A step towards adverbs: defining 'emvasally' ['In the red'] using Contextual Vocabulary Acquisition and the SNePS KRR system

TJ Burns
CSE663: Advanced Knowledge Representation and Reasoning
tjburns@cse.buffalo.edu

December 15, 2006

Abstract

Contextual Vocabulary Acquisition is a means by which cognitive agents can derive the meaning of a word based only on its context and background knowledge. This project makes use of the SNePS semantic net-based Knowledge Representation system to explore this topic for a specific passage. The passage in question is one in which the unknown word is an adverb. This presents some interesting challenges due to the fact that adverbs are traditionally harder to define than nouns or verbs. Furthermore, the definition algorithms developed for the CVA project thus far only apply to nouns and verbs. We develop our own case-frames to represent the sentence, and explore the background knowledge required to allow CASSIE (The SNePS cognitive agent) the ability to deduce a reasonable meaning of the unknown adverbial phrase 'emvasally' ['in the red']. From this we then conclude a reasonable general rule for determining the meaning of adverbs based on context in the hope that it may help in leading to the creation of a general purpose adverb definition algorithm.

1 The CVA Project and SNePS

1.1 An introduction to the CVA project and what we hope to accomplish

The ability to decipher a word's meaning based on context is an important skill that we as human beings seem to develop quite naturally and which often proves to be invaluable. When an individual is reading and encounters a word with an unknown meaning, context is often the first tool to be used in the process of building a definition. This is particularly the case if the meaning is necessary for the understanding of the rest of the context. This "self-discovery" is preferable to being told a meaning or having to look one up in a dictionary or other such resource. Contextual Vocabulary Acquisition (CVA) is a method by which this discovery can be performed. In this project we will attempt to define such an unknown word using purely contextual information and key background knowledge. We will present CASSIE (The SNePS cognitive agent) with a representation of the passage along with background knowledge that an average reader would need to determine a meaning, and finally allow CASSIE to perform 'read' the passage and perform inferences to build a reasonable meaning for the unknown word.

In representing the passage and background knowledge it will be important to remain unbiased and represent only what is found in the passage and only the background knowledge absolutely necessary for building a meaning. In effect, this means that in representing the passage itself we should not perform our own hidden inferences and simply say "it just seems obvious!" This kind of reasoning may seem obvious to us, but in reality it is not explicitly there in passage. Anywhere this kind of reasoning could be done will be filled in with background knowledge that will help lead the system to a

definition. Any further background knowledge may either be unnecessary or might, in fact, provide a meaning directly to CASSIE. This clearly undermines the idea of CVA in which we want CASSIE to search out a meaning from the context (the passage itself) using only minimal background knowledge. After all this is represented CASSIE will then be turned loose on the passage and asked to produce a definition.

In most cases a definition algorithm would be used to help CASSIE build a meaning for an unknown word, however, there only currently exists definition algorithms for nouns and verbs, and in this project we want to have CASSIE determine the meaning of an adverb. It has proven traditionally easier, not just for computers but for humans as well, to determine a meaning for nouns and verbs based on context than both adjectives and adverbs. This is somewhat intuitive when we consider that adjectives and adverbs are both modifying words and thus only seem to aid in describing other features of the context. The goal of this project will be to get CASSIE to produce a reasonable definition or 'meaning postulate' for the unknown adverb. This result can then be used to postulate possible investigations into a more general adverb definition algorithm.

It is important to note that, in general, CVA does not always produce a 'correct' definition, but instead one which can be used to further elaborate upon successive exposures to a word in new contexts. As Rapaport and Ehrlich describe, "*The system does not develop 'correct' definitions, but dictionary-like definitions enabling it to continue understanding the text."* (Rapaport and Ehrlich, 2000) This is of particular importance in this project since we don't have a definition algorithm so the definition we will be left

with in the end will be very minimal which we note that CASSIE could conceivably further define this definition to include more information. It will also be interesting to note that in this project we will be dealing with a commonly used word in different way. This is an example of how, even when we think we are sure of the meaning of a word, when it's used in a new context, we might need to reconsider.

