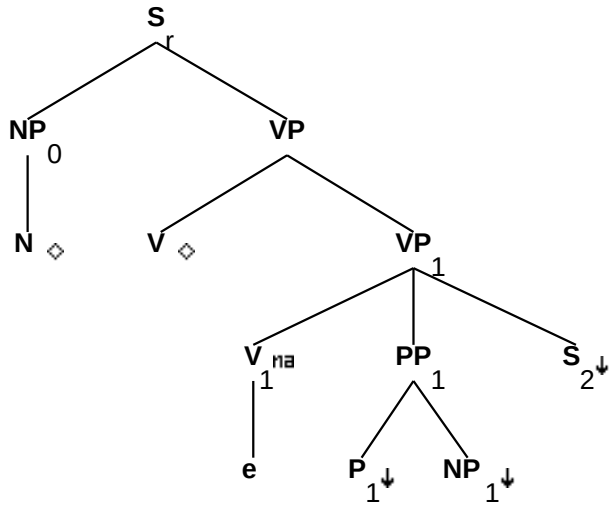
Family "TItVpnx1s2"

March 5, 2008

1 Tree "alphaItVpnx1s2"

1.1 graphe



1.2 comments

```
It-cleft with PP as clefted element
simple declarative

e.g.

     It was in the drawing room that the butler did it.
```

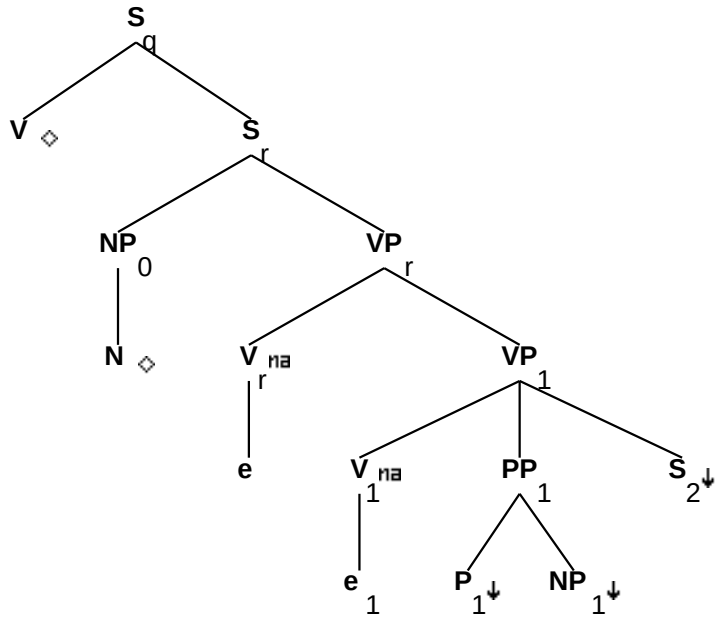
1.3 features

```
S_r.t:<assign-comp> = inf_nil/ind_nil
S_r.b:<assign-comp> = VP.t:<assign-comp>

S_r.b:<assign-case> = VP.t:<assign-case>
NP_0.t:<case> = S_r.b:<assign-case>
N.t:<case> = NP_0.b:<case>
```

S_r.b:<mode> = VP.t:<mode>
S_r.b:<comp> = nil
S_r.b:<tense> = VP.t:<tense>
NP_0.t:<agr> = S_r.b:<agr>
NP_0.t:<wh> = -
S_r.b:<agr> = VP.t:<agr>
S_r.b:<conditional> = VP.t:<conditional>
S_r.b:<passive> = VP.t:<passive>
S_r.b:<perfect> = VP.t:<perfect>
S_r.b:<progressive> = VP.t:<progressive>
VP.b:<passive> = -
VP.b:<agr> = V.t:<agr>
VP.b:<mode> = V.t:<mode>
VP.b:<tense> = V.t:<tense>
VP.b:<assign-case> = V.t:<assign-case>
VP.b:<compar> = -
NP_0.b:<agr> = N:<agr>
NP_0.b:<wh> = N:<wh>
S_2:<extracted> = -
S_2:<mode> = ind
S_2:<assign-comp> = ind_nil
S_2:<comp> = that/nil
P_1.t:<assign-case> = PP_1.b:<assign-case>
NP_1:<case> = PP_1.b:<assign-case>
PP_1.b:<wh> = NP_1:<wh>
V.b:<mode> = V_1.b:<mode>
VP.b:<mode> = VP_1.t:<mode>
VP_1.b:<mode> = V_1.t:<mode>

