Representation of "Proximity" for Evaluation by a Contextual Vocabulary Acquisition Algorithm

Scott Napieralski
stn2@cse.buffalo.edu
CSE663 - Advanced Topics in Knowledge Representation

May 5, 2002

Abstract

The topic of this investigation was the representation of sentences containing the word "proximity" for use with an algorithm that performs contextual vocabulary acquisition (CVA). Once the passages were encoded, they were tested with the existing noun definition algorithm and the output of the algorithm was analyzed in an attempt to find areas where the algorithm was deficient and could be improved. Some of the suggested improvements have been implemented and the changes to the output of the algorithm are analyzed.

1 Introduction

The topic of this investigation was the representation of sentences containing the word "proximity" for use with an algorithm that performs contextual vocabulary acquisition (CVA). The main part of the work was to create a representation of several passages containing "proximity" in SNePS (the Semantic Network Processing System). These representations had the constraint that they must only use case frames which are recognized by Karen Ehrlich's contextual vocabulary acquisition algorithm. This constraint was imposed because the purpose of creating these SNePS representations was to provide them as input to Ehrlich's algorithm. Ehrlich's algorithm also required that a significant amount of background knowledge about the passages be encoded. This knowledge

was needed to provide a context from which the algorithm could attempt to derive the meaning of "proximity". Once the passages were encoded, they were given to Ehrlich's algorithm and the output of the algorithm was analyzed in an attempt to find areas where the algorithm was deficient and could be improved.

2 Previous Work

The process of encoding passages containing "proximity" was built on the study of "proximity" that was done by Valerie Raybold Yakich for CSE676 during the Fall 2001 semester. Yakich did a preliminary representation of three passages:

Galileo Passage

> *Galileo* recently flew just 120 miles (200 kilometers) above Europa, a proximity which allowed the spacecraft to take the most detailed pictures ever of the mysterious satellite. (Powell)

Archaeologist Passage

> Archaeologists who discovered and excavated the small cave in 1983 had assumed that the brittle black material on the skulls was asphalt because of its color and the site's proximity to the largest asphalt deposits in Israel. (Discover)

Moon Passage

> Because of its proximity, Earth's only natural satellite – the Moon – became the first celestial body to be visited by humans (in 1969). (McKnight)

Yakich's representations provided some interesting insights into the problem of representing proximity, specifically in the area of background knowledge. Unfortunately, her representations introduced

several new case frames. Since Ehrlich's algorithm does not recognize these new case frames, it was necessary for me to re-encode the passages so that they conform to the constraints of the algorithm. Specifically, the only case frames that are used in the new representation of the three passages are standard SNePS case frames (which appear in the case frame dictionary) or case frames that appear in Ehrlich's dissertation.

3 Representation of the Passages

3.1 Passage 1

> *Galileo* recently flew just 120 miles (200 kilometers) above Europa, a proximity which allowed the spacecraft to take the most detailed pictures ever of the mysterious satellite. (Powell)

In the course of representing the sentence using SNePSUL some compromises were made and the information that was represented in SNePS was not identical to the information described in the passage. The following is an English rendering of the SNePS representation of the passage (Figure 1).

> *Galileo* flew 200 kilometers above Europa, a proximity which allowed *Galileo* to take a detailed picture of Europa.

There are several omissions in this representation which are common to all three passages. First, any reference to the time at which the events in the passage occurred (in this case, the word "recently") has been removed. The time at which an event occurred is not relevant to the meaning of "proximity", so this omission should not have an adverse effect on the ability of Ehrlich's algorithm to generate a definition. Since the goal of this effort was to create passages to use with the definition

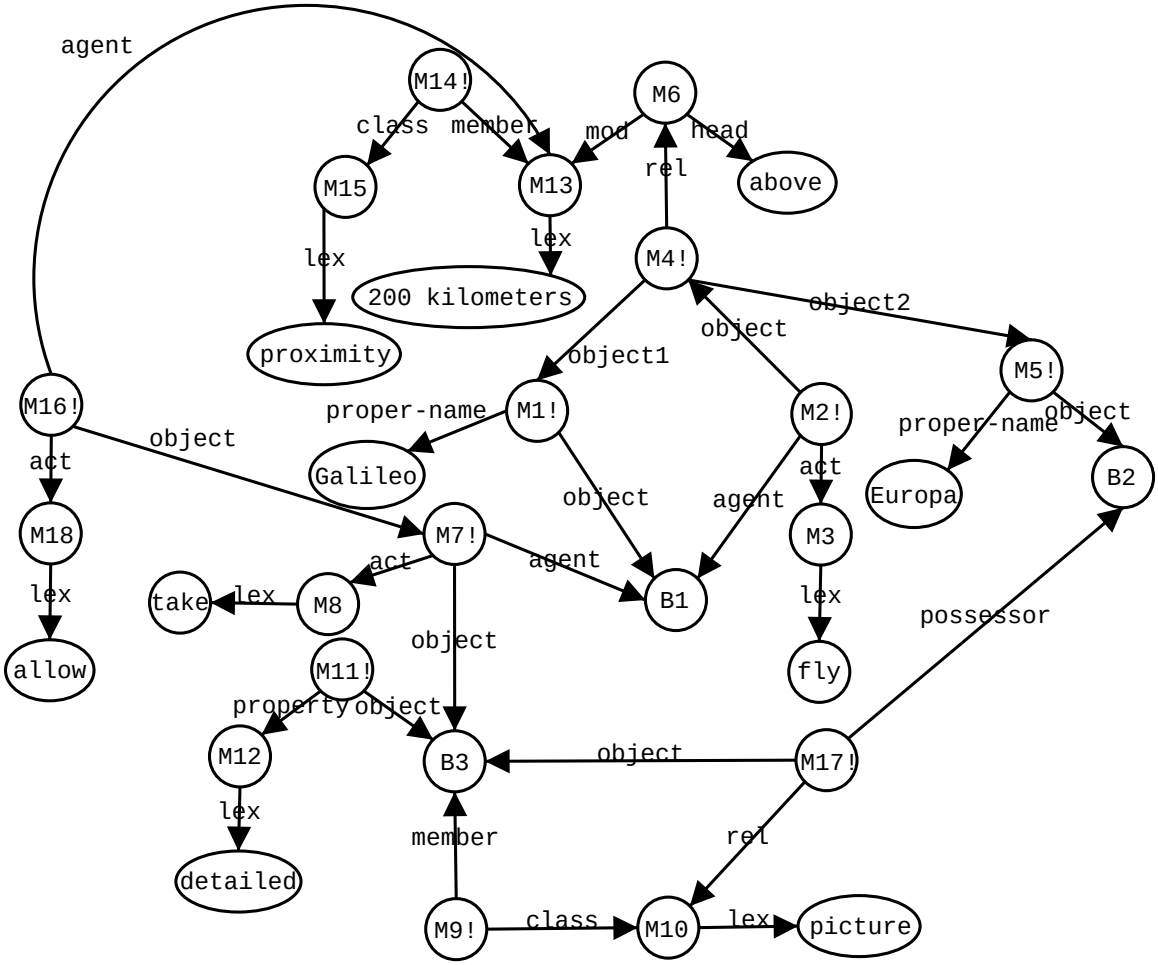

Figure 1: SNePS representation of the "Galileo" passage.

algorithm, I felt that it acceptable to leave time information out of this representation. However, if the representation were to be faithful to the idea that "proximity" is an unknown word then time information would need to be represented. Since the person interpreting the passage would have no idea what proximity meant, they would not be able to determine whether time information was relevant or not. For "proximity", it turns out that time does play a part in the definition, but for a word like "antique" time information might be central to the definition of the word.

In addition, some adjectives are omitted from the representation. The phrase "flew just 200 kilometers above" from the passage is represented as "flew 200 kilometers above". The word "just" and the implied information that 200 kilometers is a short distance in this context (which would probably take the form of background knowledge in the representation) has been left out. As in the case of time information, these adjectives were omitted because it was complicated to represent them and they were not valuable for Ehrlich's algorithm and its ability to generate a definition. The case of the omitted word "just" provides a good argument for spending the effort to encode adjectives (and the requisite background knowledge) in the future. When defining proximity it is probably useful to know that 200 kilometers (which is a proximity) is a short distance in this context. This information, coupled with the information that a more detailed picture can be taken if the object taking the picture is close to the subject of the picture, might lead a reader to define proximity as "a (possibly short) distance".

