Contextual Vocabulary Acquisition

Douglas Calderon

Contextual Vocabulary Acquisition is the process of deriving the meaning of words based on the surrounding context. To do this I used SNePS, which is a knowledge representation, reasoning, and acting system. Using these resources I have caused Cassie (our SNePS agent) to reason out the definition of Perambulate given its co-text.

University At Buffalo

Department of Computer Science and Engineering

Dmc42@buffalo.edu

5/12/2011

The Contextual Vocabulary Acquisition project is an attempt at combining many fields of expertise towards the common goal of deriving a meaning of a word given its context. This most directly has applications in developmental education. You would expect a student learning their first language to not understand a great many words. Teaching CVA can greatly increase the rate at which a child is able to attain new words and expand their vocabulary. This also has applications in learning English (or any other language) as a second language, as there will also probably be many words that are unknown to the reader, which they can then reason out using CVA.

SNePS is a knowledge representation, reasoning, and acting system that I use as the main tool to get Cassie (the SNePS agent) to attain the meaning of an unknown word. The application layer that I used to build the SNePS network is called SNePSUL, which is a syntax that isn't quite first order logic (which SNePSLOG imitates), but more of an attempt at a very simple level of English language.

My goal was to get Cassie to derive the meaning of the word Perambulate given the following Passage.

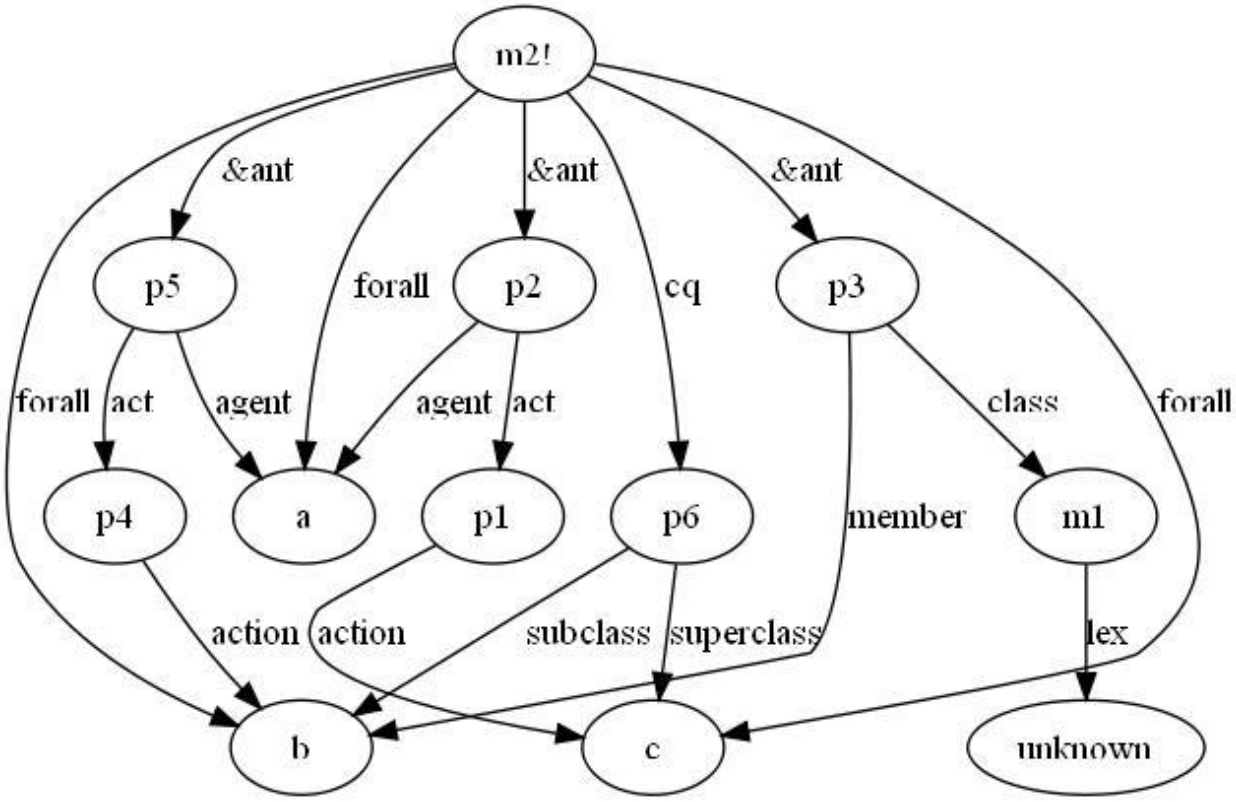
> "Katharine would calculate that she had never known her write for more than ten minutes at a time. Ideas came to her chiefly when she was in motion. She liked to **perambulate** the room with a duster in her hand, with which she stopped to polish the backs of already lustrous books, musing and romancing as she did so. Suddenly the right phrase or the penetrating point of view would suggest itself, and she would drop her duster and write ecstatically for a few breathless moments; and then the mood would pass away, and the duster would be sought for, and the old books polished again."