1.2 A brief introduction to the SNePS KRR system

As was mentioned before we will be using the SNePS Knowledge Representation system (see http://www.cse.buffalo.edu/sneps for more information) to represent the passage and background knowledge SNePS is a propositional semantic network system in which we represent propositions as case frames containing nodes and arcs relating them. This system is in effect a model of the mind of a cognitive agent CASSIE (as referred to above). We use SNePSLOG to define predicate-like case frames which will be used in building the network. SNePSLOG is an interface to SNePS made to look like standard first-order predicate logic. This allows us to produce code that appears much like FOL, but that under the surface is building networks in much the same way as they would be built in SNePSUL (The SNePS user language). One key difference to note, however, is that although SNePSLOG appears to be like FOL, wffs in SNePS are propositions which allows us to write expressions that appear to be out of the scope of FOL. The expressive power of propositions as wffs fits well with the need of a language-oriented task such as this. We will begin with a network of background knowledge tat CASSIE 'knows' before she reads the passage, and will end up with a network containing all the inferences CASSIE is able to derive from the passage given what she already knows. In this project

we will be interested primarily with the network that is created from CASSIE 'reading' the passage and we will ask questions that coax a reasonable proposition describing what CASSIE knows about the unknown word. This will serve as the definition that would normally have come from a definition algorithm. Therefore, unlike a noun or a verb, we will be interested not so much in the structure and context of the entire network, but in what inferences CASSIE will be able to derive that provide us with a reasonable definition. Such inferences might provide us with a clue of how to search the network for an adverb's definition and thus define a new adverb algorithm.

2 The Passage in Question

The following is the passage we will be using for our project:

"After operating slightly in the red for many months, the struggling young firm […] landed a large government contract." (Dulin, 1970)

The unknown word in the above passage is the word 'red.' At first we thought it would be our goal to define how the word 'red' is used in the above context since it is clearly not referring to the color. However, simply examining just the meaning of the word 'red' on its own would be to look for a noun definition and that is clearly not how the word is functioning in this sentence. In order to be more true to the way the word is being used we look at the idiomatic adverbial phrase 'in the red' which seems much more appropriate. As discussed above, however, this provides the challenge of not having an adverb definition algorithm to work with.

The context from which this passage was obtained offers a definition for the phrase as ‘at a loss.’ Anyone familiar with accounting or bookkeeping should know that this phrase originated due to the fact that traditionally financial loss is indicated in red pen or ink. Therefore, the definition provided seems reasonable. In fact, when trying to provide a definition we will try to get CASSIE to produce some sort of rule relating the unknown phrase to being at a loss.

3 Verbal Protocols

Before representing any of the passage itself or the background knowledge we ran some experiments to get a feel for how real people would perform CVA on this passage. We ran three experiments (one each for three individuals) asking the individuals to produce a definition and speak what they were thinking. These are referred to as “think aloud verbal protocols.”

Before showing the passage to any of the test subjects, I decided to modify it to include a made up adverb in place of the phrase ‘in the red.’ I did this because I assumed that most people would know what the phrase ‘in the red’ means in this context. The modified passage is as follows:

“After operating slightly *emvasally* for many months, the struggling young firm landed a large government contract.”

The first two individuals provided similar responses:

E1: "…Barely getting by, living on the edge. Perhaps the government contract gets them out."
E2: "I guess I would say operating under its own authority because the firm was already operating before it got this contract so it must have been independent."

Both these responses seem to indicate an underlying theme of time which we believe is important to note. Also, they indicate that somehow after landing the government contract the state of the firm changes in some way thus indicating some sort of before/after contrast. One other thing that came to our attention from subject 1 was that he wasn't sure what 'landing' meant in this context. This was something we had assumed to be known, but clearly could also be the subject of its own CVA project. Therefore, the information that 'landing' and 'receiving' are equivalent here was noted for necessary background knowledge. The third individual gave the following very helpful response:

E3: "I would guess that it means unsuccessfully because it's struggling [before]…Because it's young and struggling and maybe a large government contract would make things easier [after]…means more money, more business, maybe more legitimacy."

This confirms the before/after contract as being necessary and also seems to indicate that a knowledge about the properties of government contracts (money, success, etc…) being

part of this contrast. Overall it is clear from the protocols that a good definition for the adverb can be found from the state of the object performing the action which the adverb is modifying. (i.e. is it successful? Does it have money? Etc…)

4 Our SNePS Representation

4.1 Case Frames

The following standard case-frames will be used in our representation:

- agent/act/action/object
- lex
- member/class
- object/property
- object/rel/possessor

These can be found on the website

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

And the following non-standard case-frames were created:

- act/way/agent
- word
- amount/unit

The act/way/agent case-frame was created to incorporate the notion of a modifier on a verb (or action). This is what the way arc gets us since it can point to some modifying concept. The word case frame was created to express the notion of a word with unknown meaning. This will be discussed further later on. Finally the amount/unit case frame was created to be able to represent units. (as in ‘many months’)

Finally, the following case frames were created to introduce a theory of time:

- before/after
- interval/dur
- arg/stime/etime

These case frames will be crucial because much of the reasoning that CASSIE will be doing relates to time. These case frames were based on the theory of time developed by Michael Almeida. (Almeida, 1995).

