DEFINING 'OAM' VIA CONTEXTUAL VOCABULARY ACQUISITION

A report by
Divakar Dev Singh

Under the guidance of
Dr. William J. Rapaport
Professor of Computer Science & Engineering
University at Buffalo
Buffalo, New york.
CSE 499: Independent Study on

Contextual Vocabulary Acquisition

And

Artificial Intelligence

divakars@buffalo.edu

May 5, 2011

Abstract

Contextual Vocabulary Acquisition Project (or CVA project) assumes the role of a person who encounters an unfamiliar and unknown word while reading. We spent a significant time reading, and often we come across words that are new and unknown to us, a typical example would be *chirotonsor (barber)*. To model this problem faced by humans we used the Semantics Network Processing System or SNePS. SNePS is a knowledge representation, reasoning and acting system. It comprises of the SNePS based agent called CASSIE, which uses a SNePS based knowledge base. CASSIE is then provided with the SNePS representation of the passage which contains the unknown word plus all the background knowledge which is asserted into the "mind" of CASSIE using propositional rules that dictate which specific CVA definition algorithm should be used. In our study we will be implementing the CVA noun definition algorithm for the unknown word "oam". The study consisted of three main parts namely:

1. Conducting verbal protocols i.e. conducting a public study of the unknown word by asking individual humans to read the context and use their background knowledge to (try and) infer the meaning of the word.
2. To use the verbal protocols and formulate rules and background knowledge that will be later asserted into CASSIE's mind.
3. Use the SNePSUL case-frames to represent the:
    a. Background Knowledge (BK)
    b. Rules of inference or asserted knowledge
    c. Textual Knowledge, the things that CASSIE will "read"

Our objective was to compare CASSIE's results with the ones we had collected i.e. verbal protocols. We concluded that the definition given by CASSIE was acceptable and close enough to the meaning of the unknown word inferred by human readers and the one found on the

Oxford Online English Dictionary. Future work and additional ideas were also discussed in the project duration.

1. The CVA project and SNePS

The CVA projects targets the readers of English language and those who teach i.e. Educational and Reading instructors. Contextual Vocabulary Acquisition project is a tool that helps learn the meaning of a word by analyzing the textual knowledge in the light of our own acquired background knowledge. As readers we come across certain word(s) that we are unfamiliar with and in most cases it is likely that we do not make an effort to research on the word, i.e. either look it up in a dictionary or even a web based dictionary. The human brain and the neural network inside it are very complicated when it comes to replicate it. This is the reason why we cannot create a computational system that can store the entire lexicon and all the relationships associated with it. An alternate way to create an "intelligent" computational system would be to program it in a way so that it can learn the new meaning my analyzing the text around it and using some previously acquired background knowledge at the same time. CVA provides this alternate way.

The following are the underlying steps that encompass the procedure of the project:

1. Representation of background knowledge that comprises of the facts, concepts and certain rules that will provide a foundation to understand the passage (excluding the unknown word).
2. Represent the entire text or passage in SNePSUL; this is called adding the information around the text, i.e. the textual knowledge.
3. Give this represented passage already equipped with the background knowledge, to the SNePSUL agent, CASSIE who is also our reader. CASSIE "reads" the passage and creates an image of the representation.
4. The last step is to ask CASSIE what the word means. This is essentially telling CASSIE to employ the appropriate word definition algorithm. If everything input is unbiased this attempt should yield the correct dictionary definition of the unknown word.

SNePS or Semantic Network Processing System is a computational tool and a favourite utility for our project. SNePS uses propositional logic to create new entities; these entities can be either concepts or elements of the background knowledge or the textual knowledge we will add in order for CASSIE to read the passage.

We code up the information we have using the SNePSUL to create a network of nodes and arcs, the nodes can/will represent the concepts and/or the ideas conveyed by the passage and the arcs interlinking them would describe the sort of link they have.

2. The Unknown word, the passage and the objectives of the project.

For the purpose of the project and keeping in mind the time duration of completion of the project we chose to select a word from the CVA website maintained and handled by our guide and instructor Dr. Rapaport (http://www.cse.buffalo.edu/~rapaport/cva.html -> http://www.cse.buffalo.edu/~rapaport/CVA/cvapassages.html). The word and passage selected was number 14 from the Nouns section.