This passage obviously contains a plethora of information that, in the more superficial initial application I did not use. The goal of CVA as stated earlier is to develop algorithms that can be used to help teach humans, so I interviewed a few test subjects that did not know what the word perambulate meant to see what information within the passage they found necessary to surmising a reasonable definition for perambulate. While all of my informants had fairly different ways of going about discerning the meaning of perambulate, they all came up with the same general idea, and I decided that is what I would get Cassie to imitate. After digging through the minds of my informants, I had scraped out the background knowledge that they had needed in order to make an accurate assumption of the meaning of the unknown word. There were also many additional clues in the text that could also be used, and I will go into more detail on that later in this report.

The primary background knowledge was distilled down into two main applicable rules for Cassie to relate to. The first being that if the unknown word is a verb, which in this case you can reasonably say it is, and she is doing another verb, then you can assume that the two verbs are related and possible have a subclass-superclass relation. Now this is by no means an infallible truth, however it is how us humans would think, and we are not cold logical creatures. This rule states that if some object (variable a) does the act of variable b, and b is an unknown item, and if the same object (a) does another act (variable c). Then variable B is a subclass of variable C. This rule is given to Cassie in the following SNePSUL command.

```
(describe (assert forall ($a $b $c)
        &ant ((build agent *a act (build action *b))
                (build member *b class (build lex "unknown"))
                (build agent *a act (build action *c)))
        cq (build superclass *c subclass *b)))
```

Which results in a node structure that looks like this.



The second piece of background knowledge that my informants required is more of a decryption of a nuance of English that Cassie would obviously not understand intrinsically. So like all non native English speakers, I had to break it down into more hard and fast rules. In the

passage, it is said that "Suddenly the right phrase or penetrating point of view would suggest itself", which in order to relate to the earlier relation of "Ideas came to her chiefly while she was in motion" means that we need to say that the first phrase is a way of representing a person getting an idea. Essentially, the rule states that if the right phrase presents itself to some object (e), then variable e gets an idea. This is represented to Cassie in the following way.

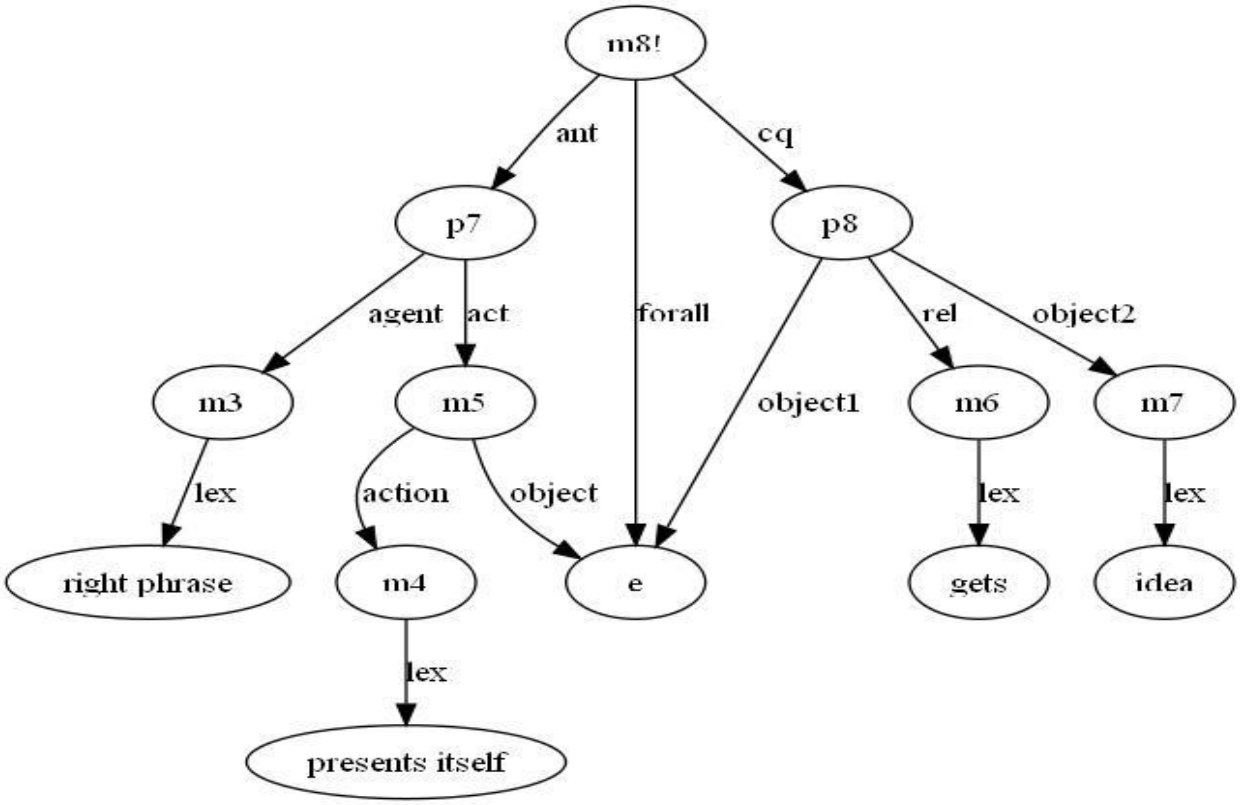
(describe (assert forall ($e)