Since we are using SNePSLOG, the above case-frames will appear using the following syntax and semantics throughout the representation (given in SNePSLOG definitions):

```
;[[Does(x,y,z)]] = The proposition that agent [[z]] performs the act
;  [[y]] in the way given by [[w]].
define-frame Does(nil act way agent)

;[[degree-to-which(x,y)]] = The concept of the degree [[y]] to which
;  the manner given by [[x]] is specified.
define-frame degree-to-which(nil manner degree)

;[[MeaningUnknown(x)]] = The proposition that the word [[x]] has an
;  unknown meaning.
define-frame MeaningUnknown(nil word)

;[[HasProperty(x,y)]] = The proposition that object [[y]] has the
;  property [[x]] at time.
define-frame HasProperty(nil property object)

;[[amt-in-units(x,y)]] = The concept of the amount [[x]] given in the
;  units of [[y]].
define-frame amt-in-units(nil amount unit)

;[[concept-called(x)]] = The concept given by [[x]].
define-frame concept-called(nil lex)

;[[Isa(x,y)]] = The proposition that [[x]] is a member of the class
[[y]].
define-frame Isa(nil member class)

;[[action-wrt(x,y)]] = The act of performing action [[x]] with respect
;  to object [[y]]].
define-frame action-wrt(nil action object)

;[[Before(x,y)]] = The proposition that time [[x]] is before time [[y]]
define-frame Before(nil before after)

;[[TimeInterval(x,y)]] = The proposition that time in the interval
;  given by [[x]] is [[y]]
define-frame TimeInterval(nil interval dur)
```

```
;[[EventTime(x,y,z)]] = The proposition that some event or state [[x]]
;  lasts for the time interval from [[y]] to [[z]].
define-frame EventTime(nil arg stime etime)

;[[Has(x,y,z)]] = The proposition that [[x]] is a [[y]] of [[z]];
define-frame Has(nil object rel possessor)
```

4.2 Background Knowledge

Based on the verbal protocols we want CASSIE to have enough background knowledge to be able to produce a rule saying that emvasally implies a lack of money. In order to do this the following background knowledge will be required:

1. Landing is the same as receiving.
2. If some agent receives something at a certain time then they have that something for all future intervals of time.
3. If some agent has a government contract at some time t then they will have money starting from that time.
4. If some agent has money during some interval of time, then they are not struggling during that interval.
5. If some agent is currently not struggling and has money, but was struggling at some earlier time period, then they did not have money during the previous time period.

The above six pieces of knowledge are all described in a very high level manner. The precise SNePSLOG representation is given in the transcript in Appendix A. We will now provide some rational for the above background knowledge.

Part 1 is used to represent the equivalence of landing and receiving as discussed earlier. This is done in order to avoid any potential ambiguities about the meaning of the word 'landed' in this context.

Part 2 defines what it means to receive and posses something. It is clear that receiving something means you will possess that thing from the time you received it onward. It could be argued that this might not always be the case (because you might then give it to someone else), but for our purposes this rule will suffice since we don't need to worry about what it means to give something.

Part 3 defines what it means financially for someone to have a government contract. It basically says that a government contract means that the individual will be receiving money from that time onward. (Again not strictly true, but for our purposes will suffice) This will help to define the contrast from having money after landing the contract to struggling before the contrast. This is done with the intent of concluding that the firm had no money before the contract was received.

Part 4 further develops this contrast by linking having money for a certain time period to not struggling for that time period. This is reasonable because most readers would conclude that the firm would no longer be struggling after receiving the contract. This rule simply makes this knowledge concrete by showing why the firm would no longer be struggling (because it now has money).

Finally, Part 5 makes this last connection by providing a rule for contrast. The verbal protocols provided good rational for this rule by showing an implicit knowledge of some before/after contrast linking the struggling with lack/possession of money. It's important

to note that all background knowledge about time in the before/after sense is all implicit (unconscious) path knowledge.

