Friday 13-04-18

Intro.

Last meeting we've edited nf.pl and satchmoplus.pl while explaining the importance of using labels, in addition to their commentary role while doing proofs, to show abducted things like names, to add certainty factors to facts, and so on. Because I run into running issues when I wanted to test basic examples to continue from where I stopped I saved these versions and went back to the un-edited version for now as I want to make sure we are perfectly done with the basics before starting doing interesting things-pretty soon I hope.

I added few words to englishopen dictionary

Part(1): clearConversation

```
startConversation :-
      format('~nstarting a new conversation ~n', []),
      retractall(kb(minutes, _)),
      retractall(fact(_)),
      retractall(_=>_),
      gensym(reset(#)).   <---- ?
```

Part(2):anchoring pronouns

```
| ?- doItAll('John loves peaches.',X).
| ?- doItAll('he also loves cats.',X).
| ?- doItAll('John likes cats?',X).
| ?- listing(fact).
fact({[tense(present)],'#0'}).
fact({(simple,'#0'),'#1'}).
fact({[man>singular],'#2'}).
fact({[woman>singular,modifier(amod,pretty)],'#3'}).
fact({['John':'NP'],'#8'}).
fact({[tense(present)],'#5'}).
```

```
fact({(simple,'#5'),'#6'}).
fact({he,'#12'}).
fact({[tense(present)],'#9'}).
fact({(simple,'#9'),'#10'}).
```

Anchoring pronouns the way it appears doesn't make sense to me. And I tried to think of 'he' as 'the male'- assuming there is only one male introduced in the conversation- so that I can link it to 'John'. But how would I know that 'John' is a male or even a human !! this may sound stupid ☺ but sorry I couldn't get it.

Part(3):modifiers

I wanted to try examples with modifiers so that the matching algorithm can skip over. The first one I've tried is 'fool'. It appeared as a modifier of type 'identity' in the parse tree. Modifiers of type identity are something we get rid of in **nfTree**.

I tried other modifiers such 'pretty' and 'fake' they appear as 'amod' modifiers in the tree; no problems.

So, I checked the dictionary for 'fool' and I found 2 entries for it; one is 'nroot' the other is 'vroot'.

Is that the reason why we get fool as identity modifier? is fixing the dictionary will fix the problem or do we need to revisit nfTree to change what we do with identity modifiers?

```
| ?- doItAll('John loves a fool dog.',X).
ts([.,
    arg(claim,
        *([time(tense(present),
                aspect(simple),
                aux(-),
                def(-),
                finite(tensed))]),
        [love,
         arg(dobj,
             *(indefinite),
             [woman>singular,
              modifier(identity, fool>singular),
              modifier(identity, a)]),
```

```
        arg(subject, *(indefinite), [man>singular, modifier(identity, a)])])])
```

Example that Works

```
| ?- set(showProof).
| ?- doItAll('a man loves a pretty woman.',X).
| ?- doItAll('a man likes a woman?',X).
```

Part(4):Query issues:

Issue(1):Query involving names, def. references or proRef

```
| ?- doItAll('John is a human?',X).
query(name(A::{[John:NP],A},
           ({[tense(present)],B}
             & ({(simple,B),C}
                 & ({[human>singular],D}
                     & ({(member,C),#5(D,C,B)}
                         => [[be,
                              {predication(xbar(v(-),n(+))),D},
                              {subject,A}],
                             #5(D, C, B)])))))
```

the theorem prover can't work on parts separated with commas, can't we treat names, def. references and pronouns as existential parts and &nd them to the rest of the form?

Issue(2):Query involving a rule within a rule

```
| ?- doItAll('every man loves a woman.',X).
({[man>singular],#7}
  => ({[tense(present)],#8}
       & ({(simple,#8),#9}
           & ({[woman>singular],#10}
               & ({(member,#9),A}
```

```
                    => [[love, {dobj,#10}, {subject,#7}], A])))))

| ?- listing(=>).
{[man>singular],A}=>{[tense(present)],'#0'(A)}&({(simple,'#0'(A)),'#1'(A)}&({[woman>singular],'#2'(A)}&(
{(member,'#1'(A)),B}=>[[love,{dobj,'#2'(A)},{subject,A}],B])).
```

Part(5):Polarity

-in the current version of matching algorithm it is expected that :
‘every man loves a woman.’ -> **‘every** human likes a woman.’ although (every) is a downward monotone on its 1st argument. (so this is wrong, we need to check polarity, **but when to do the marking?**)

-according to the standard tree polarity marking, negations like (not) reverse the polarity. Which means ‘not every man loves a woman.’ -> ‘not every human loves a woman.’ **should be true?!**