The following is the passage with the unknown word.

"Two ill dressed people … sat around a fire where the common meal was almost ready. The mother … peered at her son through the ***oam*** *of the bubbling stew."*

For the purpose of the report we are including the meaning/definition of the word also:

Oam:
Steam, vapor, condensation; warm steamy air, heat haze; (also) an aroma of cooking. (http://oed.com/view/Entry/246418?redirectedFrom=oam - eid)

For simplicity we decided to trim the sentence to a rather simpler version:

"*The mother peered through the* ***oam*** *of the blubbing stew.*"

Our overall task in the project would be to define the unknown word using the above sentence as the text. CASSIE would read the sentence and based on the background information we give it should hopefully tell us the meaning of the noun *oam.*

3. Verbal Protocols and coding the representation

As a part of the project we were suppose to collect data from people. A random sample of the university population was selected to be a part of this survey. The subjects were provided with the original sentence and were asked to not just glance over the sentence and come up with a meaning but to ponder and activate some background knowledge. They were asked to create mental imagery and associate it with the unknown word.

Subject 1 (Chris): "Fire gives creates an image and since something is cooking, that something must have some sort of steam or vapors coming out of it. Besides this something that bubbles and is being cooked has definitely some steam, thus oam is either steam or some sort of vapor."

Subject 2 (Thad): "Something that is being cooked and is bubbling and is transparent as it can be peered through means it is steam."

Subject 3(Mike): "Something that can be seen through and is hot and is bubbling means that it is steam or aroma."

Subject 4(Areea): "The origin seems to be unknown to me but it definitely means steam as it is a result of cooking."

4. Representation (Coded in SNePSUL, a LISP like syntax)

The following is the way we represented the entire project.

For CASSIE to better understand the concepts and ideas of the passage we had to make her assert some background knowledge,

```
; Load the appropriate definition algorithm:
^(load "/projects/rapaport/CVA/STN2/defun_noun.cl")
```

This part loads the predefined definition algorithms.

```
; Clear the SNePS network:

^(resetnet)
```

Clearing the SNePS network, getting rid of any previously defined nodes.

```
; OPTIONAL:
; UNCOMMENT THE FOLLOWING CODE TO TURN FULL
FORWARD INFERENCING ON:
;
```

```
; ;enter the "snip" package:
^(in-package snip)
;
We did not turn the full forward inference
; ;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:
; NB: If "intext" causes a "nil not of expected type" error,
; then comment-out the "intext" command and then
;uncomment & use the load command below, instead
^(load "/projects/rapaport/CVA/STN2/demos/rels")
(define Skf)
```

We had to define a skolem function in order to define one of our rules that used an existential quantifier.

```
;^(intext "/projects/rapaport/CVA/STN2/demos/rels")
;^load all pre-defined path definitions:
^(intext "/projects/rapaport/CVA/mkb3.CVA/paths/paths")
```

This loaded all the paths and relationships arcs that would be used to recursively say things like a man is a subclass of a living being, etc.

Before we can make our computational reader read the sentence we have to equip it with some background knowledge, rules that should be triggered when CASSIE read the passage.

```
; ====================
; BACKGROUND KNOWLEDGE:
; ====================
;Rule 1: Steam is transparent
;
;For all x and y [if x is steam of y -> x is transparent]
```

This is the first rule that says that if something has steam then the steam is transparent.

```
(describe (assert forall ($x $y)
                ant(build object *x rel (build lex "steam") possessor
                *y)
```

```
        cq (build object *x property (build lex
        "transparent"))))

;Rule 2:
;The rule in general states that there is a stew that has some oam
and that oam has the properties of
;steam i.e. being transparent. But the rule is quiet general and
says that anything that is the steam of something
;has the property of being transparent.
;
;x "has steam" z ( pos rel obj)
;z has prop transparent
;x "has oam" v
;v is transparent
;the property being transparent is w
;u is the oam and is unknown.

(describe (assert forall ($x $y $z $u $v $w)
    &ant (build possessor *x rel *y object *z)
    &ant (build object *z property *w)
    &ant (build possessor *x rel *u object *v)
    &ant (build object *v property *w)
    &ant (build object *u property (build lex
                                    "unknown"))
    cq (build superclass *y subclass *u)))
```