Now that we have provided CASSIE with enough background knowledge to conclude that the firm did not have money during this 'before' time, we need a meaning postulate (a rule from which we can conclude a reasonable meaning) that will make the connection from the word 'emvasally' to a lack of money. This rule will be stated as follows:

Meaning Postulate: If some agent has a certain property (either of the item itself or possessive) during a time interval and they perform an action in a certain manner which has an as of yet unknown meaning, then we can conclude that performing the action in the given way implies the property of the agent.

Representing this rule is where we make use of the MeaningUnknown() SNePSLOG case-frame. The use of this 'unknown' type of predicate is similar in a sense to McCarthy's theory of Circumscription. (McCarthy, 1980) We will use this rule to query CASSIE after she reads the passage to see if she can indeed conclude such a definition. The validity of this meaning postulate is based on the fundamental assumption taking in the project that the meaning of an adverb can be deduced from the states of the agent performing the modified verb.

4.3 Representation of the passage

We now turn our attention to representing the passage itself. We begin by noting the clear before/after structure of the sentence and break down the passage into two distinct segments as follows:

1) The Struggling you firm operated slightly emvasally for many months.
2) The struggling young firm landed a large government contract.

The task now becomes to represent parts 1 and 2 and connect them with a notion of time (more specifically a before/after relation).

The following are the specific definitions in SNePSLOG for part 1:

```
Isa(firm1, concept-called(Firm))!
HasProperty(concept-called(young), firm1)!
EventTime(HasProperty(concept-called(struggling), firm1), time1, time2)!
```

And the specifics of part 2 are as follows:

```
Isa(contract1, concept-called(governmentContract))!
HasProperty(concept-called(large), contract1)!
```

Now to combine parts 1 and 2 we need to introduce a theory of time:

```
Before(time1,time2)!
Before(time2,now)!
TimeInterval(Before(time1,time2),amt-in-units(concept-called(many), concept-called(months)))!

EventTime(Does(concept-called(operate), degree-to-which(concept-called(emvasally), concept-called(slightly)),firm1), time1, time2)!
EventTime(Does(action-wrt(concept-called(land),contract1),concept-called(normal),firm1), time2, time2)!
```

Note that all the information in the passage, and only the information in the passage in represented here. Everything is asserted with the ! symbol to indicate to CASSIE to

perform forward inference on everything she ‘reads.’ That is, we want her to conclude everything she can given her background knowledge about what she just read so that we can ask her what she now knows about the meaning of ‘emvasally.’

5 Results

After representing the background knowledge and have CASSIE ‘read’ the passage we then ask a series of questions to see if CASSIE now has any possible definition of what emvasally may mean. These are done in the form of the following questions:

```
; Ask Cassie what she now knows:
; We will do this first by asking whether the firm had money before landing the contract
;  (ie while operating emvasally)
EventTime(~Has(some, concept-called(money),firm1),time1,time2)?

  wff43!:  EventTime(~Has(some,concept-called(money),firm1),time1,time2)

 CPU time : 0.00

:

; Now we will ask the following question to determine if Cassie has deduced
; a reasonable conclusion about the meaning of emvasally:

; Does operating "emvasally" at a certain time imply a lack of money during that time?
(EventTime(Does(concept-called(operate),degree-to-which(concept-
called(emvasally),concept-called(slightly)),firm1),time1,time2) =>
EventTime(~Has(some,concept-called(money),firm1),time1,time2))?

  wff58!:  EventTime(Does(concept-called(operate),degree-to-which(concept-
called(emvasally),concept-called(slightly)),firm1),time1,time2) =>
EventTime(~Has(some,concept-called(money),firm1),time1,time2)


 CPU time : 0.00

:
```

By SNePS responding the way it does, we know that CASSIE dos indeed believe that the firm had no money during the ‘before’ time and that operating ‘emvasally’ implies this. We can thus reasonably conclude that CASSIE was able to determine a possible meaning

for the unknown word based on its context and appropriate background knowledge. This lends some promise to the fundamental assumption that the meaning of an adverb is associated with the properties of the acting agent.

6 Future Work

It is clear that computing the meaning of an adverb is much more difficult than nouns or verbs due to the variety of ways they are used as modifiers. It is promising, however, to see that in certain contexts the meaning appears to be related to the context in somewhat simple ways. In our passage the meaning was determined by investigating the contrasting properties of the agent performing the action the adverb was modifying. This produced a meaning postulate from which CASSIE concluded a reasonable definition for emvasally.