ant (build agent (build lex "right phrase")

act (build action (build lex "presents itself")) object *e)

cq (build object1 *e rel (build lex "gets") object2 (build lex "idea"))))

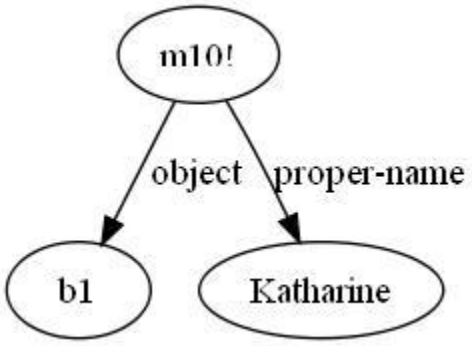
Which in Cassie's brain looks like this.



From here on out, the information represented added into Cassie is all from data gathered in the passage. First and foremost we need to tell Cassie who Katharine is, which we will then use as the object of most of the further deductions. This is just to say that there is a node in her mind that represents the object Katharine. This is done by the SNePSUL command:

(describe (add object #k proper-name (build lex "Katharine")))

Which allows me to reference Katharine by the *k base node. This appears in her had simply as:



Afterwards we read that ideas come to Katharine if she is moving. The significant rule we can draw from this is that if Katharine got an idea, the she is moving. In Cassie's terms the rule states that for a variable x, if x gets an idea, and x's proper name is Katharine, then x is moving. This is relayed to Cassie as follows.

