Defining and Revising Prior Beliefs about ‘quarrels’ through Computational Contextual Vocabulary Acquisition with SNePS and SNeBR

Steven Hsieh

Summer CVA Research Group

[sjhsieh@buffalo.edu](mailto:sjhsieh@buffalo.edu)

August 23, 2010

**Abstract**

The goal of the Contextual Vocabulary Acquisition project is to use the SNePS knowledge representation and reasoning system to effectively simulate the thought process that a human reader would follow when attempting to define an unknown word. This is done through contextual clues and the reader’s own background knowledge and without external aid such as dictionaries and other people. In this study, human subjects are given the task to define the word ‘quarrel’ in a passage. Then, SNePS and SNeBR, SNePS’s belief revision package, are used to represent these models and to define ‘quarrel’ in spite of erroneous prior beliefs about its definition. This report documents the human protocols that were run, SNePS representations of the passage and prior knowledge that were used, and compares the results of the human subjects to those of SNePS.

**1. Introduction to SNePS and the CVA Project**

When someone comes across an unfamiliar word while reading, using a dictionary is the most effective way to find the definition. Not everyone has the discipline to take the time to search in the dictionary for every single unknown word they come across.

This is where Contextual Vocabulary Acquisition project or “CVA” comes in. CVA is the “active, deliberate acquisition of word meanings from text by reasoning from contextual clues, prior knowledge, language knowledge, and hypotheses developed from prior encounters with the word, but without external sources of help such as dictionaries or people” (Rapaport and Kibby, 2002). Often, the reader must figure out the definition of an unknown word on his own relying on the surrounding contextual clues and his own prior knowledge. To do this, the reader goes through a thought process that cognitive scientists and linguists have been attempting to compile into a general theory of vocabulary acquisition. “Most published strategies are quite vague; one simply suggests to ‘look’ and ‘guess’” (Rapaport and Kibby, 2002). The CVA project is an ongoing effort to express this procedure more definitively and algorithmically through computational modeling.

Developing and refining the CVA system would have an impact on the education system. By understanding the way that intuitive CVA works, better strategies could be taught to students “so that they are able to use them successfully when they encounter hard words in their regular reading” (Rapaport and Kibby, 2002). A better understanding of CVA would open the door for “teacher modeling of CVA via think-aloud procedures, teacher scaffolding the class as it uses CVA, think-alouds for CVA in small peer groups, and individual students’ independent applications of CVA with teacher monitoring and assessment” (Rapaport and Kibby, 2002). It would allow teachers to more quickly and concretely help students develop stronger fundamentals in reading comprehension, which is a skill necessary in not just the school system, but in society and in major fields of occupation today.

In this study, the goal is to use SNePS’s noun definition algorithm to deduce an appropriate definition for the word ‘quarrel’ both in the presence and absence of prior knowledge that directly conflicts with the target definition. In the first part of the study, we use the noun definition algorithm to compute the definition of quarrel with the SNePS network being a virtual “tabula rosa” with respect to the word ‘quarrel,’ having no prior beliefs or exposure to the word before reading the passage. In the second part of the study, a conflicting piece of prior knowledge will be added to the network, with the goal of having SNeBR, SNePS’s belief revision package be called to resolve the issue and restore consistency of beliefs.

**2. The Passage and Definition**

The goal of this study was to identify the unknown word ‘quarrel’ in a passage that was taken from Christopher Paolini’s 2008 fantasy novel Brisingr. According to the Merriam-Webster dictionary, ‘quarrel’, when used as a noun like in the passage, has two definitions:

1quarrel: a square-headed bolt or arrow especially for a crossbow.

2quarrel: a usually verbal conflict between antagonists.

It is interesting to note that definitions in a dictionary are listed in chronological order starting from the earliest known usage of the word. Today, the word quarrel has evolved and the second definition is more prevalent in common usage. In the passage, however, the word ‘quarrel’ is used for its first definition. From page 511:

“Take cover!” shouted Roran, and wrenched [his horse] Snowfire's head to the right, forcing the animal to swerve behind a house [1]. A dozen quarrels buried themselves in the side of the building a second later [2]. Turning around, Roran saw that all but one of his warriors had managed to duck behind nearby houses before the soldiers fired [3]. The man who had been too slow lay bleeding in the dirt, a pair of quarrels projecting from his chest [4].

In this study we only use sentences [2] and [4] in our SNePS representations since [1] and [3] do not contain the word and do not add much more to the definition than the other two sentences already do.

**3. Human Protocols**

Before approaching the computational part of the study, some human trials were conducted to provide thought process models to simulate in SNePS. Subjects who were unfamiliar with the word were instructed to read the passage and then form a definition without any outside aid. As they read each sentence containing the word, they were asked for their most current definition.

