Friday 02-03-18

Part(1) finalizing the steps toward a QFF form.

QFF and other predicates were fixed to deal with time quantifiers properly. Example 1, 2, 3 and 4 are supposed to be examples with the right final form.

Issue(1) Are these forms correct?

Issue(2): in many examples, the quantifier introduced by the time is 'unidentified' and this is because we get def(Variable) instead of '-' or '+'.

Issue(3): in sentences with XCOMP argument the final forms seems to be right but in weird indentations—I tried to remove some of the spaces to make it look close to normal—and I don't know why?

Issue(4): XCOMP argument introduces the 'zero' subject, do we need to do anything about it?

Issue(5): I tried a negation example 'John does not love Mary', it shoed 2 problems: first 'not' is removed by nfTree as we agreed on removing any modifier of type identity. The second is that the auxiliary do doesn't seem to have a right interpretation.

Example(1) existential time and generic quantifier

```
parseOne('a man will have been eating peaches.', X), convSteps(X, Y),fixQuants(Y,Z), qff(Z, QFF), pretty(QFF).
```

Normal Form:

```
claim(exists(A::{[tense(future),tense(past)],A},
             exists(B::{(prog,A),B},
                    exists(C::{[man>singular],C},
                           generic(D::{peach>plural,D},
                                   [[eat, {dobj,D}, {subject,C}],
                                    B])))))
```

QFF:

```
claim(({[tense(future),tense(past)],#0}
        & ({(prog,#0),#1}
            & ({[man>singular],#2}
                & ([[eat, {dobj,A}, {subject,#2}], #1]
                    => {peach>plural,A})))))
```

Example(2) referential time and names

```
parseOne('John has loved Mary.', X), convSteps(X, Y),fixQuants(Y,Z), qff(Z, QFF), pretty(QFF).
```

Normal Form:

```
claim(the(A::{[tense(past)],A},
          the(B::{(perfect,A),B},
              name(C::{[John:NP],C},
                   name(D::{[Mary:NP],D},
                        [[love, {dobj,D}, {subject,C}], B])))))
```

QFF:

```
claim(name(A::{[John:NP],A},
           name(B::{[Mary:NP],B},
                [[love, {dobj,B}, {subject,A}],
                 ref(C,
                     {perfect,ref(D,{[tense(past)],D}),C})])))
```

Example(3) referential 'the' and modifiers + Issue: def(Variable)

parseOne('John ate the three ripe peaches.', X), convSteps(X, Y),fixQuants(Y,Z), qff(Z, QFF), pretty(QFF).

nfTree:

```
[.,
 arg(claim,
     *([time(tense(past), aspect(simple), aux(-), def(A), finite(tensed))]),
     [ate,
      arg(dobj,
          *(the),
          [peach>plural, modifier(amod, ripe), modifier(numAsMod, three)]),
      arg(subject, *(name), [John:NP])])]
```

Normal Form:

```
claim(unidentified(A::{[tense(past)],A},
                   unidentified(B::{(simple,A),B},
                                name(C::{[John:NP],C},
                                     the((D
                                          :: {([peach>plural,
                                                modifier(amod, ripe),
                                                modifier(numAsMod, three)],
                                               D)}),
                                         [[ate,
                                           {dobj,D},
                                           {subject,C}],
                                          B])))))
```

QFF:

```
claim(unidentified(A::{[tense(past)],A},
                   unidentified(B::{(simple,A),B},
                                name(C::{[John:NP],C},
                                     [[ate,
                                       {(dobj,
                                          ref(D,
                                              {([peach>plural,
                                                  modifier(amod, ripe),
                                                  modifier(numAsMod, three)],
                                                 D)}))},
                                       {subject,C}],
                                      B]))))
```

Example(4) specifier ‘no’

parseOne('no man is an island.', X), convSteps(X, Y),fixQuants(Y,Z), qff(Z, QFF), pretty(QFF).