This is the biggest rule on our representation. It consists of five major antecedents and one consequence. To simply the rule we are trying to say the following. Some thing -possibly the stew- (x) has some steam (z), Steam as we know is transparent or at least translucent (w). The object we defined as (x) is in possession of something else called oam (v). From the original sentence we know that the stew has oam and the mother is peering through the oam of the stew we can say that the object oam (v) is also transparent. Also we created another node that represents the concept of being unknown (u). This is a general rule that says that if something possesses steam then that steam is transparent and if the possessor has some unknown that has the property of being transparent then in out context it would be an object that is the sub class of supper class steam (the consequent).

```
;Rule 3: Stew has steam
;this is where I have to use the skolem function as we have
encountered an existential quantifier.

(describe (assert forall $x ant (build member *x
          class (build lex "stew"))
```

```
	cq (build possessor *x rel (build lex "steam") object (build
Skf steamof a1 *x))))
```

Here we say that there exists some stew that has some steam. Since SNePS does not allow to define an existential quantifier, we use a Skolem function. More information on a skolem function can be found in the SNePS User Manual 3.2.1 p23 (http://www.cse.buffalo.edu/sneps/Manuals/manual271.pdf).

Next we go on and give our computational reader a sentence to read. For the simplicity of the project and keeping in mind the time frame we had on our hands we decided to leave intricate details like

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

;There is a stew that is being cooked by the mother.
;(cooking not represented as it not essential at this stage where
we just want the definition of oam)
;The unknown in this context is the word oam.
;The stew has some oam.
;The oam has the property of being transparent.
```

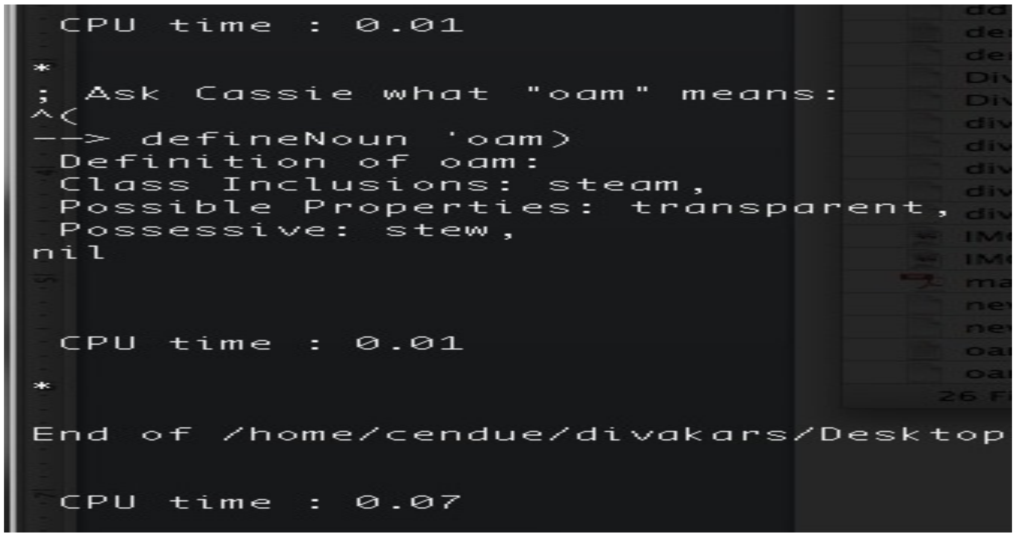
```
; The stew has some oam.

(describe (add member #mothersstew class (build lex
                "stew")))

(describe (add member #oamofstew class (build lex "oam")))

(describe (add object (build lex "oam") property (build lex
"unknown")))

(describe (add object *oamofstew property (build lex
"transparent")))



(describe (add object *oamofstew possessor *motersstew  rel
(lex "oam")))

(describe (add object *oamofstew rel (build lex oam) possessor
 *mothersstew))
```

After asserting all the background knowledge and adding all the textual knowledge we asked CASSIE what the word means, this was essentially to ask her to define the word.