The ultimate goal of a project like this one is to provide further insight into CVA for adverbs in the hope that, given enough examples like this one, an algorithm will emerge by which definitions can be computed in a similar fashion to nouns and verbs. Although future work into this particular passage is limited, we believe it would be a god pursuit to try to generalize the meaning postulate to include all properties of the acting agent rather than the possessive (which we used). Any other future work into this area should be done in examining other possible ways to produce a meaning from context, and this is best accomplished by exploring other more complex adverbial passages in detail.

It is our hope that the reader take away from this project at least the start of a set of rules by which the meaning of adverbs can be computed based on context.

Appendix A: Annotated SNePS demonstration

Script started on Tue Dec 12 18:25:19 2006
pollux {~/CSE_663} > acl

International Allegro CL Enterprise Edition

8.0 [Solaris] (Jun 30, 2006 12:35)

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/shapiro/Sneps/sneps262

; Loading /projects/shapiro/Sneps/sneps262.cl

; Loading /projects/shapiro/Sneps/Sneps262/load-sneps.lisp

; Loading

; /projects/snwiz/Install/Sneps-2.6.1/load-logical-pathnames.lisp

Loading system SNePS...10% 20% 30% 40% 50% 60% 70% 80% 90% 100%

SNePS-2.6 [PL:1a 2006/12/04 04:07:47] loaded.

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

```
cl-user(2): (snepslog)



   Welcome to SNePSLOG (A logic interface to SNePS)


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.


: demo emvasally.demo


File /home/csdue/tjburns/CSE_663/emvasally.demo is now the source of input.



 CPU time : 0.06


: ;
================================================================
========

; FILENAME:     emvasally.demo

; DATE:         Nov 13, 2006

; PROGRAMMER:   TJ Burns

;
```

;

; Lines beginning with a semi-colon are comments.

; Lines beginning with "^" are Lisp commands.

; Lines beginning with "%" are SNePSUL commands.

; All other lines are SNePSLOG commands.

;

; To use this file: run SNePSLOG; at the SNePSLOG prompt (:), type:

;

; demo "emvasally.demo" av

;

; Make sure all necessary files are in the current working directory

; or else use full path names.

;
==========

; Set SNePSLOG mode = 3

set-mode-3

Net reset

In SNePSLOG Mode 3.

Use define-frame <pred> <list-of-arc-labels>.

achieve(x1) will be represented by {<action, achieve>, <object1, x1>}

ActPlan(x1, x2) will be represented by {<act, x1>, <plan, x2>}

believe(x1) will be represented by {<action, believe>, <object1, x1>}

disbelieve(x1) will be represented by {<action, disbelieve>, <object1, x1>}

adopt(x1) will be represented by {<action, adopt>, <object1, x1>}

unadopt(x1) will be represented by {<action, unadopt>, <object1, x1>}

do-all(x1) will be represented by {<action, do-all>, <object1, x1>}

do-one(x1) will be represented by {<action, do-one>, <object1, x1>}

Effect(x1, x2) will be represented by {<act, x1>, <effect, x2>}

else(x1) will be represented by {<else, x1>}

GoalPlan(x1, x2) will be represented by {<goal, x1>, <plan, x2>}

if(x1, x2) will be represented by {<condition, x1>, <then, x2>}

ifdo(x1, x2) will be represented by {<if, x1>, <do, x2>}

Precondition(x1, x2) will be represented by {<act, x1>, <precondition, x2>}

snif(x1) will be represented by {<action, snif>, <object1, x1>}

sniterate(x1) will be represented by {<action, sniterate>, <object1, x1>}

snsequence(x1, x2) will be represented by {<action, snsequence>, <object1, x1>, <object2, x2>}

whendo(x1, x2) will be represented by {<when, x1>, <do, x2>}

wheneverdo(x1, x2) will be represented by {<whenever, x1>, <do, x2>}

withall(x1, x2, x3, x4) will be represented by {<action, withall>, <vars, x1>, <suchthat, x2>, <do, x3>, <else, x4>}

withsome(x1, x2, x3, x4) will be represented by {<action, withsome>, <vars, x1>, <suchthat, x2>, <do, x3>, <else, x4>}

CPU time : 0.00

:

; Turn off inference tracing; this is optional.

; If tracing is desired, enter "trace" instead of "untrace":

untrace inference

Untracing inference.