Finally, different words that refer to the same physical object were omitted. In this passage, the proper name *Galileo* has been substituted for the phrase "the spacecraft" and the proper name Europa has been substituted for the phrase "the mysterious satellite". These changes probably make the representation an inaccurate description of a person's mind, since the two things *Galileo* and "the spacecraft" could clearly be distinct intensional objects even though they refer to the same physical object. A more accurate representation would probably represent each phrase as a distinct

node and link the two with "equiv" or "synonym" arcs. However, when representing passages for the definition algorithm it is assumed that the definition of every word in the passage, except the unknown word, is known to the reader. Working under this assumption it is reasonable to assume that the reader would be able to make the connection between *Galileo* and "the spacecraft" and use the two interchangeably.

In addition to the issues described above, some decisions that affected only this passage bear explanation. In particular, the phrase "200 kilometers" was represented as a single node. There are many possible representations for this idea which are more complicated and try to represent the idea more precisely. However, having a more detailed representation would not add much to Cassie's understanding of the meaning of "200 kilometers". In the representation that was used, Cassie knows that "200 kilometers" is a proximity and background knowledge was added to say that "200 kilometers" is a distance. Another representation might break up the quantity "200" and the measurement "kilometers" so that kilometers could be further defined in background knowledge as 1000 meters. But a meter is simply another distance, with no more meaning to Cassie than a kilometer. Since the absolute measure of distance was not as important to the passage as the relative distance (*Galileo* was close to Europa) the representation of "200 kilometers" as a single node seemed sufficient.

3.2 Passage 2

> Archaeologists who discovered and excavated the small cave in 1983 had assumed that the brittle black material on the skulls was asphalt because of its color and the site's proximity to the largest asphalt deposits in Israel. (Discover)

As described in the previous section, the process of encoding the passage in SNePSUL does not capture all of the information that is in the passage. The following is an English approximation of

the SNePS representation of the passage, which is shown in Figure 2.

> An Archaeologist who discovered and excavated a small cave has assumed the brittle black material on a skull was asphalt because of the cave's proximity to a large asphalt deposit.

In this passage, nouns that were plural have been changed to singular. So "Archaeologists" has become "An archaeologist" and "the skulls" has become "a skull". This alteration probably has no effect on the results of Ehrlich's algorithm since the number of people who discover something and the number of objects that are discovered has no effect on the site, which is the object in which we are interested.

The information that the color of the material was a cause of the archaeologist's assumption is also not included in the SNePS representation of this passage. However, since the subject of the omitted phrase is the relation of the material to the assumption, it has no bearing on "proximity". Therefore, leaving this information out of the representation should not impair the ability of the algorithm to generate a definition.

3.3 Passage 3

> Because of its proximity, Earth's only natural satellite – the Moon – became the first celestial body to be visited by humans (in 1969). (McKnight)

The following is an English rendering of the SNePS representation of the passage, which is illustrated in Figure 3.

> Because of its proximity, Earth's natural satellite – the Moon (a celestial body) – was visited by a human first.

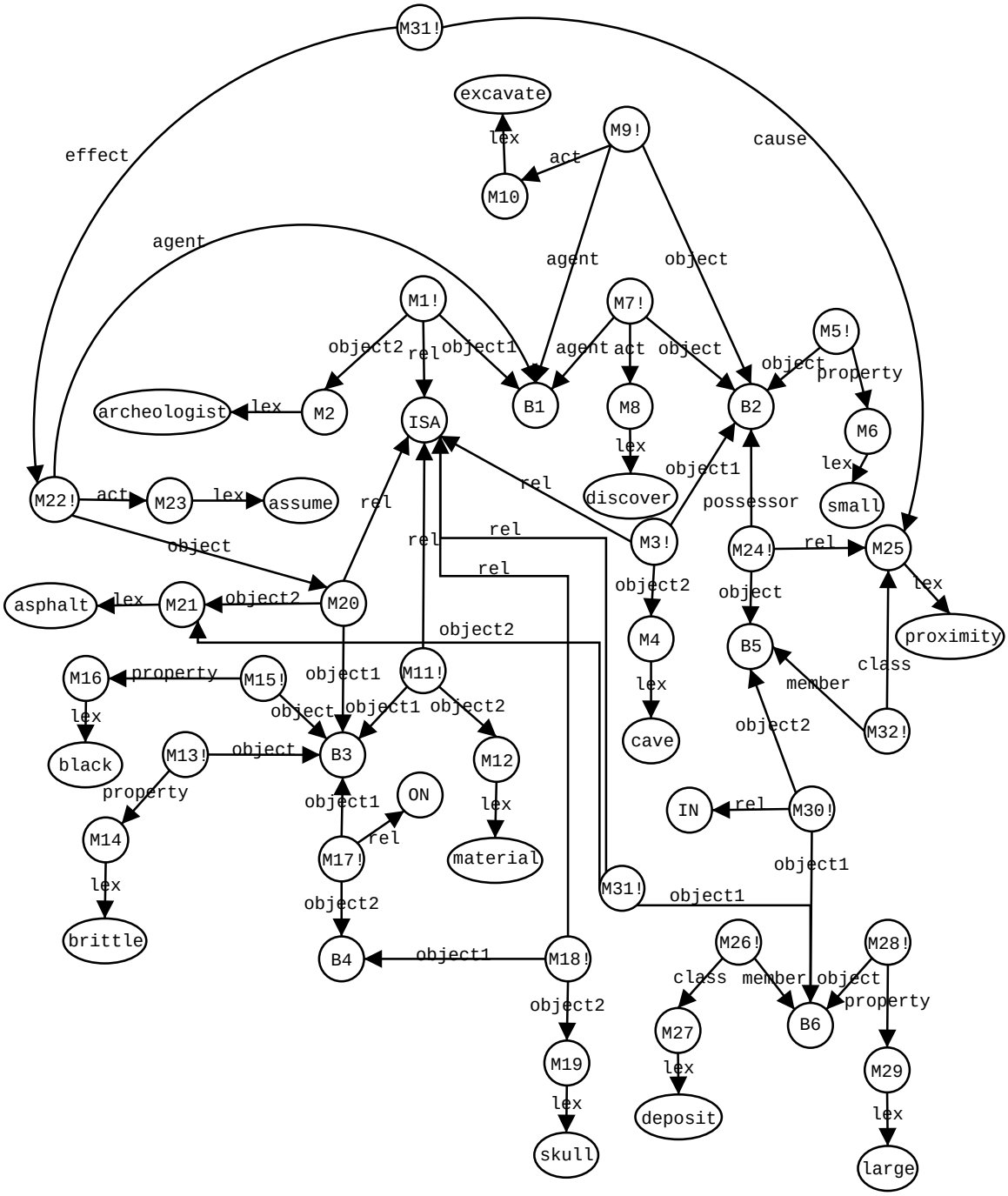

Figure 2: SNePS representation of the “Archaeologist” passage.