```
; Ask Cassie what "oam" means:
```

```
^(defineNoun 'oam)
```

After this command we were given the following output:



```
 CPU time : 0.01

*
; Ask Cassie what "oam" means:
^(
--> defineNoun 'oam)
 Definition of oam:
 Class Inclusions: steam,
 Possible Properties: transparent,
 Possessive: stew,
nil


 CPU time : 0.01

*

End of /home/cendue/divakars/Desktop

 CPU time : 0.07
```

Based on the output, we say that CASSIE was able to determine the definition of the word very close to the definition we have concluded from our verbal protocols.

5. Syntax and semantics of the SNePS case frames.

The case frames used in our project are:

1. Object/Property.
2. Lex
3. Object/Rel/Possessor
4. Member/Class
5. Superclass/Subclass

   We also used the Skolem function.

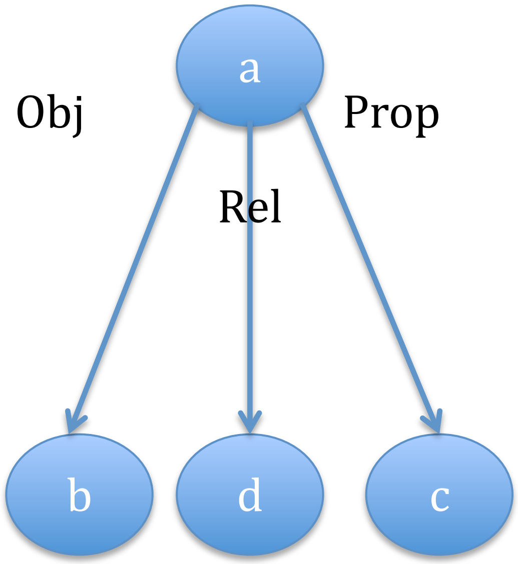
The syntax of the case frames is as following :

1. lex : a is the concept expressed using b

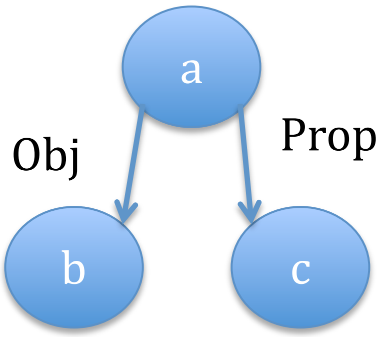
   eg. A concept expressed by a word in English is bubbling.
2. Object/Property: a is a proposition that b has a property c.
3. Object/Rel/Possessor: a is a proposition that b is a c of d.
4. Member/Class: a is proposition that b is a member of class c.
5. Superclass/Subclass: a is a proposition that b is a subclass of superclass c.

The Following are the schematic/visual representation of the above case frames:

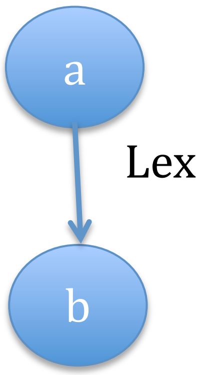
1. Object/Rel/Prop



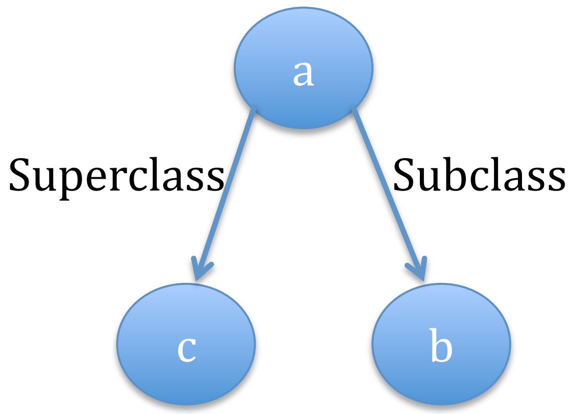
2. Object/Property



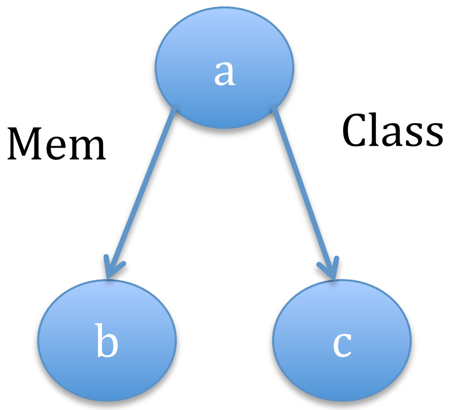
3. Lex



4. Superclass/Subclass



5. Member/Class



6. Syntax of case frames

1. Lex express the concept in double quotes

   ```
   (build lex “hot”)
   ```

2. Member/Class

   ```
   (build member (build lex “hot soup”) class (build lex
   “soup”))
   ```

3. Superclass/Subclass

   ```
   (assert superclass (build lex “greek”) subclass (build lex
   “theta”))
   ```

4. Object/Rel/Possessor

   ```
   (assert object (build lex “Ford”) possessor (build lex
   “Henry”) rel (build lex “car”))
   ```

5. Object/Property

   ```
   (assert object (build lex “bulb”) property (build lex “heat”))
   ```

7. Immediate steps.

The project that we undertook had to be constrained within the time limits of 15 weeks. But if we had another week or two at hand, out immediate next step would be to represent the entire sentence