“Take cover!” shouted Roran, and wrenched [his horse] Snowfire's head to the right, forcing the animal to swerve behind a house [1]. A dozen quarrels buried themselves in the side of the building a second later [2].

After sentence [1] and [2], the common response was that a quarrel was a type of projectile because Roran was taking cover from something that could bury itself in the side of buildings.

Turning around, Roran saw that all but one of his warriors had managed to duck behind nearby houses before the soldiers fired [3]. The man who had been too slow lay bleeding in the dirt, a pair of quarrels projecting from his chest [4].

Upon reading sentences [1] and [2], responses became more streamlined as the readers became more specific in their answers after being given further contextual clues. Responses were narrowed down to quarrels being arrows because arrows are a common type of projectile that project from where they strike.

Interestingly, some subjects were able to identify that quarrels were arrows after coming across only the first instance of ‘quarrel.’ They were able to infer that quarrels were arrows through reasoning that arrows were the projectiles used when men fought on horseback. Though not necessarily true, this logic allows readers to find the correct definition for this particular word and passage. This more involved reasoning will be disregarded for the purposes of this study.

The responses show that prior knowledge is not always immediately available, but each addition to the context will trigger the recall of additional pieces of prior knowledge, which will be continuously compiled. From these trials, “arrow” seems like a reasonable definition for ‘quarrel’ that will allow the reader to understand the text, and it will be the benchmark for the CVA noun algorithm.

**5. SNePS Representation of Prior Knowledge**

SNePS noun algorithm case frames, generalized representations of various sentence structures, are used in order to represent both the prior knowledge and the passage for the CVA noun algorithm to recognize them. The case frames can be found at <http://www.cse.buffalo.edu/~rapaport/CVA/CaseFrames/case-frames/>. Additional case frames used are found at http://www.cse.buffalo.edu/sneps/Manuals/dictionary.pdf

The reasoning behind the protocols revolves around the Identity of Indiscernibles, a part of Leibniz’s Law. It states: “if what appear to be two or more things have all their properties in common they are identical” (Kaya, 2010). While the readers only take into account the properties “can project from chests” and “can bury in buildings,” they subconsciously apply the law to ‘quarrel.’ To express it in a SNePS representation, we use the forall/ant/cq case frame in combination with the member/class and agent/act/action/object case frames.

PK1: (describe (add forall $x forall $y forall $u &ant (build member \*y

class (build lex "projectile")) &ant (build agent \*y act (build action

(build lex "can bury in") object \*u)) &ant (build agent \*x act (build

action (build lex "buries in") object \*u)) cq (build member \*x class

(build lex "projectile"))))

This translates to forall x, y, and u, if y is a member of the class “projectile,” y performs the action “can bury in” object u, u is a member of class building, and x buries in u, x is a member of class projectile. Basically, if a projectile can bury in buildings, and a quarrel does bury in a building, then a quarrel is a projectile. The same rule is applied to arrows and the actions “can project from” and “projects from” to get:

PK2: (describe (add forall $j forall $k forall $l &ant (build member \*k

class (build lex "arrow")) &ant (build agent \*k act (build action

(build lex "can project from") object \*l)) &ant (build agent \*j act (build

action (build lex "projects from") object \*l)) cq (build member \*j class

(build lex "arrow"))))

In order to get SNePS to use the rules to define ‘quarrel,’ the conditions of the rules must first be met. To accomplish this, the following prior knowledge is represented:

PK3: (describe (assert forall $a ant (build member \*a class (build lex "arrow"))

cq (build member \*a class (build lex "projectile"))))

“Arrows are projectiles.”

PK4: (describe (add member #arrow-1 class (build lex "arrow")))

“Arrow-1 is an arrow.” This is necessary, since in order for the conditions of the rules to be fulfilled, there must be a representation of an arrow and a projectile in existence in the network. Since PK3 is already in place, SNePS will infer that arrow-1 is also a projectile.

PK5: (describe (assert forall $p forall $b &ant (build member \*p class (build lex "projectile"))

&ant (build member \*b class (build lex "building"))

cq (build agent \*p act (build action

(build lex "can bury in") object \*b))))

“If p is a projectile and b is a building, p can bury in b.” This will apply to arrow-1 since SNePS inferred that arrow-1 is a projectile.

PK6: (describe (assert forall $r forall $c &ant (build member \*r class (build lex "arrow"))

&ant (build member \*c class (build lex "chest"))

cq (build agent \*r act (build action

(build lex "can project from") object \*c))))

“If r is an arrow and c is a chest, r can project from c.”