2 Tree "alphaInvItVpnx1s2"

2.1 graphe



2.2 comments

```
It-cleft with PP as clefted element
Inverted structure for Y/N questions

e.g.

    Was it in the drawing room that the butler did it?
```

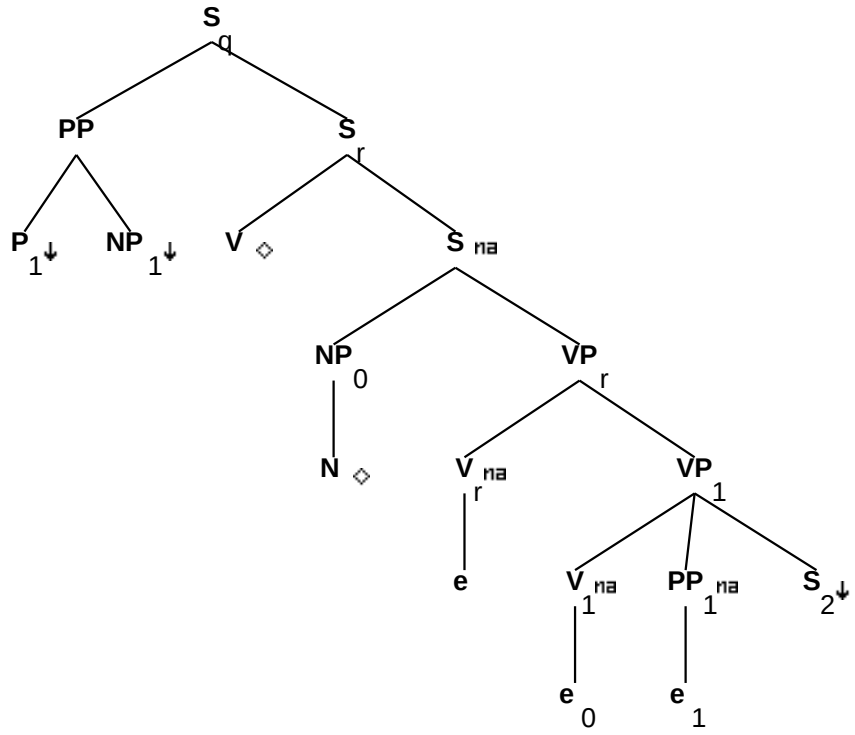
2.3 features

```
S_q.b:<inv> = +
NP_0.b:<agr> = N:<agr>
NP_0.b:<wh> = N:<wh>
NP_0.t:<agr> = S_r.b:<agr>
NP_0.t:<wh> = -
S_2.t:<assign-comp> = ind_nil
S_2:<comp> = that/nil
S_2:<extracted> = -
S_2:<mode> = ind
S_q.b:<agr> = S_r.t:<agr>
S_q.b:<assign-case> = V.t:<assign-case>
S_q.b:<comp> = nil
S_q.b:<conditional> = V.t:<conditional>
S_q.b:<mode> = V.t:<mode>
S_q.b:<passive> = -
```

```
S_q.b:<passive> = V.t:<passive>
S_q.b:<perfect> = V.t:<perfect>
S_q.b:<progressive> = -
S_q.b:<progressive> = V.t:<progressive>

S_r.b:<assign-case> = NP_0:<case>
S_r.t:<assign-comp> = S_q.b:<assign-comp>
V.t:<assign-comp> = S_q.b:<assign-comp>
S_r.b:<comp> = nil
S_r.b:<tense> = V.t:<tense>
S_r.t:<assign-case> = S_q.b:<assign-case>
S_r.t:<assign-comp> = inf_nil/ind_nil
V.t:<agr> = S_q.b:<agr>



VP_1.b:<compar> = -
VP_r.b:<compar> = -

S_r.t:<conj> = nil
P_1.t:<assign-case> = PP_1.b:<assign-case>
NP_1:<case> = PP_1.b:<assign-case>
PP_1.b:<wh> = NP_1:<wh>
V.b:<mode> = V_r.b:<mode>
V_r.b:<mode> = V_1.b:<mode>
VP_r.b:<mode> = V_r.t:<mode>
VP_r.b:<mode> = VP_1.t:<mode>
VP_1.b:<mode> = V_1.t:<mode>
```

3 Tree "alphapW1InvItVpnx1s2"

3.1 graphe



3.2 comments

```
It-cleft with PP as clefted element
wh-extraction on clefted PP
anchor 'be' inverted
no auxiliaries
```

e.g.

Where was it that the butler did it?