"Two ill dressed people … sat around a fire where the common meal was almost ready. The mother … peered at her son through the ***oam*** *of the bubbling stew."*

The purpose would be to make more general rules about cooking, making stew or a dish in general, activate memories by knowing the way people sit, see or speak. This would help CASSIE "know" general background knowledge that could be used for defining other words in similar context. Our prime aim in this project would be to create a network that would be similar to the neural network that would enable CASSIE things that are not too obvious to a computational reader. Things like humans have skins, now a computer scientist would probably giggle at this but a computational reader like CASSIE would have no clue unless she was told about it. Developing a system that thinks and asserts like a human brain would.

8. Future Goals

The future goals of this project would be to include more contexts and assert background knowledge about the mother, the two people, the fact that they are ill dressed. Facts like the people are ill dressed and they are sitting around some thing (fire) in our case would help CASSIE know more concepts about the world. We should also look into making a case frames that would ease the way we represent our background knowledge textual knowledge.

The scope of this project it very vast and leads into the intricate details of artificial intelligence. During the project we learnt how linguistics and language could be integrated with computer science to create a new curriculum to teach students vocabulary. The results of this project are being employed at the broader basis to develop a curriculum for instructors of English language.

To conclude out report, through this independent study we learnt to implement our logical skills to think like a computer and then program one to do so. We were given a glimpse of the field of Artificial

Intelligence and how we -Computer Engineers and Scientists- can venture into it along side with linguists and philosophers.

"Our ultimate objective is to make programs that learn from their experience as effectively as humans do. We shall...say that a program has common sense if it automatically deduces for itself a sufficient wide class of immediate consequences of anything it is told and what it already knows."

JOHN MCCARTHY, "Programs with Common Sense", 1958

APPENDIX A: Script of running demo

