L344 Computational Syntax and Semantics Coursework requirements

March 1, 2006

1. Begin with the following grammar:

```
:- op(1200, xfx, --->).
:- op(1200, xfx, ===>).
:- op( 30, fy, ~).
:- op(500, xfy, &).
:- op(500, xfy, #).
:- op(510, xfy, =>).
:- op(510, xfy, <=>).

s(Smx, Store0-Store) --->
   [np(NPmx, Store0-Store1), vp(VPmx, Store1-Store)],
   {beta_reduce(NPmx*VPmx, Smx)}.

vp(Vimx, Store) --->  [vi(Vimx, Store)].

vp(VPmx, Store0-Store) --->
   [vt(Vtmx, Store0-Store1), np(NPmx, Store1-Store)],
   {beta_reduce(Vtmx*NPmx, VPmx)}.

np(P^(P*Z), [stored(NPmx, Z) | Store0]-Store) --->
   [det(Detmx, Store0-Store1), nbar(Nbarmx, Store1-Store)],
   {beta_reduce(Detmx*Nbarmx, NPmx)}.

np(PNamemx, Store) ---> [pname(PNamemx, Store)].

nbar(Nmx, Store) ---> [n(Nmx, Store)].

det(Q^P^for_all(X, Q*X => P*X), S-S) ===> [every].

det(Q^P^exists(X,  Q*X & P*X), S-S) ===> [some].

n(X^soldier1(X), S-S)                    ===> [soldier].

n(X^witch1(X), S-S)                    ===> [witch].

pname(P^(P*d), S-S)                       ===> [duncan].

pname(P^(P*m), S-S)                       ===> [macbeth].

vi(X^died1(X), S-S)                      ===> [died].

vt(P^Y^(P*X^killed1(Y, X)), S-S)         ===> [killed].
```

2. Add to it (including both syntax and semantics):

   - A lexical entry for the word *Glamis*
   - A lexical entry for the word *gave*
   - A lexical entry for the word *given*
   - A lexical entry for the word *who*
   - A lexical entry for the word *castle*
   - VP rules to allow you to parse the following sentences

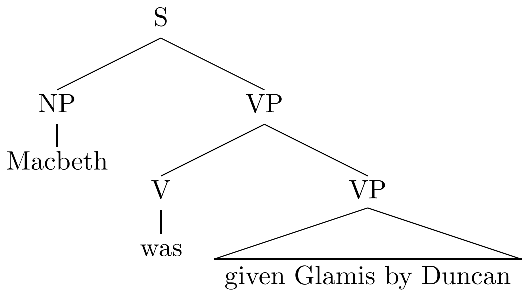
a. Duncan gave Glamis to Macbeth[1]
b. Duncan gave Macbeth Glamis.
c. Macbeth was given Glamis by Duncan
d. Glamis was given to Macbeth by Duncan

3. Add the fact `gave1(d, g, m).` (representing the translation of 'Duncan gave Glamis to Macbeth') to the database and demonstrate that the program provides correct responses to all the sentences in 2.2a-2d above.

4. Add the fact `castle1(g)` (representing that Glamis is a castle) to the database and confirm that the sentences 'Duncan gave every castle to Macbeth' and 'Duncan gave some castle to Macbeth' have the same truth conditions.

5. Modify the `answer/1` predicate so that it will give sensible answers to questions such as 'Who gave Glamis to Macbeth'.

Assume the following constituent structure analysis for passives:



Try to ensure that your grammar does not overgenerate by allowing parses of ungrammatical sentences. (Such as 'Duncan gave Glamis by Macbeth', or 'Macbeth was gave Glamis by Duncan'; use the techniques for handling number agreement as a basis for ensuring this.)

Your coursework submission should consist of

1. A program listing of all your additions and modifications, with comments explaining what the code is for.

2. Evidence that the program will do what it is supposed to do by way of successfully parsing and answering correctly a range of input sentences.

3. A discussion (of not more than 1000 words) clearly explaining why your program takes the form that it does, identifying any failings in the program (e.g. under- or over-generation) and commenting on any difficulties you have encountered.

The complete project report should be placed in my box opposite the photocopier by 12 noon on Friday, 28 April 2006.

[1]For those unfamiliar with the plot 'Glamis' is a castle.