CPU time : 0.00

:

; Clear the SNePS network:

clearkb

Knowledge Base Cleared

CPU time : 0.00

:

; OPTIONAL:

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

;

; ^(cl:load "/projects/rapaport/CVA/STN2/ff")

; Case Frames:

; ==========

;[[Does(x,y,z)]] = The proposition that agent [[z]] performs the act

; [[y]] in the way given by [[w]].

define-frame Does(nil act way agent)

Does(x1, x2, x3) will be represented by {<act, x1>, <way, x2>, <agent, x3>}

CPU time : 0.00

:

;[[degree-to-which(x,y)]] = The concept of the degree [[y]] to which

; the manner given by [[x]] is specified.

define-frame degree-to-which(nil manner degree)

degree-to-which(x1, x2) will be represented by {<manner, x1>, <degree, x2>}

CPU time : 0.00

:

;[[MeaningUnknown(x)]] = The proposition that the word [[x]] has an unknown

; meaning.

define-frame MeaningUnknown(nil word)

MeaningUnknown(x1) will be represented by {<word, x1>}

CPU time : 0.00

:

;[[HasProperty(x,y)]] = The proposition that object [[y]] has the

; property [[x]] at time.

define-frame HasProperty(nil property object)

HasProperty(x1, x2) will be represented by {<property, x1>, <object, x2>}

CPU time : 0.00

:

;[[amt-in-units(x,y)]] = The concept of the amount [[x]] given in the

; units of [[y]].

define-frame amt-in-units(nil amount unit)

amt-in-units(x1, x2) will be represented by {<amount, x1>, <unit, x2>}

CPU time : 0.00

:

;[[concept-called(x)]] = The concept given by [[x]].

define-frame concept-called(nil lex)

concept-called(x1) will be represented by {<lex, x1>}

CPU time : 0.00

:

;[[Isa(x,y)]] = The proposition that [[x]] is a member of the class [[y]].

define-frame Isa(nil member class)

Isa(x1, x2) will be represented by {<member, x1>, <class, x2>}

CPU time : 0.00

:

;[[action-wrt(x,y)]] = The act of performing action [[x]] with respect

; to object [[y]]].

define-frame action-wrt(nil action object)

action-wrt(x1, x2) will be represented by {<action, x1>, <object, x2>}

CPU time : 0.00

:

;[[Before(x,y)]] = The proposition that time [[x]] is before time [[y]]

define-frame Before(nil before after)

Before(x1, x2) will be represented by {<before, x1>, <after, x2>}

CPU time : 0.00

:

;[[TimeInterval(x,y)]] = The proposition that time in the interval given by [[x]]

; is [[y]]

define-frame TimeInterval(nil interval dur)

TimeInterval(x1, x2) will be represented by {<interval, x1>, <dur, x2>}

CPU time : 0.00

:

;[[EventTime(x,y,z)]] = The proposition that some event or state [[x]] lasts

; for the time interval from [[y]] to [[z]].

define-frame EventTime(nil arg stime etime)

EventTime(x1, x2, x3) will be represented by {<arg, x1>, <stime, x2>, <etime, x3>}

CPU time : 0.00

:

;[[Has(x,y,z)]] = The proposition that [[x]] is a [[y]] of [[z]];

define-frame Has(nil object rel possessor)

Has(x1, x2, x3) will be represented by {<object, x1>, <rel, x2>, <possessor, x3>}

CPU time : 0.00

:

; Unconscious (Path) Knowledge:

; ==========

```
; Notions about time:


;;; Make Before Transitive

define-path before (compose before (kstar (compose after- ! before))).

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


:

;;; Make After Transitive

define-path after (compose after (kstar (compose before- ! after))).

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
```

:

; BACKGROUND KNOWLEDGE:

; ==========

; We need to included information that will help deduce that the young

; firm was lacking financially before the contract.

; 1) If x lands y at time t, then x receives y at time t.

all(x,y,w,t)(EventTime(Does(action-wrt(concept-called(land),y),w,x),t,t) =>
EventTime(Does(action-wrt(concept-called(receive),y),w,x),t,t)).

wff3!: all(t,w,y,x)(EventTime(Does(action-wrt(concept-called(land),y),w,x),t,t) =>
EventTime(Does(action-wrt(concept-called(receive),y),w,x),t,t))

CPU time : 0.01

:

; 2) If x receives y at time t1, then x has y for any future interval (t1,t2)

all(x,y,w,z,t1,t2)({EventTime(Does(action-wrt(concept-called(receive),y),w,x),t1,t1),
Before(t1,t2), Isa(y,z)} &=> EventTime(Has(y,z,x),t1,t2)).