```
Script started on Wed 11 May 2011 11:44:17 PM EDT
timberlake {~/Desktop/CVA Spring 2011} > mlisp
International Allegro CL Enterprise Edition
8.2 [Linux (x86)] (Jul 9, 2010 16:05)
Copyright (C) 1985-2010, Franz Inc., Oakland, CA, USA.  All Rights
Reserved.

This development copy of Allegro CL is licensed to:
   [4549] University at Buffalo

;; 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
;;; Installing streamc patch, version 2.
Loading system SNePS...10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
SNePS-2.7 [PL:2 2011/04/19 17:07:58] loaded.
Type `(sneps)' or `(snepslog)' to get started.
cl-user(2): (sneps)


   Welcome to SNePS-2.7 [PL:2 2011/04/19 17:07:58]

Copyright (C) 1984--2010 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.

   5/11/2011 23:45:06

* (demo "oamDEMOfile.txt")

File /home/cendue/divakars/Desktop/CVA Spring 2011/oamDEMOfile.txt
is now the source of input.


 CPU time : 0.01

* ;
=====================================================================
===
; FILENAME: oamDEMOfile.txt
; DATE:           20-APR-2011
; PROGRAMMER:     DIVAKAR_DEV_SINGH
```

```
;; this template version:   snepsul-template.demo-20061005.txt

; Lines beginning with a semi-colon are comments.
; Lines beginning with "^" are Lisp commands.
; All other lines are SNePSUL commands.
;
; To use this file: run SNePS; at the SNePS prompt (*), type:
;
;     (demo "oamDEMOfile.txt" :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:


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



 CPU time : 0.05

*

; 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)

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



 CPU time : 0.00

*
; load all pre-defined relations:
; NB: If "intext" causes a "nil not of expected type" error,
;     then comment-out the "intext" command and then
;          uncomment & use the load command below, instead
^(
--> load "/projects/rapaport/CVA/STN2/demos/rels")
; Loading /projects/rapaport/CVA/STN2/demos/rels
t



 CPU time : 0.00

* (define Skf)

(Skf)

 CPU time : 0.00

* ;^(intext "/projects/rapaport/CVA/STN2/demos/rels")

;^load all pre-defined path definitions:
^(
--> intext "/projects/rapaport/CVA/mkb3.CVA/paths/paths")
Loading file /projects/rapaport/CVA/mkb3.CVA/paths/paths.
before implied by the path (compose before
                            (kstar (compose after- ! before)))
before- implied by the path (compose (kstar (compose before- !
after))
                             before-)
after implied by the path (compose after
                           (kstar (compose before- ! after)))
```

```
after- implied by the path (compose (kstar (compose after- !
before))
                            after-)
sub1 implied by the path (compose object1- superclass- ! subclass
                          superclass- ! subclass)
sub1- implied by the path (compose subclass- ! superclass subclass-
!
                           superclass object1)
super1 implied by the path (compose superclass subclass- !
superclass
                            object1- ! object2)
super1- implied by the path (compose object2- ! object1 superclass-
!
                             subclass superclass-)
superclass implied by the path (or superclass super1)
superclass- implied by the path (or superclass- super1-)




 CPU time : 0.00

*
; BACKGROUND KNOWLEDGE:
; =====================
;Rule 1: Steam is transparent
;
;For all x and y [if x is steam of y -> x is transparent]

(describe (assert forall ($x $y)
           ant(build object *x rel (build lex "steam") possessor *y)
           cq (build object *x property (build lex "transparent"))))

(m3! (forall v2 v1)
 (ant (p1 (object v1) (possessor v2) (rel (m1 (lex steam)))))
 (cq (p2 (object v1) (property (m2 (lex transparent))))))

(m3!)

 CPU time : 0.00

* ;Rule 2:
;The rule in general states that there is a stew that has some oam
and that oam has the properties of
;steam i.e. being transparent. But the rule is quiet general and
says that anything that is the steam of something
;has the property of being transparent.
;
;x "has steam" z ( pos rel obj)
;z has prop transparent
;x "has oam" v
;v is transparent
;u in oam and is unknown.

(describe (assert forall ($x $y $z $u $v $w)
      &ant (build possessor *x rel *y object *z)
      &ant (build object *z property *w)
      &ant (build possessor *x rel *u object *v)
      &ant (build object *v property *w)
      &ant (build object *u property (build lex "unknown"))
      cq (build superclass *y subclass *u)))

(m5! (forall v8 v7 v6 v5 v4 v3)
 (&ant (p7 (object v6) (property (m4 (lex unknown))))
  (p6 (object v7) (property v8))
  (p5 (object v7) (possessor v3) (rel v6))
  (p4 (object v5) (property v8))
  (p3 (object v5) (possessor v3) (rel v4)))
 (cq (p8 (subclass v6) (superclass v4))))

(m5!)

 CPU time : 0.00

*

;Rule 3: Stew has steam
;this is where I have to use the skolem function as we have
encountered an existential quantifier.
;the rule says forall



(describe (assert forall $x ant (build member *x class (build lex
"stew"))
      cq (build possessor *x rel (build lex "steam") object (build
Skf steamof a1 *x))))

(m7! (forall v9) (ant (p9 (class (m6 (lex stew))) (member v9)))
 (cq
  (p11 (object (p10 (Skf steamof) (a1 v9))) (possessor v9)
   (rel (m1 (lex steam))))))

(m7!)