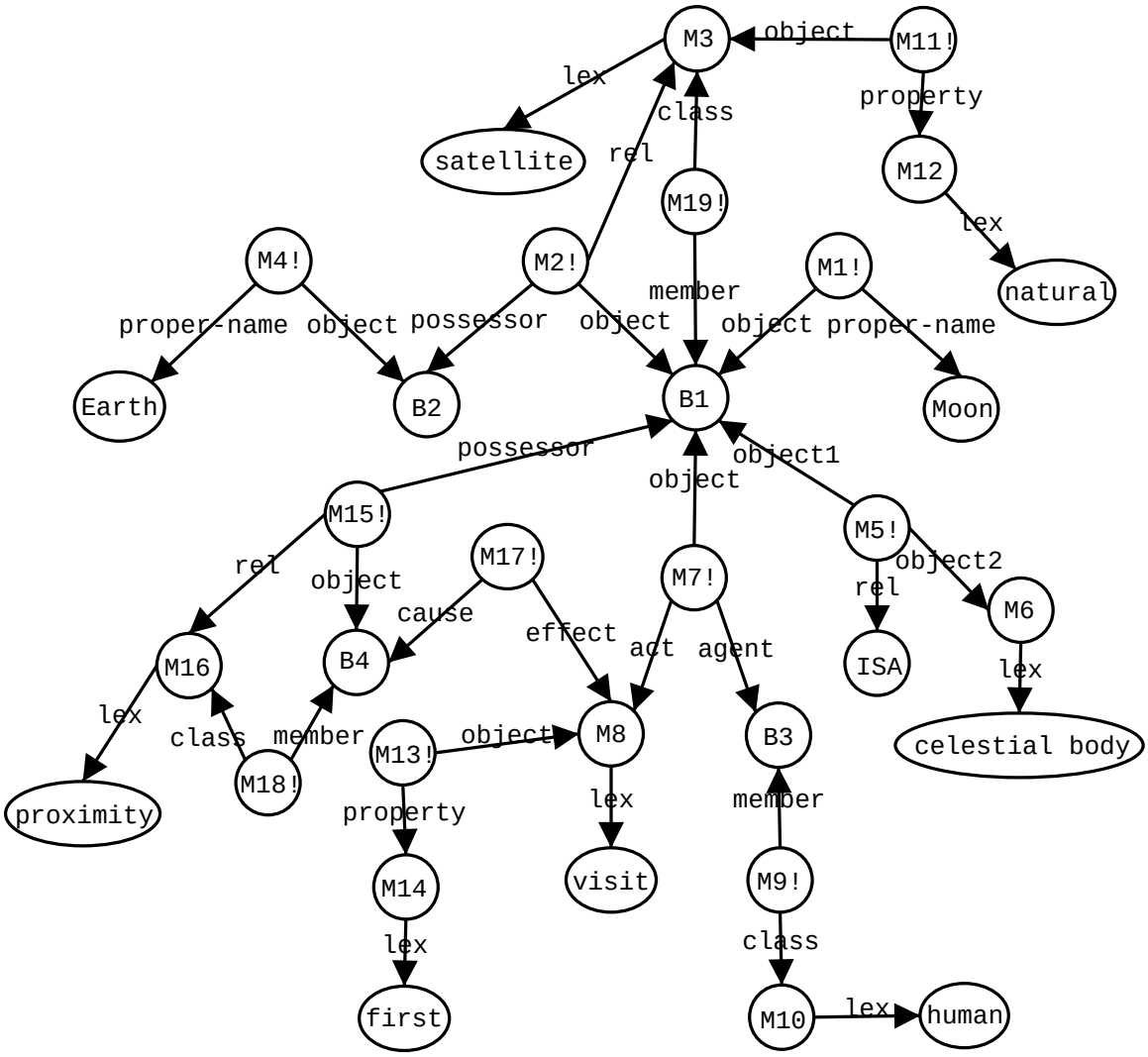

Figure 3: SNePS representation of the "Moon" passage.

The major issue with the representation of this passage is the representation of "first celestial body to be visited by humans". In the SNePS representation, this becomes "was visited by a human first". This representation accurately describes the fact that there was a visit to the Moon and that visit was the first visit by a human. However, it fails to capture the information that there were no other celestial bodies that were visited by a human before the moon was visited by a human. This information probably should be represented, since it should have some bearing on the definition of "proximity". If a person were given the information that some place was the first place of its type to be visited, it would be logical to conclude that the place is close to the starting location. That conclusion does not follow from the information that there was a first visit to a place. In the latter case, it is possible that several other places have been visited previous to the first visit to the place that is of interest.

This particular piece of information and the necessary background knowledge should be explored in the future. One possible representation is the following rule: "For all x, if x is a celestial body and x is not the moon then x was visited by humans after the Moon was visited by humans." Background knowledge concerning first visits might include the following: "If x was visited first and the visiting entity came from y then x is close to y." For this scheme to work, some modifications might need to be made to the agent-human-act-visit-object-Moon case frame so that the information that the visitor came from Earth could be represented in a standard way.

4 Modifications of the Algorithm

After each of the passages and the background knowledge to complement them was represented in SNePSUL, the algorithm was tested using the passages. Predictably, the first tests produced no definition. However, one piece of information generated by the definition was certainly incorrect.

As part of its definition, the algorithm was returning "A NIL is something that a proximity can onto". Since this information should obviously be included in the definition, I modified the function ag-act-fn (Figure 4) so that when no information could be found to fill in the slots in this sentence, nothing would be returned.

Next, I examined the representation of the *Galileo* passage to see what information the algorithm should be able to derive from it. The representation includes the information "200 kilometers is a member of the class proximity". At the very least, the algorithm should have replied "A proximity is something that 200 kilometers is". Upon further examination, the function there_exists seemed to be the source of this problem. Ehrlich's algorithm was first looking for non-basic level objects of type ¡noun¿ with a proper name, then basic level objects of type ¡noun¿ with a proper name. This was changed (Figure 6)this so that the algorithm looks for and reports basic level and non-basic level information, with a proper name or with a simple lex arc naming the object. All information that can be found is reported, not just the first pattern to be matched as in Ehrlich's version of the algorithm. This modification worked as hoped and there_exists began to return the information that "A proximity is something that 200 kilometers is."

While the modification to there_exists did improve the definition provided, the definition still seemed to be inadequate. On examining the representations more, it seemed that ownership information in the "Moon" passage and the "Archaeologist" passage was not being found. Specifically, this information was "A proximity can belong to a cave", "A proximity can belong to a satellite", and "A proximity can belong to a celestial object". Upon examination of the function that generates ownership information, indiv_rand_rels, it was discovered that the function was only attempting to find basic level information. In order to find the information in the "Archaeologist" and "Moon" passages, the non-basic level case was added to indiv_rand_rels.

After this modification was made, the algorithm was still returning unsatisfying results. Much

```
;-------------------------------------------------------------------------
;
;       function: ag-act-fn
;       input : a noun to be defined
;       returns : a list of the agent(s) and act(s) for which <noun>
;                 serves as object in an agent-act-object case frame.
;
;                                                       modified: mkb 04/2002
;-------------------------------------------------------------------------

(defun ag-act-fn (noun)
    (setq local-agent #3! ((find (compose lex- class- ! member agent- ! object
                                          member- ! class lex) ~noun)))
    (if (null local-agent)
        (setq local-agent #3! ((find (compose lex- class- ! member  agent- ! obj
ect object1- ! object2 lex)
                                                        ~noun))))
    (if (null local-agent)
        (setq local-agent #3! ((find (compose lex-  object2- object1 agent- ! ob
ject  member- ! class lex)                       ~noun))))
    (if (null local-agent)
        (setq local-agent #3! ((find (compose lex-  object2- object1 agent- ! ob
ject object1- ! object2 lex)
                         ~noun))))
    (if (null local-agent)
        (setq local-agent #3! ((find (compose lex-  object2- objects1 agent- ! o
bject member- ! class lex)                       ~noun))))
    (if (null local-agent)
        (setq local-agent #3! ((find (compose lex-  object2- objects1 agent- ! o
bject object1- ! object2 lex)                    ~noun))))

    (setq local-act #3! ((find (compose lex- act- ! object member- ! class lex)
 ~noun)))
    (if (null local-act)
        (setq local-act #3! ((find (compose lex- act- ! object object1- ! object
2 lex) ~noun))))
    (if (null local-agent)
        (setq local-agent #3! ((find (compose lex- class- ! member agent- ! onto
                         object1- ! object2 lex) ~noun))))
    (if (null local-agent)
        (setq local-agent #3! ((find (compose lex- object2- ! object1 agent- ! o
nto
                         object1- ! object2 lex) ~noun))))
    (if (null local-act)
        (setq local-act (append #3! ((find (compose lex- act- ! onto object1- !
object2 lex)                        ~noun))
                "onto")))

    ; if the last "find" didn't find anything local-act will be set to "onto"
    ; in that case, make local-act nil
    (if (not (listp local-act))
        (setq local-act nil))
    (if (or local-agent local-act)
        (list 'a noun 'is 'something 'a local-agent 'can local-act)
      nil))
```