Normal Form:

```
claim(unidentified(A::{[tense(present)],A},
                   unidentified(B::{(simple,A),B},
                                no(C::{[man>singular],C},
                                   exists(D::{[island>singular],D},
                                          [[be,
                                            {predication(xbar(v(-),n(+))),D},
                                            {subject,C}],
                                           B])))))
```

QFF:

```
claim(unidentified(A::{[tense(present)],A},
                   unidentified(B::{(simple,A),B},
                                ({[man>singular],C}
                                  => not(({[island>singular],#0(C)}
                                          & [[be,
                                              {(predication(xbar(v(-), n(+))),
                                                 #0(C))},
                                              {subject,C}],
                                             B])))))))
```

Example(5) Issue: trees with XCOMP argument; NF and QFF appear in weird indentations

parseOne('John loves eating peaches.', X), convSteps(X, Y),fixQuants(Y,Z), qff(Z, QFF), pretty(QFF).

Normal Form:

```
claim(unidentified(A::{[tense(present)],A},
                   unidentified(B::{(simple,A),B},
                                name(C::{[John:NP],C},
                                     [[love,
                                       xcomp(unidentified((D:: {[tense(present)],D}),
                                                          unidentified((E:: {(prog,D, E)}),
                                                                       exists((F:: {zero,F}),
                                                                              generic((G:: {((peach> plural), G)}),
                                                                                  [[eat, {(dobj, G)},{(subject, F)}],
                                                                                       E])))))),
                                       {subject,C}],
                                      B])))))
```

QFF:

```
claim(unidentified(A::{[tense(present)],A},
                   unidentified(B::{(simple,A),B},
                                name(C::{[John:NP],C},
                                     [[love,
                                       xcomp(unidentified((D:: {[tense(present)],D}),
                                                          unidentified((E:: {(prog,D, E)}),
                                                                       ({zero,#0}
                                                                         & ([[eat, {dobj,F},{subject,#0}],E]
                                                                             => {((peach> plural),F)}))))),
                                       {subject,C}],
                                      B])))))
```

Example(6) Issue: negation

parseOne('John does not love Mary.', X), pretty(X),convSteps(X, Y),fixQuants(Y,NF),qff(NF,QFF),pretty(QFF).

Base Tree

```
[.,
 arg(claim,
     *(time(tense(present), aspect(simple), aux(+), def(A), finite(tensed))),
     [do>s,
      arg(B,
          *(time(tense(C), aspect(D), aux(-), def(E), finite(infinitive))),
          [[[love>, modifier(identity, not)],
```

```
            arg(dobj, *(name), [Mary:NP])],
          arg(subject, *(name), [John:NP])])])]
```

nfTree

```
[.,
 arg(claim,
     *([time(tense(present), aspect(simple), aux(+), def(A), finite(tensed)),
        time(tense(B), aspect(C), aux(-), def(D), finite(infinitive))]),
     [love,
      arg(dobj, *(name), [Mary:NP]),
      arg(subject, *(name), [John:NP])])]
```

Normal Form

```
claim(unidentified(A::{tense(B),A},
                   unidentified(C::{(D,A),C},
                                name(E::{[John:NP],E},
                                     name(F::{[Mary:NP],F},
                                          [[love,
                                            {dobj,F},
                                            {subject,E}],
                                           C])))))
```

QFF

```
claim(unidentified(A::{tense(B),A},
                   unidentified(C::{(D,A),C},
                                name(E::{[John:NP],E},
                                     name(F::{[Mary:NP],F},
                                          [[love,
                                            {dobj,F},
                                            {subject,E}],
                                           C])))))
```

Part(2) Doing proofs.

Straight Syllogism example	
‘Fido is an animal’	claim(unidentified(A::{[tense(present)],A}, unidentified(B::{(simple,A),B}, name(C::{[Fido:NP],C}, ({[animal>singular],#0} & [[be, {predication(xbar(v(-),n(+))),#0}, {subject,C}], B]))))
‘all animals are mortals’	claim(unidentified(A::{[tense(present)],A}, unidentified(B::{(simple,A),B}, ({[animal>plural],C} => ([[be, {predication(xbar(v(-),n(+))),D}, {subject,C}], B] => {mortal>plural,D}))))
‘is Fido a mortal?’	query(unidentified(A::{[tense(present)],A}, unidentified(B::{(simple,A),B}, name(C::{[Fido:NP],C}, ({[mortal>singular],D} & [[be, {predication(xbar(v(-),n(+))),D}, {subject,C}], B])))) **Issu(1): in our previous discussions we haven’t dealt with queries or tried query examples. I assumed that they are going to be treated the same way ‘claim’ is treated. Above is what I get as a QFF for a query example. Does it look right?**

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