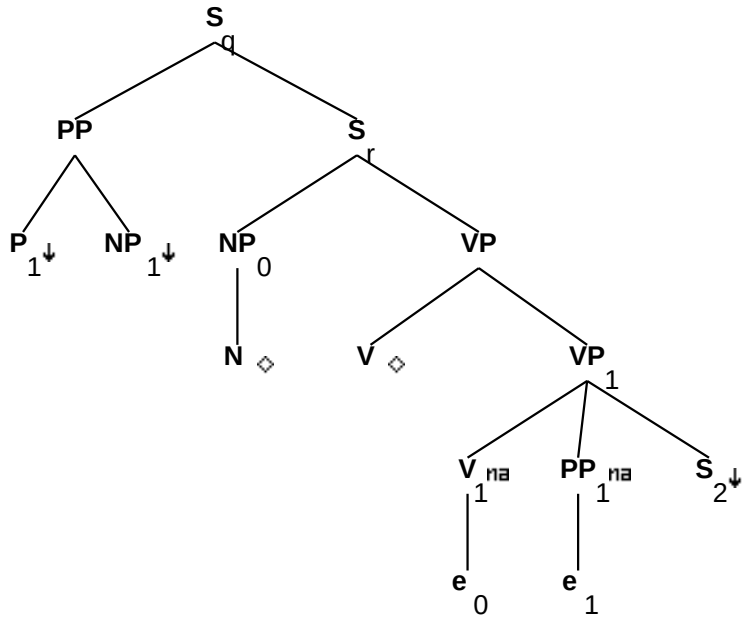
3.3 features

```
PP:<trace> = PP_1.b:<trace>
PP:<wh> = +
NP_0:<agr> = S.b:<agr>
NP_0.b:<case> = N.t:<case>
NP_0.b:<agr> = N.t:<agr>
S.t:<agr> = S_r.b:<agr>
NP_0:<case> = S.b:<assign-case>
S.t:<assign-case> = S_r.b:<assign-case>
```

```
S_r.b:<assign-case> = V.t:<assign-case>
S_2.t:<assign-comp> = ind_nil
S_2.t:<comp> = that/nil
S_2:<extracted> = -
S_2:<inv> = -

S_2:<mode> = ind
S_q.b:<assign-comp> = S_r.t:<assign-comp>
S_r.b:<assign-comp> = V.t:<assign-comp>
S_q.b:<comp> = nil
S_q.b:<conditional> = S_r.t:<conditional>
S_r.b:<conditional> = V.t:<conditional>
S_q.b:<inv> = +
S_q.b:<mode> = S_r.t:<mode>
S_q.b:<passive> = -
S_q.b:<passive> = S_r.t:<passive>
S_r.b:<passive> = V.t:<passive>
S_q.b:<perfect> = S_r.t:<perfect>
S_r.b:<perfect> = V.t:<perfect>
S_q.b:<progressive> = -
S_q.b:<progressive> = S_r.t:<progressive>
S_r.b:<progressive> = V.t:<progressive>
S_q.b:<wh> = PP:<wh>
S_q.t:<assign-comp> = inf_nil/ind_nil
S_r.b:<agr> = V.t:<agr>
S_r.b:<comp> = nil
S_r.b:<inv> = -
S_r.b:<mode> = V.t:<mode>
S_r.b:<tense> = V.t:<tense>



VP_1.b:<compar> = -
VP_r.b:<compar> = -

S_r.t:<conj> = nil
P_1.t:<assign-case> = PP.b:<assign-case>
NP_1:<case> = PP.b:<assign-case>
PP.b:<wh> = NP_1:<wh>
V.b:<mode> = V_r.b:<mode>
V_r.b:<mode> = V_1.b:<mode>
VP_r.b:<mode> = V_r.t:<mode>
VP_r.b:<mode> = VP_1.t:<mode>
VP_1.b:<mode> = V_1.t:<mode>
```

4 Tree "alphapW1ItVpnx1s2"

4.1 graphe



4.2 comments

```
It-cleft with PP as the clefted element
wh-extraction on the clefted PP
anchor 'be' not inverted
obligatory adjunction of at least on auxiliary


e.g.

     Where might it have been that the butler did it?
```