Figure 4: New ag-act-fn.

```
;-------------------------------------------------------------------------
;       function: find_individuals
;       input:    a noun to be defined
;       output:   a list of individuals of type <noun>
;                                                        created: stn 2002
;-------------------------------------------------------------------------
(defun find_individuals (noun)
  "Returns a list of things that are a <noun>."
  ; are the lex arcs needed in a object,proper-name case frame???
  (append #3! ((find (compose lex- proper-name- ! object object1- ! object2 lex)
 ~noun
                     (compose lex- proper-name- ! object object1- ! rel) "ISA"))
          #3! ((find (compose lex- proper-name- ! object member- ! class lex) ~n
oun))
          #3! ((find (compose lex- object1- ! object2 lex) ~noun
                     (compose lex- object1- ! rel) "ISA"))
          #3! ((find (compose lex- member- ! class lex) ~noun))
   ))
```


Figure 5: New helper function for there_exists.

```
;-------------------------------------------------------------------------
;
;       function: there_exists
;       input:  a noun to be defined
;       output: a list of individuals of type <noun> together with any
;               possessions, functions, actions, relations, or other
;               properties attributed to those individuals.  If individuals
;               exist, and have such properties, but aren't named, list
;               the properties anyway.
;                                                          modified: mkb 2002
;-------------------------------------------------------------------------
(defun there_exists (noun)
  (setq agent-act (ag-act-fn noun))
  (setq str (struct noun nil))
  (setq ac (acts noun))
  (setq fun (func noun))
  (cond ((setq thex1 (find_individuals noun))
         (if (and (null ac) agent-act)
             (list agent-act
                   'a noun 'is 'something
                   thex1
                   'is.
                   'structure= str
                   'function= fun
                   'actions= 'nil
                   'ownership= (indiv_rand_rels noun)
                   'possible 'properties= (indiv_rand_props noun))
           (list 'a noun 'is 'something
                 thex1
                 'is.
                 'structure= str
                 'function= fun
                 'actions= ac
                 'ownership= (indiv_rand_rels noun)
                 'possible 'properties= (indiv_rand_props noun)))
          )
         ;; if there is some object1 but we don't know its name
         (#3! ((find (compose object1- ! object2 lex) ~noun))
              (if (and (null str) (null fun) agent-act)
                  (list agent-act
                        'structure= 'nil
                        'function= 'nil
                        'actions= ac
                        'ownership= (indiv_rand_rels noun)
                        'possible 'properties= (indiv_rand_props noun))
                (list 'structure= str
                      'function= fun
                      'actions= ac
                      'ownership= (indiv_rand_rels noun)
                      'possible 'properties= (indiv_rand_props noun))))
         ;; if there is some member of the class but we don't know its name
         (#3! ((find (compose member- ! class lex) ~noun))
              (if (and (null str) (null fun) agent-act)
                  (list agent-act
                        'structure= 'nil
                        'function= 'nil
                        'actions= ac
                        'ownership= (indiv_rand_rels noun)
                        'possible 'properties= (indiv_rand_props noun))
                (list 'structure= str
                      'function= fun
                      'actions= ac
                      'ownership= (indiv_rand_rels noun)
                      'possible 'properties= (indiv_rand_props noun))))
         ))
```


Figure 6: New there_exists.

of Ehrlich's code searches for various patterns and returns the results of the first matching pattern, ignoring any other possible matches. The method lead the algorithm to report "A proximity can belong to a satellite" because "satellite" is basic level and that was the first pattern indiv_rand_rels searched for that pattern first. However, the information that "A proximity can belong to a celestial object" was not being reported because "celestial object" is a non-basic level category. While the method that Ehrlich uses may produce a more accurate model of the an adult human brain at work, the aims of this project favor reporting as much information as possible. A child who is just learning how to determine the meaning of a word from context, would probably benefit from considering all the relevant information. Therefore, I further modified indiv_rand_rels (Figure 7) to report both basic and non-basic level ownership information. This modification produced the best result, it reported all the ownership information that was in the representation of the passages.

5 Future Work

In the future, researchers working on "proximity" should encode several more passages to determine if it is possible to learn more about this word from context. A word such as "proximity" provides a unique challenge to the algorithm as it currently stands. Several of the methods used by the algorithm to find a definition, such as listing structure and functions, do not apply to this word because it is not a physical object so it does not have any structure or functions. It may be that the algorithm does the best possible job of generating a definition, but it may also be that there are additional types of information that apply to nouns which are not physical objects.

Researchers continuing to work on improvements to the algorithm should begin by altering the report_basic function, and possibly all of the other report_ functions. The report_basic function does not report some of the information that is included in the report made by there_exists. Specifically,

```
;-------------------------------------------------------------------------
;
;       function: indiv_rand_rels
;       input:  a noun to be defined
;       output: a list of those things which possess any object of type <noun>
;               Note: needs to be refined/expanded.
;
;  modified: stn 2002
;    -- added non-basic case
;    -- changed to return data from all matched cases -- not just first
;-------------------------------------------------------------------------
(defun indiv_rand_rels (noun)
  ; if there is something that can own a <noun>
  (if (setq owners
        ; find possessors of all forms and make a list of them all
        (append
         #3! ((find (compose lex- class- ! member possessor- ! rel lex) ~noun))
         #3! ((find (compose lex- object2- ! object1 possessor- rel lex) ~noun
                    (compose lex- object2- ! rel) "ISA"))))
      ; return the list of possessors in a human readable format
      (list 'a noun 'can 'belong 'to 'a owners)
    ; if there are no possessors, return nil
    nil)

  )
```


Figure 7: New indiv_rand_rels.

information of the form "A proximity is something that 200 kilometers is." This information can be very useful to a person who is trying to define a word, so it seems that it should be reported whenever it is known. Therefore, the code that generates this information should be included in report_basic, as well as in there_exists.

When more than one piece of information is generated for a specific category, as in the discussion of ownership information above, it might be useful to look for some common element to all of the results. For example, we know that a proximity can belong to a cave, a satellite, and a celestial object. It would be good for the algorithm to analyze these three things, determine that they are all physical objects, and infer that it is possible that a proximity is something that belongs to any physical object.

A Code

The revised code described in this paper can be found on the University at Buffalo's computer science systems:
/projects/stn2/CVA/define$_n$oun.lisp

The SNePSUL code for the three passages can be found at:
/projects/stn2/CVA/proximity/passage1.demo
/projects/stn2/CVA/proximity/passage2.demo
/projects/stn2/CVA/proximity/passage3.demo

A.1 Galileo Passage

```
;;; Load all valid relations
(intext "rels")

;;; Compose paths
(intext "paths")

;;; Load (updated) Ehrlich Algorithm
^(load "fastcode.lisp")

;;; Load Background Knowledge
(intext "background-knowledge.base")
```

```
;; There is an object called Galileo
(describe (add object #gal
        proper-name Galileo))

;; There is an object called Europa
(describe (add object #eur
        proper-name Europa))

;; Galileo flew 200 kilometers above europa
(describe (add agent *gal
        act (build lex fly)
        object (build head ABOVE
                  mod (build lex "200 kilometers"))))

;; There is some picture
(describe (add member #pic
        class picture))

;; Galileo took the picture
(describe (add agent *gal
        act (build lex take)
        object *pic))

;; The picture was detailed
(describe (add object *pic
        property detailed))

;; 200 kilometers is a proximity
(describe (add member (build lex "200 kilometers")
        class (build lex proximity)))

;; the proximity allowed the pictures to be taken
(describe (add agent (build lex "200 kilometers")
        act (build lex allow)
        object (build agent *gal
                  act (build lex take)
                  object *pic)))

; the pictures were pictures of Europa
(describe (add object *pic
        rel (build lex picture)
        possessor *eur))

;; what is the definition?
^(defn_noun 'proximity)
```