**4. SNePS Passage Representation**

As stated in section 2, we choose to represent only sentences [2] and [4] from the passage since [1] and [3] do not contain the word quarrel. In order to represent the sentences with the case frames, each sentence must be simplified without losing any of its meaning or integrity. The two sentences from the text are simplified as follows:

A dozen quarrels buried themselves in the side of the building a second later [2].

Sentence [2] is simplified to “A quarrel buries in a building.” The agent/act/action/object case frame is used to represent it. But first, in order for the rule PK1 to be fulfilled, there must once again be an instance of a quarrel and a building already in existence in the network. So a member of class quarrel and a member of class building must be created with the member/class case frame. This gives the following in SNEPSUL:

T1: (describe (assert member #quarrel-1 class (build lex "quarrel")))

T2: (describe (add member #building-1 class (build lex "building")))

T3: (describe (add agent \*quarrel-1 act (build action (build lex "buries in") object \*building-1)))

Here, T1 and T2 translate to: “quarrel-1” is a member of a class called “quarrel” and “building-1” is a member of a class called “buildings.” T3 translates to: the agent “quarrel-1” performs the action that is called “buries in” with respect to the object “building-1.”

The man who had been too slow lay bleeding in the dirt, a pair of quarrels projecting from his chest [4].

Sentence [4] is simplified to “A quarrel projects from a chest.” Once again, the member/class and agent/act/action/object case frames will be used, this time with “quarrel-1” and “chest-1.” “Quarrel-1” does not need to be created with the member/class case frame again, as it is already in the network. Once again, chest-1 must be created to fulfill the rule PK2.

T4: (describe (add member #chest-1 class (build lex "chest")))

T5: (describe (add agent \*quarrel-1 act (build action (build lex "projects from")

object \*chest-1)))

To simplify both of these sentences, key components are isolated and information that does not provide more information about quarrels is struck out. For example, knowing that a dozen quarrels as opposed to a single quarrel buried in the side of the building and knowing that who’s chest the quarrels project from do not add any value to the information already given.

**6. Additional Prior Knowledge to Evoke SNeBR**

The second part of this study attempts to simulate a reader having a prior belief that the only definition of ‘quarrel’ is “a usually verbal conflict between antagonists.” This is fairly common since the first definition of quarrel is rarely used today. The goal is for SNePS to detect that there is a contradiction of beliefs and to call on SNeBR, its built in belief revision package, to make all of the beliefs consistent. A contradiction must be made between two beliefs represented in the SNePS network.

To do so, we represent that “a quarrel is an argument.” Then we introduce a rule that “nothing can be both an arrow and an argument.” This serves to create a contradiction when SNePS deduces that a quarrel is an arrow from the passage. For the rule, the forall/ant/cq case frame is once again used, but now the min/max/arg case frame is introduced in order to represent “nothing is both an x and a y.” In SNEPSUL:

PK7: (describe (assert forall $x ant (build member \*x class (build lex "quarrel"))

cq (build member \*x class (build lex "argument"))))

PK8: (describe (assert forall $x

ant (build member \*x class (build lex "argument"))

cq (build min 0 max 0 arg (build member \*x

class (build lex "arrow")))))

PK7 translates to: “for all x where x is a member of class argument, a minimum of 0 and a maximum of 0 of the following statement is true: x is a member of class arrow.” The goal is for SNeBR to be evoked when SNePS detects the contradiction so that PK7 will be struck, since it is not the correct definition of quarrel in this instance.

**7. Results and Conclusion**

In part 1, after creating the network with background knowledge and the text, the CVA noun algorithm defines quarrel and returns:

; Ask Cassie what "QUARREL" means:

^(

--> defineNoun "quarrel")

Definition of quarrel:

Possible Class Inclusions: arrow, projectile,

Possible Actions: projects from chest, can project from chest, can bury in building, buries

in building,

nil

This is the definition we were seeking, and shows that SNePS was able to successfully deduce the definition of ‘quarrel’ with its noun definition algorithm.

Moving on to part 2, after SNeBR revises the inconsistent beliefs, the CVA noun definition algorithm defines ‘quarrel’ as:

Definition of quarrel:

Possible Class Inclusions: arrow, projectile,

Possible Actions: projects from chest, can project from chest,

nil

Once again, this is the definition we are seeking. SNeBR and SNePS successfully revised conflicting beliefs and found the proper definition of ‘quarrel’ in this passage.