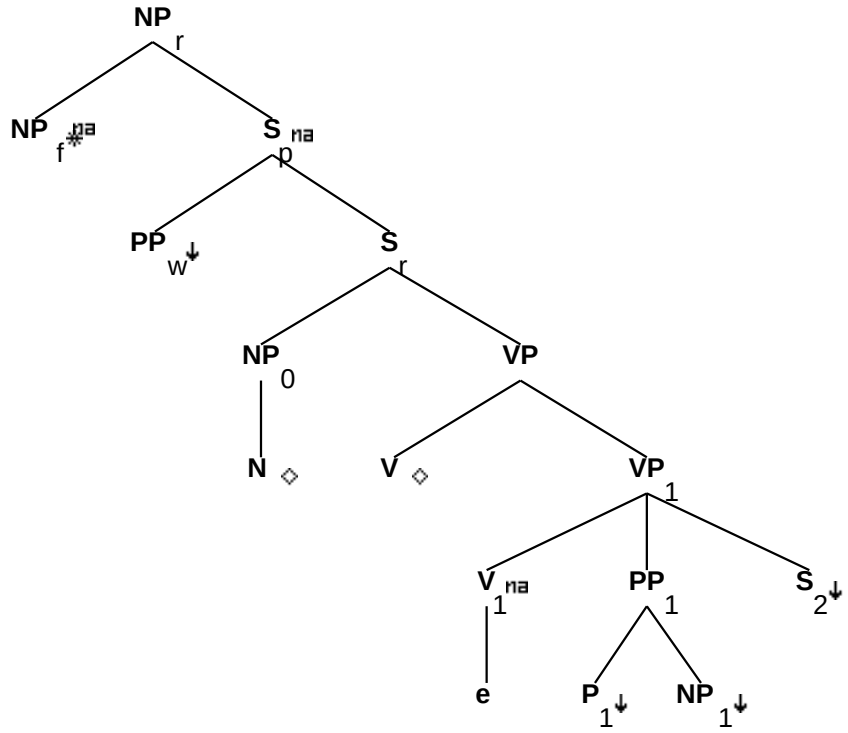
4.3 features

```
S_r.b:<mode> = inf/ger/ppart/base
PP:<trace> = PP_1.b:<trace>
PP:<wh> = +
NP_0.b:<agr> = N:<agr>
NP_0.b:<wh> = N:<wh>
NP_0.t:<agr> = S_r.b:<agr>
NP_0.t:<case> = S_r.b:<assign-case>
NP_0.t:<wh> = -
N.t:<case> = NP_0.b:<case>
S_2:<assign-comp> = ind_nil
S_2:<comp> = that/nil
S_2:<extracted> = -
```

```
S_2:<mode> = ind
S_r.b:<agr> = VP.t:<agr>
S_r.b:<assign-case> = VP.t:<assign-case>
S_r.b:<assign-comp> = VP.t:<assign-comp>
S_r.b:<comp> = nil
S_r.b:<conditional> = VP.t:<conditional>

S_r.b:<mode> = VP.t:<mode>
S_r.b:<passive> = VP.t:<passive>
S_r.b:<perfect> = VP.t:<perfect>
S_r.b:<progressive> = VP.t:<progressive>
S_r.b:<tense> = VP.t:<tense>
S_r.t:<assign-comp> = inf_nil/ind_nil

S_q.b:<comp> = nil
S_q.b:<conditional> = S_r.t:<conditional>
S_q.b:<inv> = +
S_q.b:<mode> = S_r.t:<mode>
S_q.b:<passive> = -
S_q.b:<passive> = S_r.t:<passive>
S_q.b:<perfect> = S_r.t:<perfect>
S_q.b:<progressive> = -
S_q.b:<progressive> = S_r.t:<progressive>
S_q.b:<wh> = PP:<wh>
S_q.t:<assign-comp> = inf_nil/ind_nil
VP.b:<agr> = V.t:<agr>
VP.b:<assign-case> = V.t:<assign-case>

VP.b:<mode> = V.t:<mode>
VP.b:<passive> = -
VP.b:<tense> = V.t:<tense>

VP_1.b:<compar> = -
VP.b:<compar> = -
S_r.t:<conj> = nil
P_1.t:<assign-case> = PP.b:<assign-case>
NP_1:<case> = PP.b:<assign-case>
PP.b:<wh> = NP_1:<wh>
V.b:<mode> = V_1.b:<mode>
VP.b:<mode> = VP_1.t:<mode>
VP_1.b:<mode> = V_1.t:<mode>
```

5 Tree "betaNpxItVpnx1s2"

5.1 graphe



5.2 comments

```
It-cleft with PP as clefted element
simple declarative

e.g.

     It was in the drawing room that the butler did it.
```