A.2 Archaeologist Passage

```
;;; Load all valid relations
(intext "rels")

;;; Load Ehrlich Algorithm
^(load "fastcode.lisp")

;; Load Ehrlich's Background Knowledge
;(intext "brachet.only")
;(intext "cat.base")

;;; Load Background Knowledge
(intext "background-knowledge.base")

;; There is some archeologist
(describe (add object1 #Arch
          rel ISA
          object2 (build lex archeologist)))

;; There is some cave
(describe (add object1 #Cave
          rel ISA
          object2 (build lex cave)))

;; The cave is small
(describe (add object *Cave
          property (build lex small)))

;; The archeologist discovered the small cave
(describe (add agent *Arch
          act (build lex discover)
          object *Cave))

;; The archeologist excavated the small cave
(describe (add agent *Arch
          act (build lex excavate)
          object *Cave))

;; There is some material
(describe (add object1 #Material
          rel ISA
          object2 (build lex material)))

;; The material is brittle
(describe (add object *Material
          property (build lex brittle)))
```

```
;; The material is black
(describe (add object *Material
          property (build lex black)))

;; There is some skull
(describe (add object1 #Skull
          rel ISA
          object2 (build lex skull)))

;; The material is on the skull
(describe (add object1 *Material
          rel ON
          object2 *Skull))

;; The archeologist assumed the material was asphalt
(describe (add agent *Arch
          act (build lex assume)
          object (build object1 *Material
                rel ISA
                object2 (build lex asphalt))))

;; The cave has a proximity
(describe (add object #Prox
          rel (build lex proximity)
          possessor *Cave))

;; The cave's proximity is a proximity
(describe (add member *Prox class (build lex proximity)))

;; There is a deposit
(describe (add member #Deposit
          class (build lex deposit)))

;; The deposit is large
(describe (add object *Deposit
          property (build lex large)))

;; The deposit is asphalt
(describe (add object1 *Deposit
          rel ISA
          object2 (build lex asphalt)))

;; The deposit is in the proximity of the cave.
(describe (add object1 *Deposit
          rel IN
          object2 *Prox))
```

```
;; The proximity caused the assumption
(describe (add cause *Prox
          effect (build agent *Arch
                 act (build lex assume)
                 object (build object1 *Material
                           rel ISA
                           object2 (build lex asphalt)))))

;; what is the definition?
^(defn_noun 'proximity)
```

A.3 Moon Passage

```
;;; Load all valid relations
(intext "rels")

;;; Load Ehrlich Algorithm
^(load "fastcode.lisp")

;; Load Ehrlich's Background Knowledge
;(intext "brachet.only")
;(intext "cat.base")

;;; Load Background Knowledge
(intext "background-knowledge.base")

;; there is an object called the Moon
(describe (add object #Moon proper-name Moon))

;; there is an object called the Earth
(describe (add object #Earth proper-name Earth))

;; The moon is a satellite
(describe (add member *Moon class (build lex satellite)))

;; the moon is Earth's natural satellite
(describe (add object *Moon
     rel (build lex satellite)
     possessor *Earth))

;; it is a natural satellite
(describe (add object (build lex satellite)
        property (build lex natural)))

;; the moon is a celestial body
```

```
(describe (add object1 *Moon
          rel ISA
          object2 (build lex "celestial body")))

;; the moon was first visited by a human
(describe (add agent (build member #manOnMoon
                    class  (build lex human))
          act (build mod visit head first)
          object *Moon))

;; the moon has a proximity
(describe (add object #ProximityOfMoon
          rel (build lex proximity)
          possessor *Moon))

;; the moon's proximity is a member of the class proximity
(describe (add member *ProximityOfMoon
           class (build lex proximity)))

;; the cause of the visit was the proximity
(describe (add cause *ProximityOfMoon
          effect (build agent (build member #manOnMoon
                              class  (build lex human))
                act (build mod visit head first)
                object *Moon)))

;; what is the definition?
^(defn_noun 'proximity)
```