Though this study had a successful run, there is still work to be done as it could still be expanded upon. A logical next step to this study would be to try to represent the more involved thought process in reasoning that arrows were used when men fought on horseback. Also, one may notice from the belief revision section of the study, the definition of ‘quarrel’ as an “argument” was entirely struck out, yet that definition is still a correct definition of quarrel. Work could be done to simply add the “arrow” definition of quarrel to the network, and represent something along the lines of “a quarrel can be either an arrow or a quarrel,” rather than having just one absolute definition of quarrel for every passage. As one can see from this study, automated computational CVA is still a very complex problem with many open ends. Some long-term future work that could be done includes an automated belief revision package that would not need a user to decide which inconsistent beliefs to be revised. A more involved project that would also be helpful is automated abduction. In this study, Leibniz’s law had to be programmed into the SNePS network, but normally one would think that that would be something that a computational agent should be able to do itself. Finally, a natural language front end to CVA would be very important, as it would more closely simulate the actual reading of a passage as well as a reader’s thought process. In spite of all the obstacles, this and other related studies show that CVA is a very promising field that eventually could be ported into educational strategies to for reading comprehension that could be applied in the real world.**Transcript of Demo for Part 1 of Study:**

==============================================================

Starting image `/util/acl/composer'

with no arguments

in directory `nil'

on machine `localhost'.

International Allegro CL Enterprise Edition

8.2 [Linux (x86)] (Jul 9, 2010 16:08)

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):

;;; Note: The hosting emacs appears to have neither the emacs-mule nor

;;; mule-ucs utf-8 encodings. Thus, Allegro CL international character

;;; support is limited in this emacs session.

cl-user(2): :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:1 2009/03/24 14:53:32] loaded.

Type `(sneps)' or `(snepslog)' to get started.

cl-user(3): (sneps)

Welcome to SNePS-2.7 [PL:1 2009/03/24 14:53:32]

Copyright (C) 1984--2007 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.

8/28/2010 16:21:50

\* (demo "QUARREL.demo")

File /home/eedue/sjhsieh/QUARREL.demo is now the source of input.

CPU time : 0.00

\* ; =======================================================================

; FILENAME: QUARREL.demo

; DATE: 07/05/10

; PROGRAMMER: Steven Hsieh

;; 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 "QUARREL.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:

^(

--> load "/projects/rapaport/CVA/STN2/defun\_noun.cl")

; Loading /projects/rapaport/CVA/STN2/defun\_noun.cl

t

CPU time : 0.02

\*

; 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)

; 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")

(intext "/projects/rapaport/CVA/STN2/demos/rels")

Loading 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")

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:

; =====================

;define relations

(define object agent member class lex)

object is already defined.

agent is already defined.

member is already defined.

class is already defined.

lex is already defined.

(object agent member class lex)

CPU time : 0.00

\*

;Quarrels are arguments

;(describe (assert forall $x ant (build member \*x class (build lex

;"quarrel")) cq (build member \*x class (build lex "argument"))))

;Nothing is both an argument and a arrow

;(describe (assert forall $x

; cq (build min 0 max 1

; arg (build member \*x class (build lex "argument"))

; arg (build member \*x class (build lex "arrow")))))

;Are there any projectiles?

(describe (deduce member $x class (build lex "projectile")))

CPU time : 0.00

\*

;Are there any arrows?

(describe (deduce member $x class (build lex "arrow")))

CPU time : 0.00

\*

;Arrows are projectiles

(describe (assert forall $a ant (build member \*a class (build lex

"arrow")) cq (build member \*a class (build lex "projectile"))))

(m3! (forall v3) (ant (p3 (class (m2 (lex arrow))) (member v3)))

(cq (p4 (class (m1 (lex projectile))) (member v3))))

(m3!)

CPU time : 0.00

\*

;Arrow a1

(describe (add member #arrow-1 class (build lex "arrow")))

(m5! (class (m1 (lex projectile))) (member b1))

(m4! (class (m2 (lex arrow))) (member b1))

(m5! m4!)

CPU time : 0.00

\*

;If p is a projectile and b is a building, p can bury in b

(describe (assert forall $p forall $b &ant (build member \*p class (build

lex "projectile")) &ant (build member \*b class (build lex "building"))

cq (build agent \*p act (build action (build lex "can bury in") object

\*b))))

(m8! (forall v5 v4)

(&ant (p6 (class (m6 (lex building))) (member v5))

(p5 (class (m1 (lex projectile))) (member v4)))

(cq (p8 (act (p7 (action (m7 (lex can bury in))) (object v5))) (agent v4))))

(m8!)