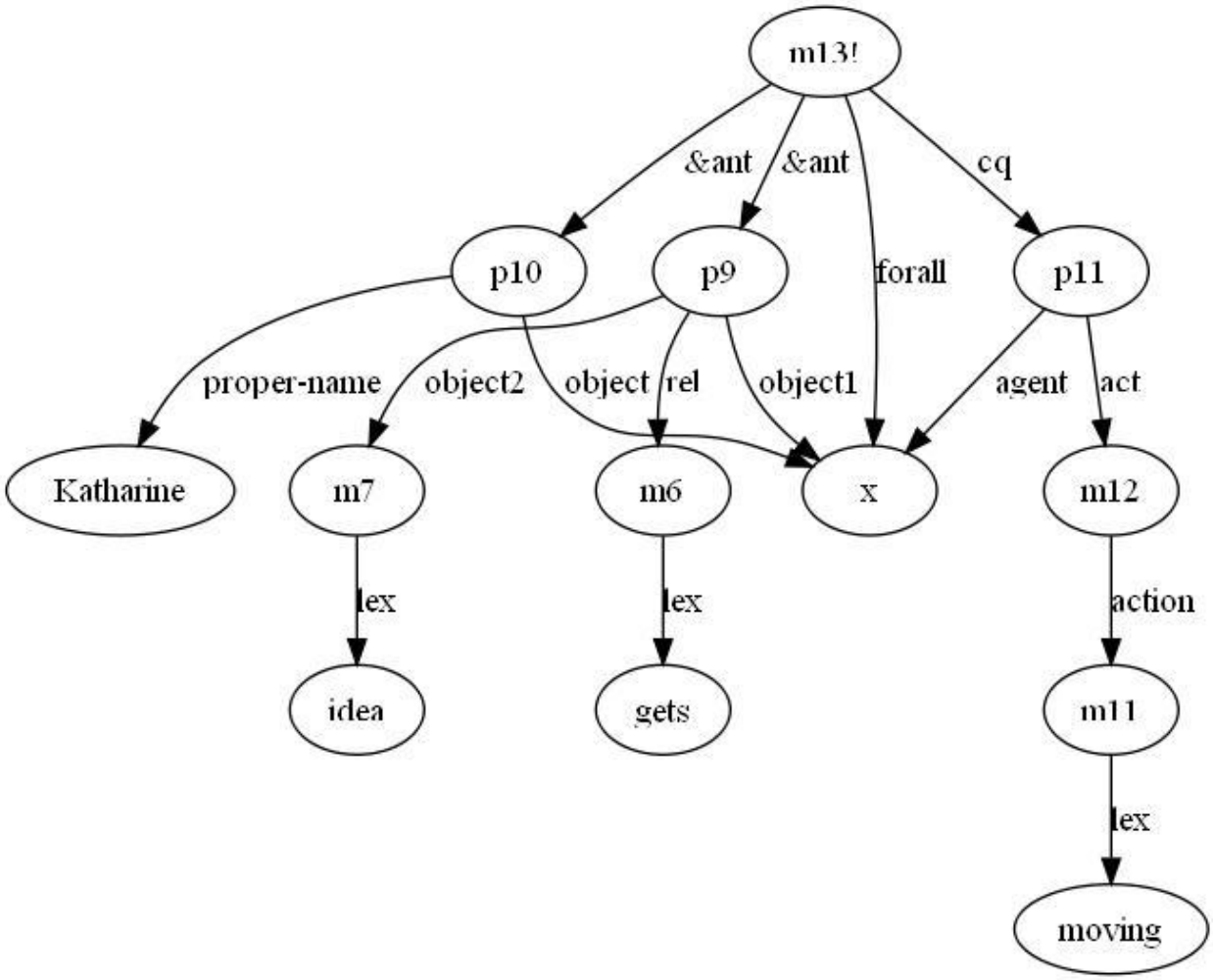
(describe (add forall $x

&ant ((build object1 *x rel (build lex "gets") object2 (build lex "idea"))

(build object *x proper-name (build lex "Katharine")))

cq (build agent *x act (build action (build lex "moving")))))

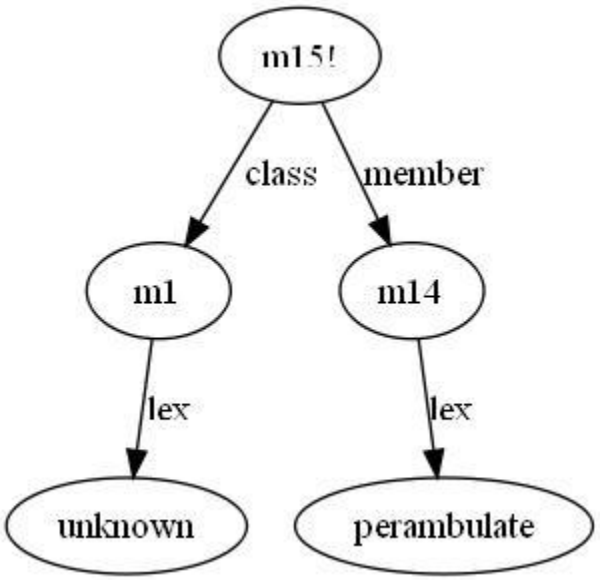
This appears like so in Cassie's mind:



After this, we read the word perambulate, which we want to designate as unknown. This is a more invasive way of stating that a word is unknown, if this were implemented in a much larger semantic network, Cassie would be able to read the word "perambulate" and then search her memory for the word, and if she does not have a definition of it, then determine that it is unknown and work from there. We simply state that the word perambulate belongs to the class of unknown things in this manner.

(describe (add member (build lex "perambulate") class (build lex "unknown")))

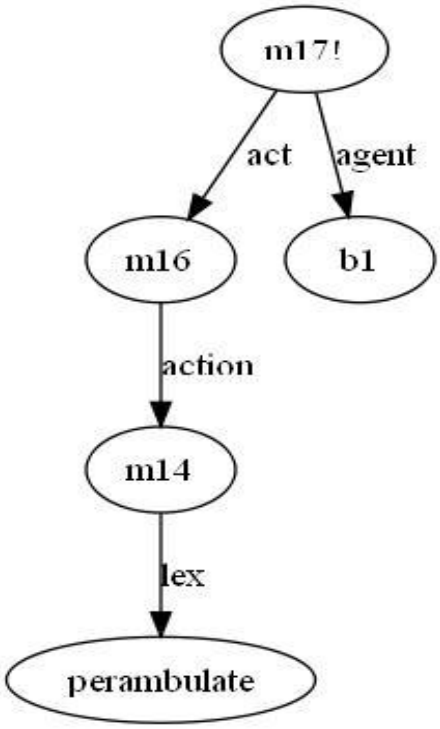
This proposition looks like this in Cassie's mind:



After this we are told that Katharine perambulates, so we must represent this to Cassie in a simple agent act action case-frame.