B Sample Runs

B.1 Galileo Passage

```
* (demo "passage1.demo")

File /home/stn2/classes/cse663/project/code/passage1.demo is now the source of input.


 CPU time : 0.01

* ;;; Load all valid relations
(intext "rels")
File rels is now the source of input.


 CPU time : 0.01
```

```
* ACT is already defined.
ACTION is already defined.
EFFECT is already defined.
OBJECT1 is already defined.
OBJECT2 is already defined.

(A1 A2 A3 A4 ACT ACTION AFTER AGENT ANTONYM ASSOCIATED BEFORE CAUSE CLASS
 DIRECTION EFFECT EQUIV ETIME FROM HEAD IN INDOBJ INSTR INTO LEX LOCATION
 KN_CAT MANNER MEMBER MEMBERS MOD MODE OBJECT OBJECTS OBJECT1 OBJECTS1 OBJECT2
 ON ONTO PART PLACE POSSESSOR PROPER-NAME PROPERTY PURPOSE REL SKF STIME
 SUBCLASS SUPERCLASS SYNONYM TIME TO WHOLE)

 CPU time : 0.06

*

End of file rels


 CPU time : 0.07

*
;;; Compose paths
(intext "paths")
File paths is now the source of input.


 CPU time : 0.01

*
BEFORE implied by the path (COMPOSE BEFORE (KSTAR (COMPOSE AFTER- ! BEFORE)))
BEFORE- implied by the path (COMPOSE (KSTAR (COMPOSE BEFORE- ! AFTER)) BEFORE-)


 CPU time : 0.00

*
AFTER implied by the path (COMPOSE AFTER (KSTAR (COMPOSE BEFORE- ! AFTER)))
AFTER- implied by the path (COMPOSE (KSTAR (COMPOSE AFTER- ! BEFORE)) AFTER-)


 CPU time : 0.00

*
SUB1 implied by the path (COMPOSE OBJECT1- SUPERCLASS- ! SUBCLASS SUPERCLASS-
                          ! SUBCLASS)
SUB1- implied by the path (COMPOSE SUBCLASS- ! SUPERCLASS SUBCLASS- !
```

```
                          SUPERCLASS OBJECT1)


 CPU time : 0.00

*
SUPER1 implied by the path (COMPOSE SUPERCLASS SUBCLASS- ! SUPERCLASS OBJECT1-
                            ! OBJECT2)
SUPER1- implied by the path (COMPOSE OBJECT2- ! OBJECT1 SUPERCLASS- ! SUBCLASS
                             SUPERCLASS-)


 CPU time : 0.00

*
SUPERCLASS implied by the path (OR SUPERCLASS SUPER1)
SUPERCLASS- implied by the path (OR SUPERCLASS- SUPER1-)


 CPU time : 0.01

*

End of file paths


 CPU time : 0.03

*
;;; Load (updated) Ehrlich Algorithm
^(
--> load "fastcode.lisp")
; Loading /home/stn2/classes/cse663/project/code/fastcode.lisp
T



 CPU time : 0.88

*
;;; Load Background Knowledge
(intext "background-knowledge.base")
File background-knowledge.base is now the source of input.


 CPU time : 0.00
```

```
*
(M2! (FORALL V2 V1)
 (&ANT (P3 (CLASS V1) (MEMBER V2)) CQ (P2 (OBJECT V2) (REL V1))
  (P1 (OBJECT1 V1) (OBJECT2 (M1 (LEX basic ctgy))) (REL ISA))))

(M2!)

 CPU time : 0.01

*
(M4! (OBJECT1 (M3 (LEX PICTURE))) (OBJECT2 (M1 (LEX basic ctgy))) (REL ISA))

(M4!)

 CPU time : 0.01

*
(M6! (OBJECT1 (M5 (LEX PROXIMITY))) (OBJECT2 (M1 (LEX basic ctgy))) (REL ISA))

(M6!)

 CPU time : 0.02

*
(M9! (FORALL V3)
 (ANT (P5 (OBJECT1 (P4 (LEX V3))) (OBJECT2 (M7 (LEX SPACECRAFT))) (REL ISA)))
 (CQ (P6 (CLASS (M8 (LEX VEHICLE))) (MEMBER (P4)))))

(M9!)

 CPU time : 0.02

*
(M10! (OBJECT1 (M8 (LEX VEHICLE))) (OBJECT2 (M1 (LEX basic ctgy))) (REL ISA))

(M10!)

 CPU time : 0.01

*
(M11! (FORALL V3)
 (ANT (P5 (OBJECT1 (P4 (LEX V3))) (OBJECT2 (M7 (LEX SPACECRAFT))) (REL ISA)))
 (CQ (P7 (ACT FLY) (AGENT V3))))

(M11!)

 CPU time : 0.00
```

```
*
(M13! (FORALL V3)
 (ANT (P5 (OBJECT1 (P4 (LEX V3))) (OBJECT2 (M7 (LEX SPACECRAFT))) (REL ISA)))
 (CQ (P8 (OBJECT1 (P4)) (OBJECT2 (M12 (LEX SPACE))) (REL IN))))

(M13!)

 CPU time : 0.01

*
(M15! (OBJECT1 (M14 (LEX GALILEO))) (OBJECT2 (M7 (LEX SPACECRAFT))) (REL ISA))

(M15!)

 CPU time : 0.01

*
(M17! (FORALL V4)
 (ANT (P10 (OBJECT1 (P9 (LEX V4))) (OBJECT2 (M12 (LEX SPACE))) (REL IN)))
 (CQ
  (P12 (MIN 0) (MAX 0)
   (ARG (P11 (OBJECT1 (P9)) (OBJECT2 (M16 (LEX EARTH))) (REL ON))))))

(M17!)

 CPU time : 0.02

*
(M18! (FORALL V4)
 (ANT (P11 (OBJECT1 (P9 (LEX V4))) (OBJECT2 (M16 (LEX EARTH))) (REL ON)))
 (CQ
  (P13 (MIN 0) (MAX 0)
   (ARG (P10 (OBJECT1 (P9)) (OBJECT2 (M12 (LEX SPACE))) (REL IN))))))

(M18!)

 CPU time : 0.01

*
(M20! (OBJECT1 (M19 (LEX SATELLITE))) (OBJECT2 (M1 (LEX basic ctgy)))
 (REL ISA))

(M20!)

 CPU time : 0.02
```

```
*
(M22! (CLASS (M19 (LEX SATELLITE))) (MEMBER (M21 (LEX EUROPA))))

(M22!)

 CPU time : 0.02

*
(M23! (OBJECT1 (M21 (LEX EUROPA))) (OBJECT2 (M12 (LEX SPACE))) (REL IN))

(M23!)

 CPU time : 0.01

*
(M26! (OBJECT1 (M24 (LEX 200 kilometers))) (OBJECT2 (M25 (LEX DISTANCE)))
 (REL ISA))

(M26!)

 CPU time : 0.01

*
(M28! (FORALL V6 V5)
 (&ANT
  (P18 (OBJECT1 (P16 (LEX V5))) (OBJECT2 (P17 (LEX V6)))
   (REL (M27 (HEAD ABOVE) (MOD (M24 (LEX 200 kilometers))))))
  (P15 (OBJECT1 V6) (OBJECT2 SPACE) (REL IN))
  (P14 (OBJECT1 V5) (OBJECT2 SPACE) (REL IN)))
 (CQ (P19 (OBJECT1 V5) (OBJECT2 V6) (REL NEAR))))

(M28!)

 CPU time : 0.11

*
(M29! (FORALL V7) (ANT (P20 (CLASS PICTURE) (MEMBER V7)))
 (CQ (P21 (OBJECT1 V7) (OBJECT2 phys obj) (REL ISA))))

(M29!)

 CPU time : 0.01

*
(M31! (OBJECT1 (M30 (LEX DEPOSIT))) (OBJECT2 (M1 (LEX basic ctgy))) (REL ISA))

(M31!)
```

```
 CPU time : 0.01

*
(M34! (FORALL V8)
 (ANT (P22 (OBJECT1 V8) (OBJECT2 (M32 (LEX ARCHEOLOGIST))) (REL ISA)))
 (CQ (P23 (CLASS (M33 (LEX HUMAN))) (MEMBER V8))))

(M34!)

 CPU time : 0.01

*
(M35! (OBJECT1 (M33 (LEX HUMAN))) (OBJECT2 (M1 (LEX basic ctgy))) (REL ISA))

(M35!)

 CPU time : 0.00

*
(M36! (FORALL V9) (ANT (P24 (ACT STUDY) (AGENT V9) (OBJECT ARTIFACT)))
 (CQ (P25 (OBJECT1 V9) (OBJECT2 (M32 (LEX ARCHEOLOGIST))) (REL ISA))))

(M36!)

 CPU time : 0.02

*
(M37! (FORALL V10) (ANT (P26 (OBJECT1 V10) (OBJECT2 SKULL) (REL ISA)))
 (CQ (P27 (OBJECT1 V10) (OBJECT2 ARTIFACT) (REL ISA))))

(M37!)

 CPU time : 0.02

*
(M38! (SUBCLASS EXCAVATE) (SUPERCLASS DIG))

(M38!)

 CPU time : 0.00

*
(M39! (FORALL V11) (ANT (P28 (OBJECT V11) (PROPERTY SMALL)))
 (CQ (P30 (MIN 0) (MAX 0) (ARG (P29 (OBJECT V11) (PROPERTY LARGE))))))

(M39!)
```

```
 CPU time : 0.01

*
(M40! (FORALL V12) (ANT (P31 (OBJECT V12) (PROPERTY LARGE)))
 (CQ (P33 (MIN 0) (MAX 0) (ARG (P32 (OBJECT V12) (PROPERTY SMALL))))))

(M40!)

 CPU time : 0.02

*
(M41! (FORALL V13) (ANT (P34 (OBJECT1 V13) (OBJECT2 CAVE) (REL ISA)))
 (CQ (P35 (OBJECT1 V13) (OBJECT2 phys obj) (REL ISA))))

(M41!)

 CPU time : 0.01

*
(M42! (CLASS COLOR) (MEMBER BLACK))

(M42!)

 CPU time : 0.01

*
(M43! (FORALL V14) (ANT (P36 (OBJECT V14) (PROPERTY COLOR)))
 (CQ (P37 (OBJECT1 V14) (OBJECT2 phys obj) (REL ISA))))

(M43!)

 CPU time : 0.01

*
(M44! (FORALL V15) (ANT (P38 (OBJECT1 V15) (OBJECT2 ASPHALT) (REL ISA)))
 (CQ (P39 (CLASS STONE) (MEMBER V15))))

(M44!)

 CPU time : 0.01

*
(M45! (FORALL V16) (ANT (P40 (CLASS STONE) (MEMBER V16)))
 (CQ (P41 (OBJECT1 V16) (OBJECT2 phys obj) (REL ISA))))

(M45!)
```

```
 CPU time : 0.03

*
(M46! (FORALL V17) (ANT (P42 (OBJECT V17) (PROPERTY BRITTLE)))
 (CQ (P43 (OBJECT V17) (PROPERTY BREAKABLE))))

(M46!)

 CPU time : 0.01

*
(M47! (FORALL V18) (ANT (P44 (OBJECT1 V18) (OBJECT2 DEPOSIT) (REL ISA)))
 (CQ (P45 (OBJECT1 V18) (OBJECT2 phys obj) (REL ISA))))

(M47!)

 CPU time : 0.01

*
(M49! (OBJECT1 (M48 (LEX MOON))) (OBJECT2 (M12 (LEX SPACE))) (REL IN))

(M49!)

 CPU time : 0.01

*
(M50! (OBJECT1 (M16 (LEX EARTH))) (OBJECT2 (M12 (LEX SPACE))) (REL IN))

(M50!)

 CPU time : 0.01

*
(M51! (FORALL V19) (ANT (P46 (CLASS HUMAN) (MEMBER V19)))
 (CQ (P47 (OBJECT1 V19) (OBJECT2 EARTH) (REL ON))))

(M51!)

 CPU time : 0.01

*
(M52! (FORALL V20) (ANT (P48 (OBJECT1 V20) (OBJECT2 EARTH) (REL ON)))
 (CQ (P50 (MIN 0) (MAX 0) (ARG (P49 (OBJECT1 V20) (OBJECT2 SPACE) (REL IN))))))

(M52!)
```

```
 CPU time : 0.02

*
(M53! (FORALL V21) (ANT (P51 (OBJECT1 V21) (OBJECT2 SPACE) (REL IN)))
 (CQ (P53 (MIN 0) (MAX 0) (ARG (P52 (OBJECT1 V21) (OBJECT2 EARTH) (REL ON))))))

(M53!)

 CPU time : 0.01

*

End of file background-knowledge.base


 CPU time : 0.65

*
;; There is an object called Galileo
(describe (add object #gal
        proper-name Galileo))

(M54! (OBJECT B1) (PROPER-NAME GALILEO))

(M54!)

 CPU time : 0.03

*
;; There is an object called Europa
(describe (add object #eur
        proper-name Europa))

(M55! (OBJECT B2) (PROPER-NAME EUROPA))

(M55!)

 CPU time : 0.02

*
;; Galileo flew 200 kilometers above europa
(describe (add agent *gal
        act (build lex fly)
        object (build head ABOVE
                  mod (build lex "200 kilometers"))))

(M57! (ACT (M56 (LEX FLY))) (AGENT B1)
```

```
 (OBJECT (M27 (HEAD ABOVE) (MOD (M24 (LEX 200 kilometers))))))

(M57!)

 CPU time : 0.03

*
;; There is some picture
(describe (add member #pic
        class picture))

Since
((M29! (FORALL V7) (ANT (P20 (CLASS (PICTURE)) (MEMBER V7)))
  (CQ (P21 (OBJECT1 V7) (OBJECT2 (phys obj)) (REL (ISA))))))
and
((P20 (CLASS (PICTURE)) (MEMBER (V7 <-- B3))))
I infer
((P21 (OBJECT1 (V7 <-- B3)) (OBJECT2 (phys obj)) (REL (ISA))))

(M59! (OBJECT1 B3) (OBJECT2 phys obj) (REL ISA))
(M58! (CLASS PICTURE) (MEMBER B3))

(M59! M58!)

 CPU time : 0.06

*
;; Galileo took the picture
(describe (add agent *gal
        act (build lex take)
        object *pic))

(M61! (ACT (M60 (LEX TAKE))) (AGENT B1) (OBJECT B3))

(M61!)

 CPU time : 0.02

*
;; The picture was detailed
(describe (add object *pic
        property detailed))

(M62! (OBJECT B3) (PROPERTY DETAILED))

(M62!)
```

```
 CPU time : 0.02

*
;; 200 kilometers is a proximity
(describe (add member (build lex "200 kilometers")
        class (build lex proximity)))

(M63! (CLASS (M5 (LEX PROXIMITY))) (MEMBER (M24 (LEX 200 kilometers))))

(M63!)

 CPU time : 0.02

*
;; the proximity allowed the pictures to be taken
(describe (add agent (build lex "200 kilometers")
        act (build lex allow)
        object (build agent *gal
                  act (build lex take)
                  object *pic)))
(M65! (ACT (M64 (LEX ALLOW))) (AGENT (M24 (LEX 200 kilometers)))
 (OBJECT (M61! (ACT (M60 (LEX TAKE))) (AGENT B1) (OBJECT B3))))

(M65!)

 CPU time : 0.03

*

; the pictures were pictures of Europa
(describe (add object *pic
        rel (build lex picture)
        possessor *eur))

(M67! (OBJECT B3) (REL (M3 (LEX PICTURE))))
(M66! (OBJECT B3) (POSSESSOR B2) (REL (M3)))

(M67! M66!)

 CPU time : 0.03

*
;; what is the definition?
^(
--> defn_noun 'proximity)
(CLASS INCLUSIONS= NIL STRUCTURE= NIL FUNCTION= NIL ACTIONS=
 (POSSIBLE ACTIONS= (ALLOW)) OWNERSHIP= NIL POSSIBLE PROPERTIES= NIL SYNONYMS=
```

```
 NIL)

;; Note that the information "A proximity is something that 200 kilometer is"
;; is not included here, that information is produced by there_exists.
```