 CPU time : 0.00

*



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

;There is a stew that is being cooked by the mother.( cooking not
represented )
;The unknown in this context is the word oam.
;The stew has some oam.
;The oam has the property of being transparent.
```

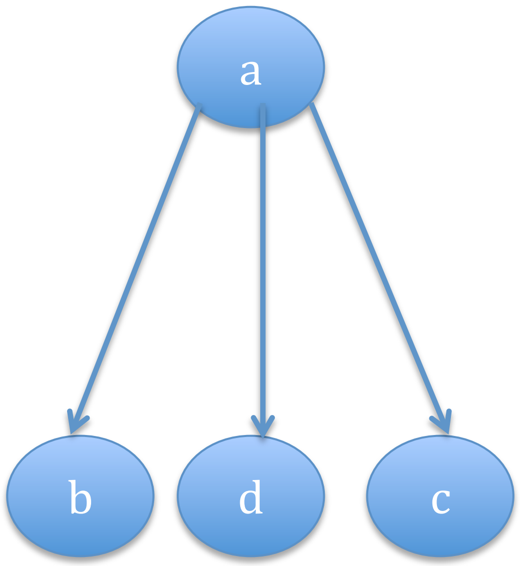
```
(describe (add member #mothersstew class (build lex "stew")))

(m11! (object (m9 (Skf steamof) (a1 b1)))
 (property (m2 (lex transparent))))
(m10! (object (m9)) (possessor b1) (rel (m1 (lex steam))))
(m8! (class (m6 (lex stew))) (member b1))

(m11! m10! m8!)

 CPU time : 0.01

* (describe (add member #oamofstew class (build lex "oam")))

(m17! (class (m16 (lex oam))) (member b2))

(m17!)

 CPU time : 0.00

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

(m18! (object (m16 (lex oam))) (property (m4 (lex unknown))))

(m18!)

 CPU time : 0.01

* (describe (add object *oamofstew property (build lex
"transparent")))

(m19! (object b2) (property (m2 (lex transparent))))

(m19!)

 CPU time : 0.00

* ;(describe (add object *mothersstew possessor *oamofstew rel
steamof))
(describe (add object *oamofstew possessor *motersstew  rel (lex
"oam")))

(m20! (object b2) (rel lex oam))

(m20!)

 CPU time : 0.00

* (describe
  (add object *oamofstew rel (build lex oam) possessor
*mothersstew))

(m22! (subclass (m16 (lex oam))) (superclass (m1 (lex steam))))
(m21! (object b2) (possessor b1) (rel (m16)))
```

```
(m22! m21!)

 CPU time : 0.00

*
;Describing all the nodes
(describe *nodes)

(m22! (subclass (m16 (lex oam))) (superclass (m1 (lex steam))))
(m21! (object b2) (possessor b1) (rel (m16)))
(m20! (object b2) (rel lex oam))
(m19! (object b2) (property (m2 (lex transparent))))
(m18! (object (m16)) (property (m4 (lex unknown))))
(m17! (class (m16)) (member b2))
(m11! (object (m9 (Skf steamof) (a1 b1))) (property (m2)))
(m10! (object (m9)) (possessor b1) (rel (m1)))
(m8! (class (m6 (lex stew))) (member b1))
(m7! (forall v9) (ant (p9 (class (m6)) (member v9)))
 (cq
  (p11 (object (p10 (Skf steamof) (a1 v9))) (possessor v9) (rel
(m1)))))
(m5! (forall v8 v7 v6 v5 v4 v3)
 (&ant (p7 (object v6) (property (m4))) (p6 (object v7) (property
v8))
  (p5 (object v7) (possessor v3) (rel v6))
  (p4 (object v5) (property v8))
  (p3 (object v5) (possessor v3) (rel v4)))
 (cq (p8 (subclass v6) (superclass v4))))
(m3! (forall v2 v1) (ant (p1 (object v1) (possessor v2) (rel (m1))))
 (cq (p2 (object v1) (property (m2)))))

(m22! m21! m20! lex m19! m18! m17! m16 oam b2 m11! m10! m9 m8! b1
m7!
 p11 p10 steamof p9 m6 stew v9 m5! p8 p7 m4 unknown p6 p5 p4 p3 v8
v7
 v6 v5 v4 v3 m3! p2 m2 transparent p1 m1 steam v2 v1)

 CPU time : 0.02

*
; Ask Cassie what "oam" means:
^(
--> defineNoun 'oam)
 Definition of oam:
 Class Inclusions: steam,
 Possible Properties: transparent,
 Possessive: stew,
nil



 CPU time : 0.01

*
```

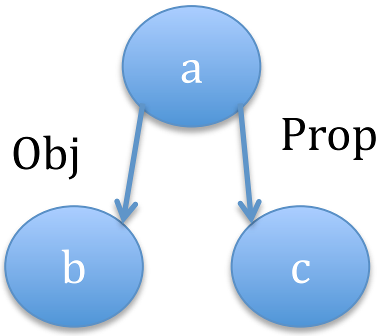
```
End of /home/cendue/divakars/Desktop/CVA Spring 2011/oamDEMOfile.txt
demonstration.


