Friday 23-02-18

Issue(1): We've agreed once on removing any 'present' tense item from the time list unless it is the only one like in the example 'John loves Mary'. However, I started to think that doing it after or with the ***extractQQ*** part might be a better plan. 'a man is eating every peach' is the example that made me think about this issue as the time list consist of 'present' tenses only!
Another reason is that in ***extractQQ,*** dealing with the time sequence list goes through steps: (1) **tenseList:** extract tenses list; Path, (2) **timeToQuantifie:** determines the time quantifier from the def(Sign) of the first entry and (3) **getAspect:** determines the aspect that relates the time to the event from the last entry in the list. With that being said,I was afraid that we might be getting rid of important information by refining the time list before doing ***extractQQ.***
Below examples shows what we get in case we reduced the time list before extractQQ and what we get in case we didn't reduce it at all: Example: **parseOne('a man will have been eating every peach.', X), convSteps(X, Y),extractQQ(Y,T,St).**

Non reduced time list	reduced time list
Tree after convSteps: [., arg(claim, *([time(tense(future), aspect(A), aux(+), def(-), finite(tensed)), time(tense(present), aspect(B), aux(+), def(+), finite(infinitive)), time(tense(past), aspect(perfect), aux(+), def(C), finite(participle)), time(tense(present), aspect(prog), aux(-), def(D), finite(participle))]), [eat, arg(dobj, *(universal), [peach>singular]), arg(subject, *(indefinite), [man>singular])])]	**Tree after convSteps:** [., arg(claim, *([time(tense(future), aspect(A), aux(+), def(-), finite(tensed)), time(tense(past), aspect(perfect), aux(+), def(B), finite(participle))]), [eat, arg(dobj, *(universal), [peach>singular]), arg(subject, *(indefinite), [man>singular])])]

qlf form:

```
claim(opaque(qq([time(tense(future),
                     aspect(A),
                     aux(+),
                     def(-),
                     finite(tensed)),
                 time(tense(present),
                     aspect(B),
                     aux(+),
                     def(+),
                     finite(infinitive)),
                 time(tense(past),
                     aspect(perfect),
                     aux(+),
                     def(C),
                     finite(participle)),
                 time(tense(present),
                     aspect(prog),
                     aux(-),
                     def(D),
                     finite(participle))],
                [eat,
                 qq(universal::{[peach>singular],E},
                    {dobj,E}),
                 qq(indefinite::{[man>singular],F},
                    {subject,F})])))
```

After extractQQ

Stack:

```
[(existential:: {[tense(future),tense(present),tense(past),tense(present)],A}),
  existential::{prog,A,B},
  indefinite::{[man>singular],C},
  universal::{[peach>singular],D}]
```

Term:

```
[[eat, {dobj,D}, {subject,C}], B]
```

After applyQuant:

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

qlf form:

```
claim(opaque(qq([time(tense(future),
                     aspect(A),
                     aux(+),
                     def(-),
                     finite(tensed)),
                 time(tense(past),
                     aspect(perfect),
                     aux(+),
                     def(B),
                     finite(participle))],
                [eat,
                 qq(universal::{[peach>singular],C},
                    {dobj,C}),
                 qq(indefinite::{[man>singular],D},
                    {subject,D})])))
```

After extractQQ

Stack:

```
[existential::{[tense(future),tense(past)],A},
  existential::{perfect,A,B},
  indefinite::{[man>singular],C},
  universal::{[peach>singular],D}]
```

Term:

```
[[eat, {dobj,D}, {subject,C}], B]
```

After applyQuant:

```
existential(A::{[tense(future),tense(past)],A},
            existential(A,B::{perfect,A,B},
                        exists(C::{[man>singular],C},
                               forall(D::{[peach>singular],D},
                                      [[eat, {dobj,D}, {subject,C}],
                                                   B])))))
```

Issue(2):are you happy with the normal form we get from applying quantifiers? what can we change? what can we do more?

Issue(3): ‘Generics’ discussed treatments to be applied at this stage:

generics

```
  peaches = qq((universal :: {dobj,E}), {[peach>singular],E})
```

generics' ← recommended one

```
  peaches = qq((generic :: {[peach>singular],E}), {dobj,E}),
  {dobj, E, F} => peach(F)
```