B.2 Archaeologist Passage

```
* (demo "passage2.demo")

File /home/stn2/classes/cse663/project/code/passage2.demo is now the source of input.


 CPU time : 0.01

* ;;; Load all valid relations
(intext "rels")
File rels is now the source of input.

 CPU time : 0.00

(A1 A2 A3 A4 ACT ACTION AFTER AGENT ANTONYM ASSOCIATED BEFORE CAUSE CLASS
 DIRECTION EFFECT EQUIV ETIME FROM HEAD IN INDOBJ INSTR INTO LEX LOCATION
 KN_CAT MANNER MEMBER MEMBERS MOD MODE OBJECT OBJECTS OBJECT1 OBJECTS1 OBJECT2
 ON ONTO PART PLACE POSSESSOR PROPER-NAME PROPERTY PURPOSE REL SKF STIME
 SUBCLASS SUPERCLASS SYNONYM TIME TO WHOLE)

 CPU time : 0.09

*

End of file rels


 CPU time : 0.09

*
;;; Load Ehrlich Algorithm
^(
--> load "fastcode.lisp")
; Loading /home/stn2/classes/cse663/project/code/fastcode.lisp
T

 CPU time : 0.90

*
;; There is some archeologist
(describe (add object1 #Arch
```

```
          rel ISA
          object2 (build lex archeologist)))

Since
((M34! (FORALL V8)
  (ANT (P22 (OBJECT1 V8) (OBJECT2 (M32 (LEX (ARCHEOLOGIST)))) (REL (ISA))))
  (CQ (P23 (CLASS (M33 (LEX (HUMAN)))) (MEMBER V8)))))
and
((P22 (OBJECT1 (V8 <-- B4)) (OBJECT2 (M32 (LEX (ARCHEOLOGIST)))) (REL (ISA))))
I infer
((P23 (CLASS (M33 (LEX (HUMAN)))) (MEMBER (V8 <-- B4))))

Since
((M77! (FORALL V32)
  (ANT (P80 (OBJECT1 V32) (OBJECT2 (M32 (LEX (ARCHEOLOGIST)))) (REL (ISA))))
  (CQ (P81 (CLASS (M33 (LEX (HUMAN)))) (MEMBER V32)))))
and
((P80 (OBJECT1 (V32 <-- B4)) (OBJECT2 (M32 (LEX (ARCHEOLOGIST)))) (REL (ISA))))
I infer
((P81 (CLASS (M33 (LEX (HUMAN)))) (MEMBER (V32 <-- B4))))

(M92! (CLASS (M33 (LEX HUMAN))) (MEMBER B4))
(M91! (OBJECT1 B4) (OBJECT2 (M32 (LEX ARCHEOLOGIST))) (REL ISA))

(M92! M91!)

 CPU time : 0.22

*
;; There is some cave
(describe (add object1 #Cave
          rel ISA
          object2 (build lex cave)))

(M94! (OBJECT1 B5) (OBJECT2 (M93 (LEX CAVE))) (REL ISA))

(M94!)

 CPU time : 0.04

*
;; The cave is small
(describe (add object *Cave
          property (build lex small)))

(M96! (OBJECT B5) (PROPERTY (M95 (LEX SMALL))))
```

```
(M96!)

 CPU time : 0.04

*
;; The archeologist discovered the small cave
(describe (add agent *Arch
          act (build lex discover)
          object *Cave))

(M98! (ACT (M97 (LEX DISCOVER))) (AGENT B4) (OBJECT B5))

(M98!)

 CPU time : 0.04

*
;; The archeologist excavated the small cave
(describe (add agent *Arch
          act (build lex excavate)
          object *Cave))

(M100! (ACT (M99 (LEX EXCAVATE))) (AGENT B4) (OBJECT B5))

(M100!)

 CPU time : 0.03

*
;; There is some material
(describe (add object1 #Material
          rel ISA
          object2 (build lex material)))

(M102! (OBJECT1 B6) (OBJECT2 (M101 (LEX MATERIAL))) (REL ISA))

(M102!)

 CPU time : 0.05

*
;; The material is brittle
(describe (add object *Material
          property (build lex brittle)))

(M104! (OBJECT B6) (PROPERTY (M103 (LEX BRITTLE))))
```

```
(M104!)

 CPU time : 0.04

*
;; The material is black
(describe (add object *Material
          property (build lex black)))

(M106! (OBJECT B6) (PROPERTY (M105 (LEX BLACK))))

(M106!)

 CPU time : 0.04

*
;; There is some skull
(describe (add object1 #Skull
          rel ISA
          object2 (build lex skull)))

(M108! (OBJECT1 B7) (OBJECT2 (M107 (LEX SKULL))) (REL ISA))

(M108!)

 CPU time : 0.12

*
;; The material is on the skull
(describe (add object1 *Material
          rel ON
          object2 *Skull))

(M109! (OBJECT1 B6) (OBJECT2 B7) (REL ON))

(M109!)

 CPU time : 0.05

*
;; The archeologist assumed the material was asphalt
(describe (add agent *Arch
          act (build lex assume)
          object (build object1 *Material
                rel ISA
                object2 (build lex asphalt))))
```

```
(M113! (ACT (M110 (LEX ASSUME))) (AGENT B4)
 (OBJECT (M112 (OBJECT1 B6) (OBJECT2 (M111 (LEX ASPHALT))) (REL ISA))))

(M113!)

 CPU time : 0.05

*
;; The cave has a proximity
(describe (add object #Prox
          rel (build lex proximity)
          possessor *Cave))

(M115! (OBJECT B8) (REL (M5 (LEX PROXIMITY))))
(M114! (OBJECT B8) (POSSESSOR B5) (REL (M5)))

(M115! M114!)

 CPU time : 0.04

*
;; The cave's proximity is a proximity
(describe (add member *Prox class (build lex proximity)))

(M116! (CLASS (M5 (LEX PROXIMITY))) (MEMBER B8))

(M116!)

 CPU time : 0.03

*
;; There is a deposit
(describe (add member #Deposit
          class (build lex deposit)))

(M117! (CLASS (M30 (LEX DEPOSIT))) (MEMBER B9))

(M117!)

 CPU time : 0.05

*
;; The deposit is large
(describe (add object *Deposit
          property (build lex large)))

(M119! (OBJECT B9) (PROPERTY (M118 (LEX LARGE))))
```

```
(M119!)

 CPU time : 0.05

*
;; The deposit is asphalt
(describe (add object1 *Deposit
          rel ISA
          object2 (build lex asphalt)))

(M120! (OBJECT1 B9) (OBJECT2 (M111 (LEX ASPHALT))) (REL ISA))

(M120!)

 CPU time : 0.12

*
;; The deposit is in the proximity of the cave.
(describe (add object1 *Deposit
          rel IN
          object2 *Prox))

(M121! (OBJECT1 B9) (OBJECT2 B8) (REL IN))

(M121!)

 CPU time : 0.04

*
;; The proximity caused the assumption
(describe (add cause *Prox
          effect (build agent *Arch
                act (build lex assume)
                object (build object1 *Material
                          rel ISA
                          object2 (build lex asphalt)))))

(M122! (CAUSE B8)
 (EFFECT (M113! (ACT (M110 (LEX ASSUME))) (AGENT B4)
          (OBJECT
           (M112 (OBJECT1 B6) (OBJECT2 (M111 (LEX ASPHALT))) (REL ISA))))))

(M122!)

 CPU time : 0.04
```

```
*
;; what is the definition?
^(
--> defn_noun 'proximity)
(CLASS INCLUSIONS= NIL STRUCTURE= NIL FUNCTION= NIL ACTIONS=
 (POSSIBLE ACTIONS= (ALLOW)) OWNERSHIP= (A PROXIMITY CAN BELONG TO A (CAVE))
 POSSIBLE PROPERTIES= NIL SYNONYMS= NIL)
```