 CPU time : 0.11

* (lisp)
"End of SNePS"
cl-user(3): :ex
; Exiting
timberlake {~/Desktop/CVA Spring 2011} > exit
exit

Script done on Wed 11 May 2011 11:45:29 PM EDT
```

APPENDIX B : SNePS Case Frames

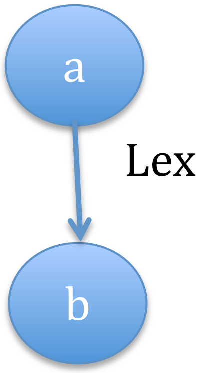
1. Object/Rel/Prop



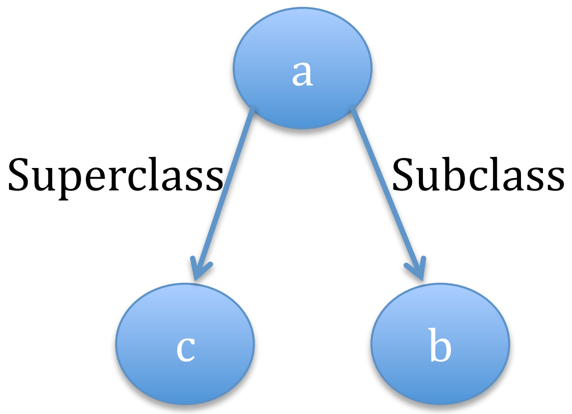
2. Object/Property



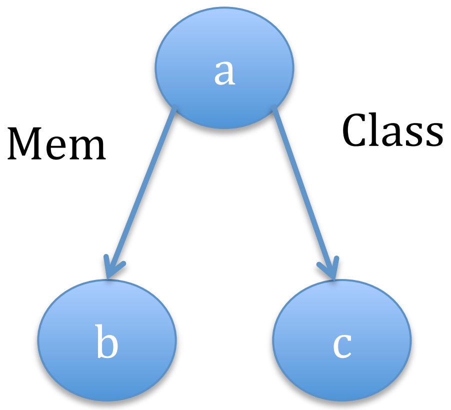
3. Lex



4. Superclass/Subclass



5. Member/Class



APPENDIX C : Case frame syntax

1. Lex express the concept in double quotes

   (build lex "hot")

2. Member/Class

   (build member (build lex "hot soup") class (build lex "soup"))

3. Superclass/Subclass
   (assert superclass (build lex "greek") subclass (build lex "theta"))
4. Object/Rel/Possessor
   (assert object (build lex "Ford") possessor (build lex "Henry") rel (build lex "car"))

5. Object/Property

   (assert object (build lex "bulb") property (build lex "heat"))

REFERENCES

1. Dr. William J. Rapaport http://www.cse.buffalo.edu/~rapaport/cva.html
2. CVA case frames from Scott Napieralski's Dictionary of CVA SNePS case frames. (http://www.cse.buffalo.edu/~rapaport/CVA/CaseFrames/case-frames/)
3. Oxford English Dictionary www.oed.com
4. As a template and motivation, the SNePS project report by Alber Goldfain was used. http://www.cse.buffalo.edu/~rapaport/CVA/Harbinger/ag33.CVA_harbinger.pdf
5. CVA case frames. http://www.cse.buffalo.edu/~rapaport/CVA/CaseFrames/case-frames/
6. The SNePS User Manual
7. Wikipedia for general information on AI.
8. Software used LISP, MAC OSX Terminal, gedit, MS WORD, MS POWERPOINT and Picasa.

A Special thanks to our distinguished guide and instructor in this study –
Dr. William J Rapaport for guiding us and pointing our mistakes, weaknesses and strengths, and being a wonderful teacher.