CPU time : 0.00

\*

;If r is an arrow and c is a chest, r can project from c

(describe (assert forall $r forall $c &ant (build member \*r class (build

lex "arrow")) &ant (build member \*c class (build lex "chest")) cq (build

agent \*r act (build action (build lex "can project from") object \*c))))

(m11! (forall v7 v6)

(&ant (p10 (class (m9 (lex chest))) (member v7))

(p9 (class (m2 (lex arrow))) (member v6)))

(cq

(p12 (act (p11 (action (m10 (lex can project from))) (object v7)))

(agent v6))))

(m11!)

CPU time : 0.00

\*

; CASSIE READS THE PASSAGE:

; =========================

;building-1 is a building

(describe (add member #building-1 class (build lex "building")))

(m14! (act (m13 (action (m7 (lex can bury in))) (object b2))) (agent b1))

(m12! (class (m6 (lex building))) (member b2))

(m5! (class (m1 (lex projectile))) (member b1))

(m14! m12! m5!)

CPU time : 0.00

\*

;chest-1 is a chest

(describe (add member #chest-1 class (build lex "chest")))

(m17! (act (m16 (action (m10 (lex can project from))) (object b3))) (agent b1))

(m15! (class (m9 (lex chest))) (member b3))

(m4! (class (m2 (lex arrow))) (member b1))

(m17! m15! m4!)

CPU time : 0.01

\*

;(describe (add forall $i forall $j ant (build member \*j

;class (build lex "arrow")) cq (build member \*i class (build lex "arrow"))))

;quarrel-1 is a quarrel

(describe (assert member #quarrel-1 class (build lex "quarrel")))

(m19! (class (m18 (lex quarrel))) (member b4))

(m19!)

CPU time : 0.00

\*

;quarrel-1 projects from chest-1

(describe (add agent \*quarrel-1 act (build action (build lex "projects from") object \*chest-1)))

(m22! (act (m21 (action (m20 (lex projects from))) (object b3))) (agent b4))

(m22!)

CPU time : 0.00

\*

(describe (add forall $j forall $k forall $l &ant (build member \*k

class (build lex "arrow")) &ant (build agent \*k act (build action

(build lex "can project from") object \*l)) &ant (build agent \*j act (build

action (build lex "projects from") object \*l)) cq (build member \*j class (build lex "arrow"))))

(m27! (act (m13 (action (m7 (lex can bury in))) (object b2))) (agent b4))

(m26! (act (m16 (action (m10 (lex can project from))) (object b3))) (agent b4))

(m25! (class (m1 (lex projectile))) (member b4))

(m24! (class (m2 (lex arrow))) (member b4))

(m23! (forall v10 v9 v8)

(&ant

(p17 (act (p16 (action (m20 (lex projects from))) (object v10))) (agent v8))

(p15 (act (p14 (action (m10)) (object v10))) (agent v9))

(p13 (class (m2)) (member v9)))

(cq (p18 (class (m2)) (member v8))))

(m22! (act (m21 (action (m20)) (object b3))) (agent b4))

(m17! (act (m16)) (agent b1))

(m4! (class (m2)) (member b1))

(m27! m26! m25! m24! m23! m22! m17! m4!)

CPU time : 0.00

\*

;quarrel-1 buries in building-1

(describe (add agent \*quarrel-1 act (build action (build lex "buries

in") object \*building-1)))

(m30!

(act (m29

(action

(m28

(lex buries

in)))

(object b2)))

(agent b4))

(m30!)

CPU time : 0.00

\*

(describe (add forall $x forall $y forall $u &ant (build member \*y

class (build lex "projectile")) &ant (build agent \*y act (build action

(build lex "can bury in") object \*u)) &ant (build agent \*x act (build

action (build lex "buries in") object \*u)) cq (build member \*x class (build lex "projectile"))))

(m32! (forall v13 v12 v11)

(&ant

(p23 (act (p22 (action (m31 (lex buries in))) (object v13))) (agent v11))

(p21 (act (p20 (action (m7 (lex can bury in))) (object v13))) (agent v12))

(p19 (class (m1 (lex projectile))) (member v12)))

(cq (p24 (class (m1)) (member v11))))

(m27! (act (m13 (action (m7)) (object b2))) (agent b4))

(m25! (class (m1)) (member b4))

(m14! (act (m13)) (agent b1))

(m5! (class (m1)) (member b1))

(m32! m27! m25! m14! m5!)

CPU time : 0.00

\*

; Ask Cassie what "QUARREL" means:

^(

--> defineNoun "quarrel")

Definition of quarrel:

Possible Class Inclusions: arrow, projectile,

Possible Actions: projects from chest, can project from chest, can bury in building, buries

in building,

nil

CPU time : 0.01

\*

End of /home/eedue/sjhsieh/QUARREL.demo demonstration.

CPU time : 0.08 **Transcript of Demo for Part 2 of Study:**

==============================================================

Starting image `/util/acl/composer'

with no arguments

in directory `nil'

on machine `localhost'.