B.3 Moon Passage

```
* (demo "passage3.demo")

File /home/stn2/classes/cse663/project/code/passage3.demo is now the source of input.


 CPU time : 0.00

* ;;; Load all valid relations
(intext "rels")
File rels is now the source of input.

(A1 A2 A3 A4 ACT ACTION AFTER AGENT ANTONYM ASSOCIATED BEFORE CAUSE CLASS
 DIRECTION EFFECT EQUIV ETIME FROM HEAD IN INDOBJ INSTR INTO LEX LOCATION
 KN_CAT MANNER MEMBER MEMBERS MOD MODE OBJECT OBJECTS OBJECT1 OBJECTS1 OBJECT2
 ON ONTO PART PLACE POSSESSOR PROPER-NAME PROPERTY PURPOSE REL SKF STIME
 SUBCLASS SUPERCLASS SYNONYM TIME TO WHOLE)

 CPU time : 0.09

*

End of file rels


 CPU time : 0.11

*
;;; Load Ehrlich Algorithm
^(
--> load "fastcode.lisp")
; Loading /home/stn2/classes/cse663/project/code/fastcode.lisp
T

 CPU time : 0.90

*
;; there is an object called the Moon
```

```
(describe (add object #Moon proper-name Moon))

(M145! (OBJECT B10) (PROPER-NAME MOON))

(M145!)

 CPU time : 0.11

*
;; there is an object called the Earth
(describe (add object #Earth proper-name Earth))

(M146! (OBJECT B11) (PROPER-NAME EARTH))

(M146!)

 CPU time : 0.06

*
;; The moon is a satellite
(describe (add member *Moon class (build lex satellite)))

(M147! (CLASS (M19 (LEX SATELLITE))) (MEMBER B10))

(M147!)

 CPU time : 0.06

*
;; the moon is Earth's natural satellite
(describe (add object *Moon
     rel (build lex satellite)
     possessor *Earth))

(M149! (OBJECT B10) (REL (M19 (LEX SATELLITE))))
(M148! (OBJECT B10) (POSSESSOR B11) (REL (M19)))

(M149! M148!)

 CPU time : 0.07

*
;; it is a natural satellite
(describe (add object (build lex satellite)
        property (build lex natural)))

(M151! (OBJECT (M19 (LEX SATELLITE))) (PROPERTY (M150 (LEX NATURAL))))
```

```
(M151!)

 CPU time : 0.06

*
;; the moon is a celestial body
(describe (add object1 *Moon
          rel ISA
          object2 (build lex "celestial body")))

(M153! (OBJECT1 B10) (OBJECT2 (M152 (LEX celestial body))) (REL ISA))

(M153!)

 CPU time : 0.11

*
;; the moon was first visited by a human
(describe (add agent (build member #manOnMoon
                    class  (build lex human))
          act (build mod visit head first)
          object *Moon))

(M156! (ACT (M155 (HEAD FIRST) (MOD VISIT)))
 (AGENT (M154 (CLASS (M33 (LEX HUMAN))) (MEMBER B12))) (OBJECT B10))

(M156!)

 CPU time : 0.06

*
;; the moon has a proximity
(describe (add object #ProximityOfMoon
          rel (build lex proximity)
          possessor *Moon))

(M158! (OBJECT B13) (REL (M5 (LEX PROXIMITY))))
(M157! (OBJECT B13) (POSSESSOR B10) (REL (M5)))

(M158! M157!)

 CPU time : 0.05

*
;; the moon's proximity is a member of the class proximity
(describe (add member *ProximityOfMoon
```

```
          class (build lex proximity)))

(M159! (CLASS (M5 (LEX PROXIMITY))) (MEMBER B13))

(M159!)

 CPU time : 0.04

*
;; the cause of the visit was the proximity
(describe (add cause *ProximityOfMoon
          effect (build agent (build member #manOnMoon
                                class  (build lex human))
                act (build mod visit head first)
                object *Moon)))

(M162! (CAUSE B13)
 (EFFECT (M161 (ACT (M155 (HEAD FIRST) (MOD VISIT)))
          (AGENT (M160 (CLASS (M33 (LEX HUMAN))) (MEMBER B14))) (OBJECT B10))))

(M162!)

 CPU time : 0.07

*
;; what is the definition?
^(
--> defn_noun 'proximity)
(CLASS INCLUSIONS= NIL STRUCTURE= NIL FUNCTION= NIL ACTIONS=
 (POSSIBLE ACTIONS= (ALLOW)) OWNERSHIP=
 (A PROXIMITY CAN BELONG TO A (SATELLITE celestial body CAVE)) POSSIBLE
 PROPERTIES= NIL SYNONYMS= NIL)
```

References

[McKnight] McKnight, Tom L., and Darrel Hess (2000) *Physical Geography, a Landscape Appreciation.* Sixth Edition, Prentice-Hall, Upper Saddle River, New Jersey. p. 5.

[Powell] Powell, Corey S., (1998) "Europa Up Close" Discover Science News for Astronomy and Physics. Posted March 5th. Available on the web at *http* : $//www.discover.com/science_news/astroscience.html$

[Discover] "Strange Skulls," *Discover* (1998, February) Vol. 19, No 2. Available on the web at *http* : $//www.discover.com/cover_story/skull.html$.

[Yakich] Yakich, Valerie Raybold. (2001) *Contextual Vocabulary Acquisition of "Proximity"* Report for CSE676 – Knowledge Representation