wff4!: all(t2,t1,z,w,y,x)({Isa(y,z),Before(t1,t2),EventTime(Does(action-wrt(concept-called(receive),y),w,x),t1,t1)} &=> {EventTime(Has(y,z,x),t1,t2)})

CPU time : 0.04

:

; 3) If x has a government contract for some time interval t, then x has some money starting for the same interval.

all(x,y,z,t1,t2)((EventTime(Has(y,z,x),t1,t2) and Isa(y,concept-called(governmentContract))) => EventTime(Has(some,concept-called(money),x),t1,t2)).

wff7!: all(t2,t1,z,y,x)((Isa(y,concept-called(governmentContract)) and EventTime(Has(y,z,x),t1,t2)) => EventTime(Has(some,concept-called(money),x),t1,t2))

CPU time : 0.01

:

; 4) If x has some money during some interval (t1,t2), then x is not struggling during that interval.

all(x,t1,t2)(EventTime(Has(some, concept-called(money),x),t1,t2) => EventTime(~HasProperty(concept-called(struggling),x),t1,t2)).

wff9!: all(t2,t1,x)(EventTime(Has(some,concept-called(money),x),t1,t2) => EventTime(~HasProperty(concept-called(struggling),x),t1,t2))

CPU time : 0.00

:

; 5) If x has some money and is not struggling at time interval (t2,t3), but x was struggling at time interval (t1,t2), then

; x did not have money during time interval (t1,t2).

all(x,t1,t2,t3)({EventTime(Has(some, concept-called(money),x),t2,t3), EventTime(~HasProperty(concept-called(struggling),x),t2,t3),EventTime(HasProperty(concept-called(struggling),firm1),t1,t2), Before(t1,t2)} &=>

EventTime(~Has(some, concept-called(money),x),t1,t2)).

wff11!: all(t3,t2,t1,x)({Before(t1,t2),EventTime(HasProperty(concept-called(struggling),firm1),t1,t2),EventTime(~HasProperty(concept-called(struggling),x),t2,t3),EventTime(Has(some,concept-called(money),x),t2,t3)} &=> {EventTime(~Has(some,concept-called(money),x),t1,t2)})

CPU time : 0.00

:

; 6) If x does not have y during a time interval (t1,t2), and if x performs an action z in a certain manner w which has an unknown

; meaning, then it can be infered that performing z in the manner w implies that x does not have y for the same interval.

; (This will be our meaning postulate)

all(x,y,Y,z,w,D,t1,t2)({EventTime(~Has(y,Y,x),t1,t2), EventTime(Does(z,degree-to-which(w,D),x),t1,t2), MeaningUnknown(w)} &=>

(EventTime(Does(z,degree-to-which(w,D),x),t1,t2) =>
EventTime(~Has(y,Y,x),t1,t2))).

wff12!: all(t2,t1,D,w,z,Y,y,x)({MeaningUnknown(w),EventTime(Does(z,degree-to-which(w,D),x),t1,t2),EventTime(~Has(y,Y,x),t1,t2)} &=> {EventTime(Does(z,degree-to-which(w,D),x),t1,t2) => EventTime(~Has(y,Y,x),t1,t2)})

CPU time : 0.01

:

; CASSIE READS THE PASSAGE:

; ==========

; "After operating slighty emvasally [in the red] for many months,

; the struggling young firm landed a large government contract."

;

; Part 1) "The struggling young firm operated slighty emvasally..."

; Note that we must constraing the struggling property to be in this time period

; since it might not be the case that the firm is always struggling.

Isa(firm1, concept-called(Firm))!

wff14!: Isa(firm1,concept-called(Firm))

CPU time : 0.04

: HasProperty(concept-called(young), firm1)!

wff18!: HasProperty(concept-called(young),firm1)

CPU time : 0.01

: EventTime(HasProperty(concept-called(struggling), firm1), time1, time2)!

wff19!: EventTime(HasProperty(concept-called(struggling),firm1),time1,time2)

CPU time : 0.26

:

; Part 2) "[The struggling young firm] landed a large government contract."

Isa(contract1, concept-called(governmentContract))!

wff25!: Isa(contract1,concept-called(governmentContract))

CPU time : 0.02

: HasProperty(concept-called(large), contract1)!

wff27!: HasProperty(concept-called(large),contract1)

CPU time : 0.04

:

; Now to combine parts 1 and 2 we need to introduce a theory of time:

; Use Before/After/Now theory and corresponding case frame.