International Allegro CL Enterprise Edition

8.2 [Linux (x86)] (Jul 9, 2010 16:08)

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):

;;; Note: The hosting emacs appears to have neither the emacs-mule nor

;;; mule-ucs utf-8 encodings. Thus, Allegro CL international character

;;; support is limited in this emacs session.

cl-user(2): :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:1 2009/03/24 14:53:32] loaded.

Type `(sneps)' or `(snepslog)' to get started.

cl-user(3): (sneps)

Welcome to SNePS-2.7 [PL:1 2009/03/24 14:53:32]

Copyright (C) 1984--2007 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.

8/28/2010 16:24:46

\* (demo "QUARREL2.demo")

File /home/eedue/sjhsieh/QUARREL2.demo is now the source of input.

CPU time : 0.01

\* ; =======================================================================

; FILENAME: QUARREL2.demo

; DATE: 07/05/10

; PROGRAMMER: Steven Hsieh

;; 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 "QUARREL2.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:

^(

--> load "/projects/rapaport/CVA/STN2/defun\_noun.cl")

; Loading /projects/rapaport/CVA/STN2/defun\_noun.cl

t

CPU time : 0.03

\*

; 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)

; 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")

(intext "/projects/rapaport/CVA/STN2/demos/rels")

Loading file /projects/rapaport/CVA/STN2/demos/rels.

CPU time : 0.00

\*

; 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:

; =====================

;define relations

(define object agent member class lex)

object is already defined.

agent is already defined.

member is already defined.

class is already defined.

lex is already defined.

(object agent member class lex)

CPU time : 0.00

\*

;Quarrels are arguments

(describe (assert forall $x ant (build member \*x class (build lex

"quarrel")) cq (build member \*x class (build lex "argument"))))

(m3! (forall v1) (ant (p1 (class (m1 (lex quarrel))) (member v1)))

(cq (p2 (class (m2 (lex argument))) (member v1))))

(m3!)

CPU time : 0.00

\*

;Nothing is both an argument and a arrow

;(describe (assert forall $x

; cq (build min 0 max 1

; arg (build member \*x class (build lex "argument"))

; arg (build member \*x class (build lex "arrow")))))

;Are there any projectiles?

(describe (deduce member $x class (build lex "projectile")))

CPU time : 0.01

\*

;Are there any arrows?

(describe (deduce member $x class (build lex "arrow")))

CPU time : 0.00

\*

;Arrows are projectiles

(describe (assert forall $a ant (build member \*a class (build lex

"arrow")) cq (build member \*a class (build lex "projectile"))))

(m6! (forall v4) (ant (p5 (class (m5 (lex arrow))) (member v4)))

(cq (p6 (class (m4 (lex projectile))) (member v4))))

(m6!)

CPU time : 0.00

\*

;Arrow a1

(describe (add member #arrow-1 class (build lex "arrow")))

(m8! (class (m4 (lex projectile))) (member b1))

(m7! (class (m5 (lex arrow))) (member b1))

(m8! m7!)

CPU time : 0.00

\*

;If p is a projectile and b is a building, p can bury in b

;(describe (assert forall $p forall $b &ant (build member \*p class (build

;lex "projectile")) &ant (build member \*b class (build lex "building"))

;cq (build agent \*p act (build action (build lex "can bury in") object

;\*b))))

;If r is an arrow and c is a chest, r can project from c

(describe (assert forall $r forall $c &ant (build member \*r class (build

lex "arrow")) &ant (build member \*c class (build lex "chest")) cq (build

agent \*r act (build action (build lex "can project from") object \*c))))

(m11! (forall v6 v5)

(&ant (p8 (class (m9 (lex chest))) (member v6))

(p7 (class (m5 (lex arrow))) (member v5)))

(cq

(p10 (act (p9 (action (m10 (lex can project from))) (object v6)))

(agent v5))))

(m11!)

CPU time : 0.00

\*

; CASSIE READS THE PASSAGE:

; =========================

;building-1 is a building

;(describe (add member #building-1 class (build lex "building")))

;chest-1 is a chest

(describe (add member #chest-1 class (build lex "chest")))

(m14! (act (m13 (action (m10 (lex can project from))) (object b2))) (agent b1))

(m12! (class (m9 (lex chest))) (member b2))

(m7! (class (m5 (lex arrow))) (member b1))

(m14! m12! m7!)

CPU time : 0.00

\*

;(describe (add forall $i forall $j ant (build member \*j

;class (build lex "arrow")) cq (build member \*i class (build lex "arrow"))))

;quarrel-1 is a quarrel

(describe (assert member #quarrel-1 class (build lex "quarrel")))

(m15! (class (m1 (lex quarrel))) (member b3))

(m15!)