(describe (add agent *k act (build action (build lex "perambulate"))))

Which appears in Cassie's mind like this:

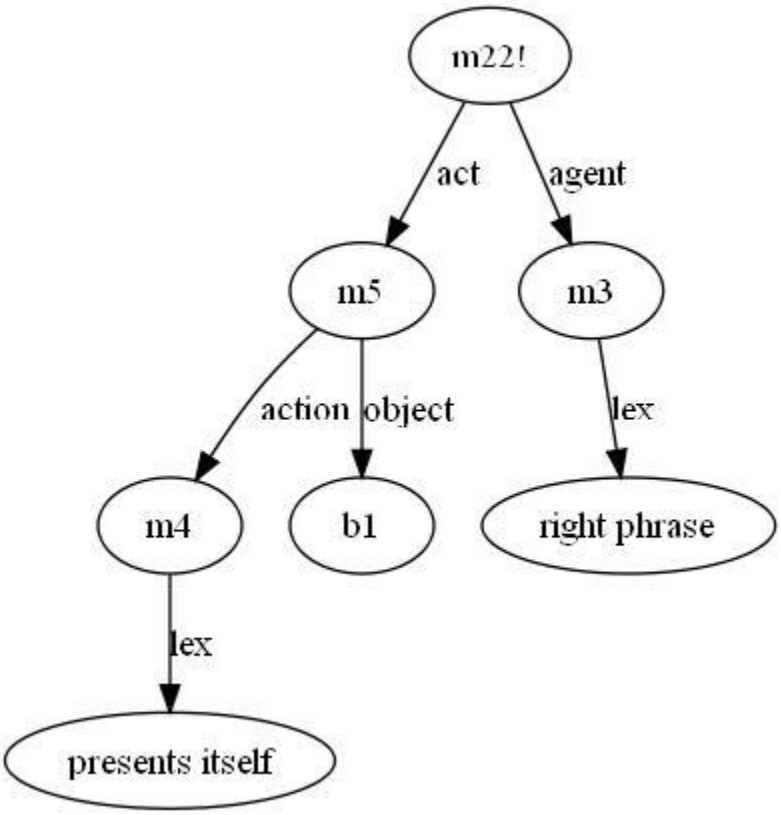


Lastly, there is the statement that the right phrase presents itself to Katharine, this is done via the SNePSUL command:

(describe (add agent (build lex "right phrase")

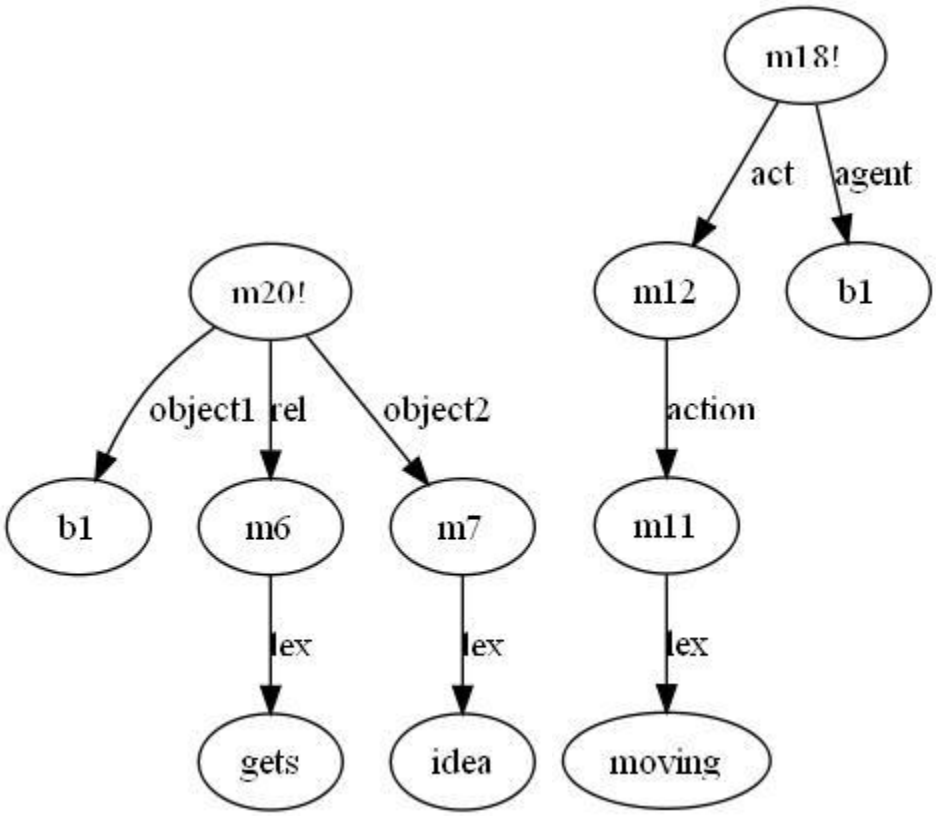
act (build action (build lex "presents itself")) object *k))