;

; "Before...(Part 1)...for many months...[After]...(Part 2)."

Before(time1,time2)!

wff20!: Before(time1,time2)

CPU time : 0.04

: Before(time2,now)!

wff29!: Before(time1,now)

wff28!: Before(time2,now)

CPU time : 0.01

: TimeInterval(Before(time1,time2),amt-in-units(concept-called(many), concept-called(months)))!

wff33!: TimeInterval(Before(time1,time2),amt-in-units(concept-called(many),concept-called(months)))

CPU time : 0.01

:

EventTime(Does(concept-called(operate), degree-to-which(concept-called(emvasally), concept-called(slightly)),firm1), time1, time2)!

wff39!: EventTime(Does(concept-called(operate),degree-to-which(concept-called(emvasally),concept-called(slightly)),firm1),time1,time2)

CPU time : 0.22

: EventTime(Does(action-wrt(concept-called(land),contract1),concept-called(normal),firm1), time2, time2)!

wff57!: EventTime(~HasProperty(concept-called(struggling),firm1),time2,now)

wff56!: EventTime(Has(some,concept-called(money),firm1),time2,now)

wff55!: EventTime(Has(contract1,concept-called(governmentContract),firm1),time2,now) and Isa(contract1,concept-

called(governmentContract))

wff54!: EventTime(Has(contract1,concept-called(governmentContract),firm1),time2,now)

wff52!: EventTime(Does(action-wrt(concept-called(receive),contract1),concept-called(normal),firm1),time2,time2)

wff49!: EventTime(Does(action-wrt(concept-called(land),contract1),concept-called(normal),firm1),time2,time2)

wff43!: EventTime(~Has(some,concept-called(money),firm1),time1,time2)

CPU time : 0.05

:

; Cassie knows going into this that she does not know the meaning of "emvasally"

MeaningUnknown(concept-called(emvasally))!

wff58!: EventTime(Does(concept-called(operate),degree-to-which(concept-called(emvasally),concept-called(slightly)),firm1),time1,time2) => EventTime(~Has(some,concept-called(money),firm1),time1,time2)

wff43!: EventTime(~Has(some,concept-called(money),firm1),time1,time2)

wff40!: MeaningUnknown(concept-called(emvasally))

CPU time : 0.02

:

; Ask Cassie what she now knows:

; ==========

; We will do this first by asking whether the firm had money before landing the contract

; (ie while operating emvasally)

EventTime(~Has(some, concept-called(money),firm1),time1,time2)?

wff43!: EventTime(~Has(some,concept-called(money),firm1),time1,time2)

CPU time : 0.00

:

; Now we will ask the following question to determine if Cassie has deduced

; a reasonable conclusion about the meaning of emvasally:

; Does operating "emvasally" at a certain time imply a lack of money during that time?

(EventTime(Does(concept-called(operate),degree-to-which(concept-called(emvasally),concept-called(slightly)),firm1),time1,time2) => EventTime(~Has(some,concept-called(money),firm1),time1,time2))?

wff58!: EventTime(Does(concept-called(operate),degree-to-which(concept-called(emvasally),concept-called(slightly)),firm1),time1,time2) => EventTime(~Has(some,concept-called(money),firm1),time1,time2)

CPU time : 0.00

:

CPU time : 0.00

:

End of /home/csdue/tjburns/CSE_663/emvasally.demo demonstration.

CPU time : 0.94

:

References

1. Almeida, Michael J (1995)., Time in narratives. In Judith F. Duchan, Gail A. Bruder, and Lynne E. Hewitt, editors, *Deixis in Narrative: A Cognitive Science Perspective*, pages 159-189. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ.

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

3. McCarthy, John (1980), "Circumscription--A Form of Non-Monotonic Reasoning", *Artificial Intelligence* 13: 27-39, 171-172.

4. Napieralski, Scott (2002), “Noun Algorithm Case Frames”, [http://www.cse.buffalo.edu/~rapaport/CVA/CaseFrames/case-frames/]

5. Rapaport, William J., & Ehrlich, Karen (2000), “A Computational Theory of Vocabulary Acquisition”, in Lucja M. Iwanska & Stuart C. Shapiro (eds.), Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language (Menlo Park, CA/Cambridge, MA: AAAI Press/MIT Press): 5.