CPU time : 0.00

\*

;quarrel-1 projects from chest-1

(describe (add agent \*quarrel-1 act (build action (build lex "projects from") object \*chest-1)))

(m18! (act (m17 (action (m16 (lex projects from))) (object b2))) (agent b3))

(m18!)

CPU time : 0.00

\*

(describe (add forall $j forall $k forall $l &ant (build member \*k

class (build lex "arrow")) &ant (build agent \*k act (build action

(build lex "can project from") object \*l)) &ant (build agent \*j act (build

action (build lex "projects from") object \*l)) cq (build member \*j class (build lex "arrow"))))

(m22! (class (m4 (lex projectile))) (member b3))

(m21! (act (m13 (action (m10 (lex can project from))) (object b2))) (agent b3))

(m20! (class (m5 (lex arrow))) (member b3))

(m19! (forall v9 v8 v7)

(&ant

(p15 (act (p14 (action (m16 (lex projects from))) (object v9))) (agent v7))

(p13 (act (p12 (action (m10)) (object v9))) (agent v8))

(p11 (class (m5)) (member v8)))

(cq (p16 (class (m5)) (member v7))))

(m18! (act (m17 (action (m16)) (object b2))) (agent b3))

(m14! (act (m13)) (agent b1))

(m7! (class (m5)) (member b1))

(m22! m21! m20! m19! m18! m14! m7!)

CPU time : 0.00

\*

;quarrel-1 buries in building-1

;(describe (add agent \*quarrel-1 act (build action (build lex "buries

;in") object \*building-1)))

;(describe (add forall $x forall $y forall $u &ant (build member \*y

;class (build lex "projectile")) &ant (build agent \*y act (build action

;(build lex "can bury in") object \*u)) &ant (build agent \*x act (build

;action (build lex "buries in") object \*u)) cq (build member \*x class (build lex "projectile"))))

(describe (assert forall $x

ant (build member \*x class (build lex "argument"))

cq (build min 0 max 0 arg (build member \*x

class (build lex "arrow")))))

(m23! (forall v10) (ant (p17 (class (m2 (lex argument))) (member v10)))

(cq (p19 (min 0) (max 0) (arg (p18 (class (m5 (lex arrow))) (member v10))))))

(m23!)

CPU time : 0.00

\*

CPU time : 0.06

\* (describe (deduce member $x class (build lex "arrow")))

A contradiction was detected within context default-defaultct.

The contradiction involves the newly derived proposition:

(m25! (min 0) (max 0) (arg (m20! (class (m5 (lex (arrow)))) (member (b3)))))

and the previously existing proposition:

(m20! (class (m5 (lex (arrow)))) (member (b3)))

You have the following options:

1. [C]ontinue anyway, knowing that a contradiction is derivable;

2. [R]e-start the exact same run in a different context which is

not inconsistent;

3. [D]rop the run altogether.

(please type c, r or d)

=><= r

In order to make the context consistent you must delete at least

one hypothesis from each of the following sets of hypotheses:

(m23! m19! m18! m15! m12! m11! m7! m3!)

In order to make the context consistent you must delete

at least one hypothesis from the set listed below.

An inconsistent set of hypotheses:

1 : (m23! (forall v10) (ant (p17 (class (m2 (lex (argument)))) (member v10)))

(cq

(p19 (min 0) (max 0) (arg (p18 (class (m5 (lex (arrow)))) (member v10)))))) (2 supported propositions: (m25! m23!) )

2 : (m19! (forall v9 v8 v7)

(&ant

(p15 (act (p14 (action (m16 (lex (projects from)))) (object v9)))

(agent v7))

(p13 (act (p12 (action (m10 (lex (can project from)))) (object v9)))

(agent v8))

(p11 (class (m5 (lex (arrow)))) (member v8)))

(cq (p16 (class (m5 (lex (arrow)))) (member v7)))) (3 supported propositions: (m21! m20! m19!) )

3 : (m18! (act (m17 (action (m16 (lex (projects from)))) (object (b2))))

(agent (b3))) (3 supported propositions: (m21! m20! m18!) )

4 : (m15! (class (m1 (lex (quarrel)))) (member (b3))) (3 supported propositions: (m25! m24! m15!) )

5 : (m12! (class (m9 (lex (chest)))) (member (b2))) (4 supported propositions: (m21! m20! m14! m12!) )

6 : (m11! (forall v6 v5)

(&ant (p8 (class (m9 (lex (chest)))) (member v6))