This appears like this in Cassie's mind:

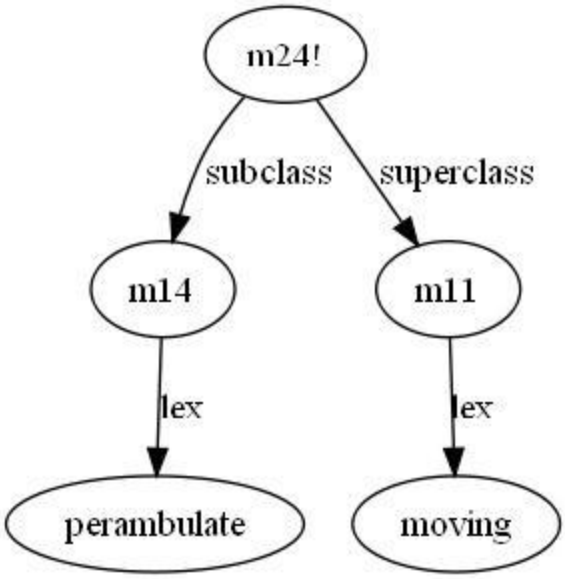


However, at the same time as this is put in, Cassie take this information and many of the previously instantiated rules fire. One realizing that because the right phrase presented itself, to some object, in this case Katharine, then Katharine gets an idea. After that, the rule fires that if Katharine gets an idea, then she is moving.

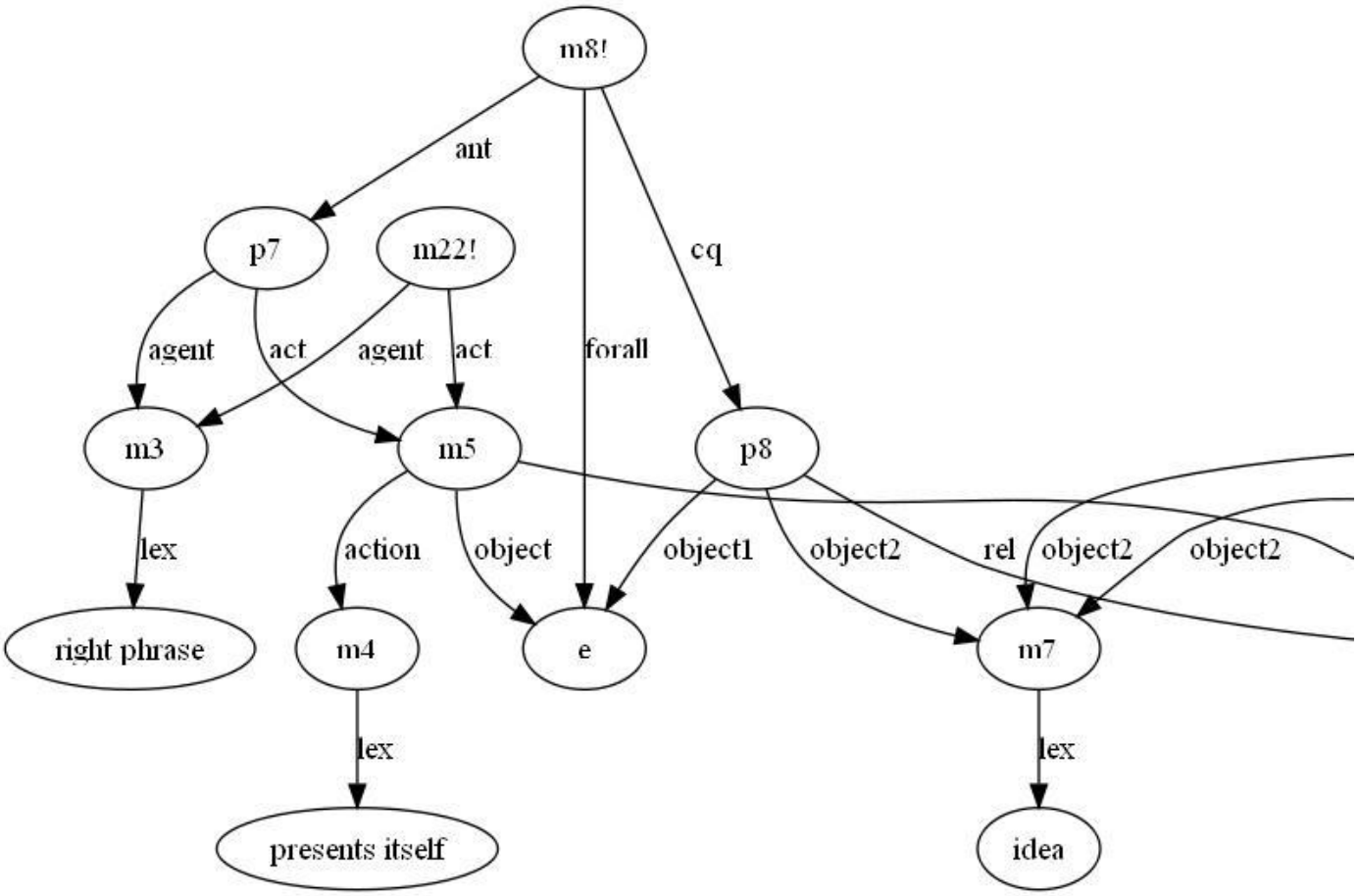
These two deductions appear in Cassie's head as:

