Friday 09-03-18

Part(1): Before starting proofs

(A) Scoping:

```
X=[existential::{[tense(present)],A},existential::{(simple,A),B},name::{[Fido:NP],C},indefinite::{[dog>singular],D}],sortQStack(X,Y).
```

```
Y=[name::{[Fido:NP],C},existential::{(simple,A),B},existential::{[tense(present)],A},indefinite::{[dog>singular],D}]
```

Issue(1) Because I used qsort, quantifiers with same scope score will be resorted as well!. you have defined qsort/4 in useful, but I didn't know how to used it in this case to avoid this problem.
Issue(2) is *not* a modifier or a specifier? currently it a appears as modifier(identity, not), which means it is not in the quantifiers stack, cant do the scoping on it.

(B) set the scene

```
doItAll(TXT,XN):-
      parseOne(TXT, X0),
      convSteps(X0,X1),
      fixQuants(X1,X2),
      qff(X2,X3),
      pretty(X3),
      anchor(X3, X4).
      X4=..[SPEECHACT,XN],
     (SPEECHACT = claim ->
      addToMinutes(XN);
      SPEECHACT = query ->
      tryToAnswer(XN);
      format('Er ??? ~w~~n', [SPEECHACT])).
```

```
   addToMinutes(Tree):-
     pretty(Tree),
     format('~nAdd the tree to Minutes~n', []).
/** (retract(kb(minutes, MINUTES)) ->
     assert(kb(minutes, MINUTES & Tree));
     assert(kb(minutes, Tree))).**/

   startConversation :-
      retractall(kb(minutes, _)).

   tryToAnswer(Tree):-
      pretty(Tree),
      format('~nTry to answer the query~n', []).

   anchor(X, X).
```

Part(2) Doing proofs.

Straight Syllogism example	
‘Fido is an animal’	claim(({[tense(present)],#0} & ({(simple,#0),#1} & name(A::{[Fido:NP],A}, ({[animal>singular],#2} & [[be, {predication(xbar(v(-),n(+))),#2}, {subject,A}], #1]))))
‘all animals are mortals’	claim(({[tense(present)],#0} & ({(simple,#0),#1} & ({[animal>plural],A} => ([[be, {predication(xbar(v(-),n(+))),B}, {subject,A}], #1] => {mortal>plural,B}))))
‘is Fido a mortal?’	query(({[tense(present)],A} & ({(simple,A),B} & name(C::{[Fido:NP],C}, ({[mortal>singular],D} & [[be, {predication(xbar(v(-),n(+))),D}, {subject,C}], B]))))

Example that needs the involvement of natural logic matching algorithm	
‘Fido is a fat dog’	name(A::{[Fido:NP],A}, ({[dog>singular,modifier(amod,fat)],#0} & [be, {predication(xbar(v(-),n(+))),#0}, {subject,A}]))
‘all animals are mortals’	claim(({[tense(present)],#0}

	& ({(simple,#0),#1} & ({[animal>plural],A} => ([[be, {predication(xbar(v(-),n(+))),B}, {subject,A}], #1] => {mortal>plural,B}))))
'is Fido a mortal?'	query(({[tense(present)],A} & ({(simple,A),B} & name(C::{[Fido:NP],C}, ({[mortal>singular],D} & [[be, {predication(xbar(v(-),n(+))),D}, {subject,C}], B]))))

Part(3): Parser Issues

1.There are number of example used to have parse trees until last week's dictionaries update.

'John is eating peaches' X 'John is a fool' ✔

'John has loved Mary' X 'John has peaches' ✔

My observation is that when auxs are used as helping verbs they don't work.

2.Can you show me how to add open class words and verbs, because most of the time 'no such word X' is stopping me from trying FraCas examples that are similar to the ones we are working on.

3.'want to be', 'going to' and relative clauses are frequently used constructions in FraCas. We don't have parse trees for the first two and not sure about the accurateness of the last.

4.I would like to know what other quantifiers could do: both, neither, each, at least/most, few, a few, a lot, lots of, most. And and, or , nor.