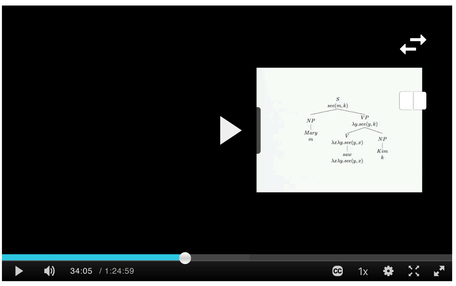
Lecture 18

Grammars and some comments provided below.



Class starts at around 3:20 in the video. At 1:05:30, I tried a query with "the son of the mother of the father of Maggie", and that doesn't work. The problem was I defined predicates for female and male, but I was moving quickly and didn't add all the necessary facts. I said Bart is male, but I never said Homer is male. Because son was defined as a male child, and Homer was not male, Homer couldn't be a son. In the version of the grammar pasted below, I added some comments and filled in the rest of the knowledge.

```
% First we define some knowledge about the world
% using facts and rules.
%
% This is not natural language, just some facts
% about the world we are modeling.

% We saw these in the very beginning of the quarter.

male(bart).
male(homer).
male(abe).

female(lisa).
female(maggie).
female(marge).
female(mona).

child(bart).
child(lisa).
child(maggie).

adult(homer).
adult(marge).
adult(mona).
adult(abe).

boy(X) :- male(X), child(X).
girl(X) :- female(X), child(X).

man(X) :- male(X), adult(X).
woman(X) :- female(X), adult(X).

mother(marge, bart).
mother(marge, lisa).
mother(marge, maggie).
mother(mona, homer).

father(homer, bart).
father(homer, lisa).
father(homer, maggie).
father(abe, homer).

parent(X, Y) :- mother(X, Y).
parent(X, Y) :- father(X, Y).

son(X, Y) :- male(X), parent(Y, X).
daughter(X, Y) :- female(X), parent(Y, X).
sibling(X, Y) :- parent(Z, X), parent(Z, Y), X \= Y.

% Now the language part. This could be written in a
% separate prolog file. There is no reason the world
% knowledge and the language knowledge need to be in
% the same file.

% Some lexical rules

% Proper names
% Notice that each proper name has a meaning,
% which happens to look like the name. These
% meanings have to match what we have in our facts
% about the world.
%
% One way to think about this is that in our
% knowledge above, we said that marge is the
% mother of bart. Now, we can say how the
% meaning "marge" is expressed in English.

propn(homer) --> [homer].
propn(bart) --> [bart].
propn(abe) --> [abe].
propn(mona) --> [mona].
propn(lisa) --> [lisa].
propn(marge) --> [marge].
propn(maggie) --> [maggie].

% nouns that refer to unary predicates.
% (here, we say how the predicate boy/1 is
% expressed in English.)

n(X, boy(X)) --> [boy].
n(X, girl(X)) --> [girl].
n(X, man(X)) --> [man].
n(A, woman(A)) --> [woman].

% nouns that refer to binary predicates

n(X, Y, parent(X, Y)) --> [parent].
n(X, Y, father(X, Y)) --> [father].
n(X, Y, mother(X, Y)) --> [mother].
n(X, Y, sibling(X, Y)) --> [sibling].
n(C, P, daughter(C, P)) --> [daughter].
n(C, P, son(C, P)) --> [son].

% preposition

p --> [of].

% Now some phrase structure rules.
% One could add vp and s rules to do
% things like Marge is the mother of Bart,
% which would be true. But in this
% very simple example, we just have
% simple noun phrase rules.

% NP

np(X) --> propn(X).

np(X) --> det, n(X, Pred),
 { call(Pred) }.

np(X) --> det, n(X, Y, Pred), pp_comp(Y),
 { call(Pred) }.

% PP

pp_comp(X) --> p, np(X).

% determiners. In this example we
% don't do anything interesting
% with these.

det --> [the].
det --> [a].
```

Then we can try some queries:

```
np(P, [marge], []).
P = marge
false.
```

If we press enter once, we get marge, and if we press enter again, we get False. False here doesn't mean there is no solution. Clearly there is a solution, as we see in P = marge. That means that the meaning of the noun phrase "marge" is *marge*. What false means here is that prolog could not determine with certainty that P = marge is the only solution, so it looked for more solutions, and didn't find any. False here means: I didn't find anything else.

```
np(P, [the, mother, of, bart], []).
P = marge
```

Now let's try sibling. And I'll press semicolon after each solution, to see if there are more.

```
np(P, [the, mother, of, the, sibling, of, bart], []).
P = marge
P = marge
P = marge
P = marge
false.
```

Why does Marge appear four times?

Consider the following:

```
np(P, [the, sibling, of, bart], []).
P = lisa
P = maggie
P = lisa
P = maggie
false.
```

We know that Bart has two siblings, but why does each sibling appear twice? Take a look at the definition of sibling, and the following should explain why we get each sibling twice:

```
np(P, [a, parent, of, bart], []).
P = marge
P = homer
false.
```

Of course, we can compose these in any way we want, which is the whole point:

```
np(P, [the, mother, of, the, father, of, a, sibling, of, maggie], []).
P = mona
```