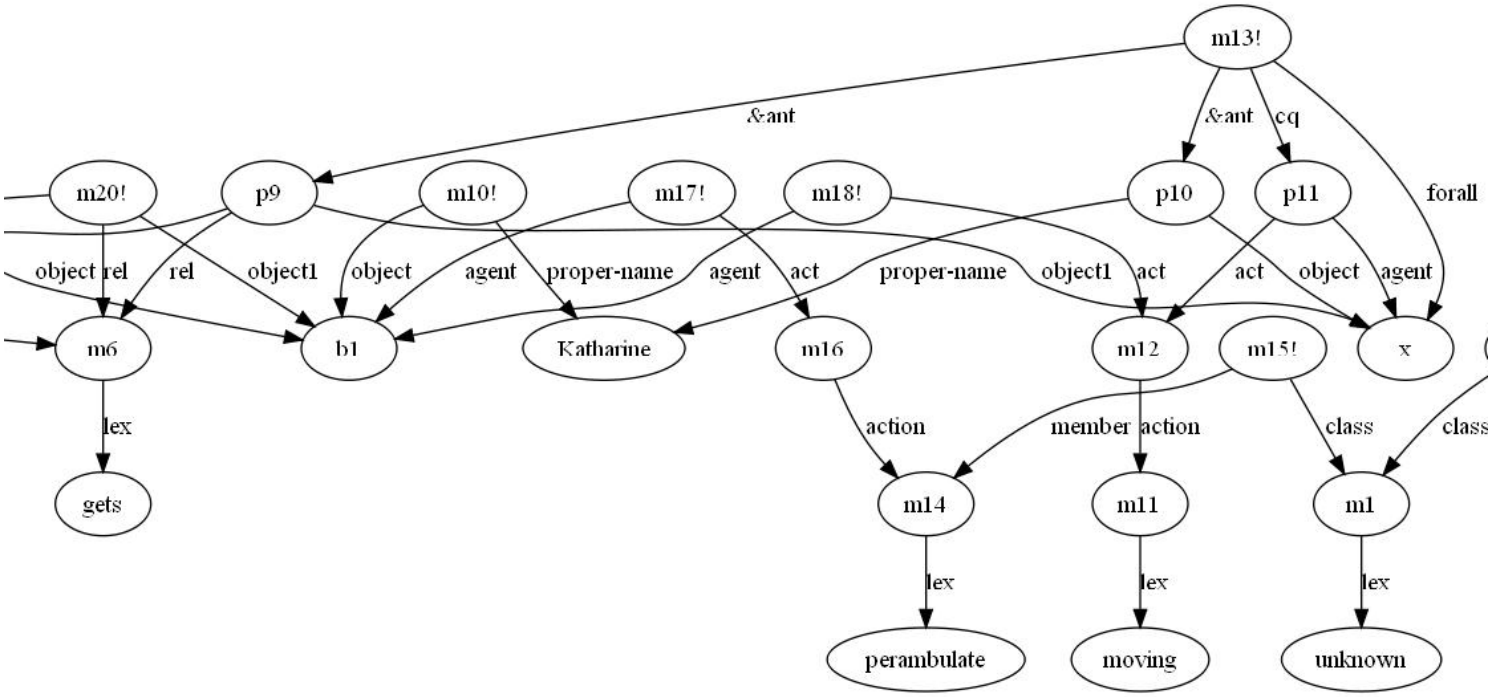


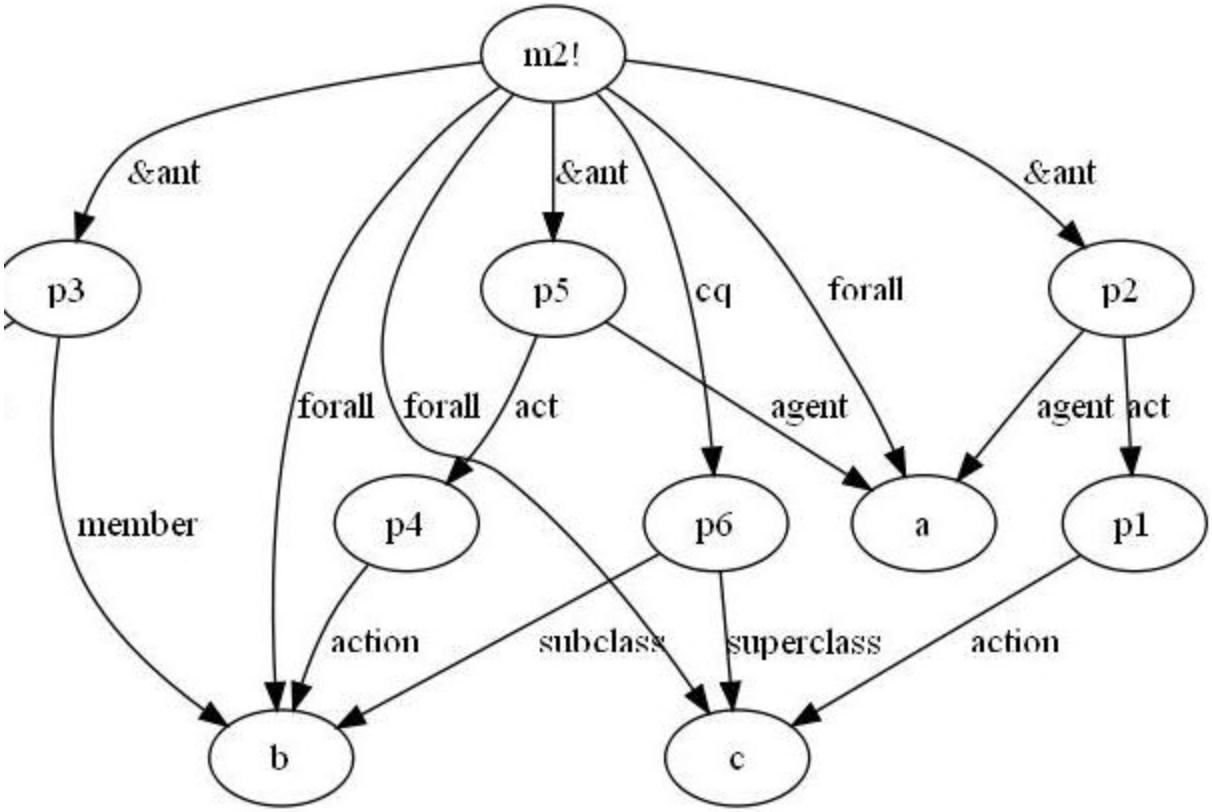
Given all of these new connections, the last rule fires, stating that because Katharine is perambulating, and perambulate is unknown, and Katharine is also moving, then perambulate is a subclass of moving.



When the entire network is drawn together, it looks like this:




m13!
&ant
&ant
cq
forall
m20!
p9
m10!
m17!
m18!
p10
p11
object
rel
rel
object1
object
agent
proper-name
agent
act
proper-name
object1
act
act
object
agent
m6
b1
Katharine
m16
m12
m15!
x
lex
action
member
action
class
class
gets
m14
m11
m1
lex
lex
lex
perambulate
moving
unknown


m2!
&ant
&ant
&ant
p3
p5
cq
forall
p2
forall
forall
act
agent
agent
act
member
p4
p6
a
p1
action
subclass
superclass
action
b
c

Following Steps

If I had a few more days to work on the project, I would get the algorithm to not recognize unknown as a super class of perambulate. This is possibly caused by the way I was representing how an idea presents itself to Katharine. Instead of using the act action object case frame, try to reference Katharine as the primary particle.

In a more long term scale, I would like to incorporate more information from the original context into Cassie's logical circuit. This would allow for more specific rules due, and in the grand scheme, allow all of the rules to be thrown into background knowledge of Cassie's brain and have them not conflict with any other potential rules.