(p7 (class (m5 (lex (arrow)))) (member v5)))

(cq

(p10 (act (p9 (action (m10 (lex (can project from)))) (object v6)))

(agent v5)))) (4 supported propositions: (m21! m20! m14! m11!) )

7 : (m7! (class (m5 (lex (arrow)))) (member (b1))) (4 supported propositions: (m21! m20! m14! m7!) )

8 : (m3! (forall v1) (ant (p1 (class (m1 (lex (quarrel)))) (member v1)))

(cq (p2 (class (m2 (lex (argument)))) (member v1)))) (3 supported propositions: (m25! m24! m3!) )

Enter the list number of a hypothesis to examine or

[d] to discard some hypothesis from this list,

[a] to see ALL the hypotheses in the full context,

[r] to see what you have already removed,

[q] to quit revising this set, or

[i] for instructions

(please type a number OR d, a, r, q or i)

=><= d

Enter the list number of a hypothesis to discard,

[c] to cancel this discard, or [q] to quit revising this set.

=><= 8

The consistent set of hypotheses:

1 : (m23! (forall v10) (ant (p17 (class (m2 (lex (argument)))) (member v10)))

(cq

(p19 (min 0) (max 0) (arg (p18 (class (m5 (lex (arrow)))) (member v10)))))) (1 supported proposition: (m23!) )

2 : (m19! (forall v9 v8 v7)

(&ant

(p15 (act (p14 (action (m16 (lex (projects from)))) (object v9)))

(agent v7))

(p13 (act (p12 (action (m10 (lex (can project from)))) (object v9)))

(agent v8))

(p11 (class (m5 (lex (arrow)))) (member v8)))

(cq (p16 (class (m5 (lex (arrow)))) (member v7)))) (3 supported propositions: (m21! m20! m19!) )

3 : (m18! (act (m17 (action (m16 (lex (projects from)))) (object (b2))))

(agent (b3))) (3 supported propositions: (m21! m20! m18!) )

4 : (m15! (class (m1 (lex (quarrel)))) (member (b3))) (1 supported proposition: (m15!) )

5 : (m12! (class (m9 (lex (chest)))) (member (b2))) (4 supported propositions: (m21! m20! m14! m12!) )

6 : (m11! (forall v6 v5)

(&ant (p8 (class (m9 (lex (chest)))) (member v6))

(p7 (class (m5 (lex (arrow)))) (member v5)))

(cq

(p10 (act (p9 (action (m10 (lex (can project from)))) (object v6)))

(agent v5)))) (4 supported propositions: (m21! m20! m14! m11!) )

7 : (m7! (class (m5 (lex (arrow)))) (member (b1))) (4 supported propositions: (m21! m20! m14! m7!) )

Enter the list number of a hypothesis to examine or

[d] to discard some hypothesis from this list,

[a] to see ALL the hypotheses in the full context,

[r] to see what you have already removed,

[q] to quit revising this set, or

[i] for instructions

(please type a number OR d, a, r, q or i)

=><= q

The following (not known to be inconsistent) set of

hypotheses was also part of the context where the

contradiction was derived:

(m6!)

Do you want to inspect or discard some of them?

=><= no

Do you want to add a new hypothesis? no

(m20! (class (m5 (lex arrow))) (member b3))

(m7! (class (m5)) (member b1))

(m20! (class (m5 (lex arrow))) (member b3))

(m7! (class (m5)) (member b1))

(m20! m7!)

CPU time : 0.01

; Ask Cassie what "QUARREL" means:

\* ^(defineNoun "quarrel")

--> Definition of quarrel:

Possible Class Inclusions: arrow, projectile,

Possible Actions: projects from chest, can project from chest,

nil

End of /home/eedue/sjhsieh/QUARREL2.demo demonstration.

\*

CPU time : 0.01

**References**

1. Napieralski, Scott (Fall 2002), “Noun Algorithm Case Frames”,

[http://www.cse.buffalo.edu/~rapaport/CVA/CaseFrames/case-frames].

1. Shapiro, Stuart C., & Rapaport, William J. (1992-94), “A Dictionary of SNePS Case

Frames”, retrieved from [http://www.cse.buffalo.edu/sneps/Manuals/

dictionary.pdf].

1. Rapaport, William J., & Kibby, Michael W. (2002b), "ROLE: Contextual Vocabulary

Acquisition: From Algorithm to Curriculum", retrieved from [http://www.cse.buffalo.edu/~rapaport/CVA/role2002.pdf]

1. Kaya, Serder (August 2010), Philosophy Professor, "identity of

indiscernibles" [http://www.philosophyprofessor.com/philosophies/

identity-of-indiscernibles.php].