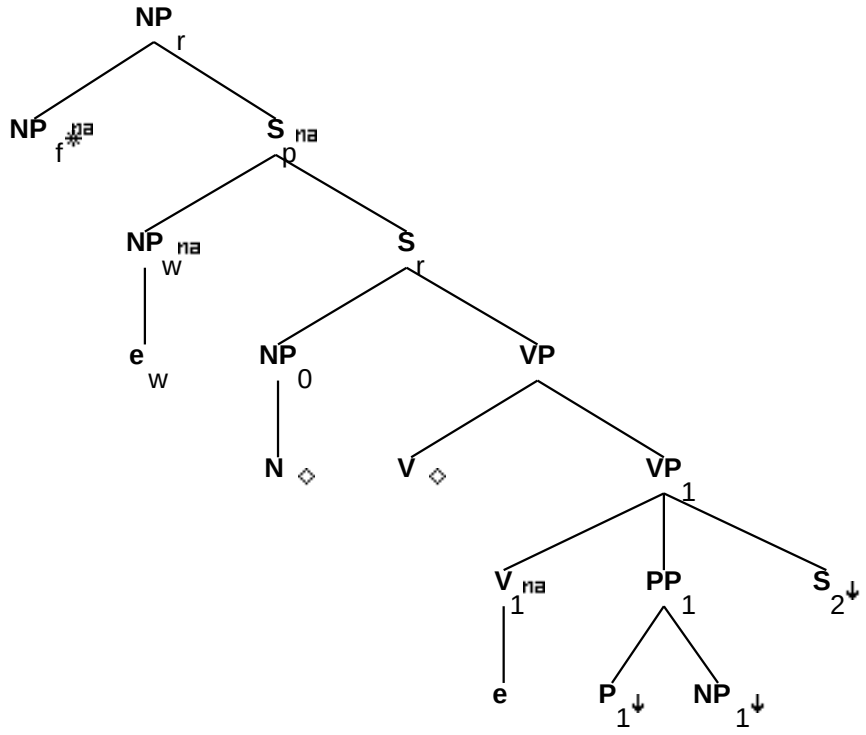
5.3 features

```
S_r.b:<assign-comp> = VP.t:<assign-comp>

S_r.b:<assign-case> = VP.t:<assign-case>
NP_0.t:<case> = S_r.b:<assign-case>
N.t:<case> = NP_0.b:<case>


S_r.b:<mode> = VP.t:<mode>
S_r.b:<comp> = nil
S_r.b:<tense> = VP.t:<tense>
NP_0.t:<agr> = S_r.b:<agr>
```

```
NP_0.t:<wh> = -
S_r.b:<agr> = VP.t:<agr>
S_r.b:<conditional> = VP.t:<conditional>
S_r.b:<passive> = VP.t:<passive>
S_r.b:<perfect> = VP.t:<perfect>
S_r.b:<progressive> = VP.t:<progressive>
VP.b:<passive> = -
VP.b:<agr> = V.t:<agr>
VP.b:<mode> = V.t:<mode>
VP.b:<tense> = V.t:<tense>
VP.b:<assign-case> = V.t:<assign-case>
VP.b:<compar> = -
NP_0.b:<agr> = N:<agr>
NP_0.b:<wh> = N:<wh>
S_2:<extracted> = -
S_2:<mode> = ind
S_2:<assign-comp> = ind_nil
S_2:<comp> = that/nil
S_r.t:<inv> = -
PP_w.t:<wh> = +
NP_r.b:<wh> = NP_f.t:<wh>
NP_r.b:<agr> = NP_f.t:<agr>
NP_r.b:<case> = NP_f.t:<case>
NP_f.b:<case> = acc/nom
S_r.t:<comp> = nil
NP_r.b:<rel-clause> = +
NP_f.b:<case> = nom/acc
P_1.t:<assign-case> = PP_1.b:<assign-case>
NP_1:<case> = PP_1.b:<assign-case>
PP_1.b:<wh> = NP_1:<wh>
NP_r.b:<pron> = NP_f.t:<pron>

V.b:<mode> = V_1.b:<mode>
VP.b:<mode> = VP_1.t:<mode>
VP_1.b:<mode> = V_1.t:<mode>
```

6 Tree "betaNcItVpnx1s2"

6.1 graphe



6.2 comments

```
It-cleft with PP as clefted element
simple declarative

e.g.

     It was in the drawing room that the butler did it.
```

6.3 features

```
S_r.b:<assign-comp> = VP.t:<assign-comp>

S_r.b:<assign-case> = VP.t:<assign-case>
NP_0.t:<case> = S_r.b:<assign-case>
N.t:<case> = NP_0.b:<case>


S_r.b:<mode> = VP.t:<mode>
S_r.b:<comp> = nil
S_r.b:<tense> = VP.t:<tense>
NP_0.t:<agr> = S_r.b:<agr>
```

```
NP_0.t:<wh> = -
S_r.b:<agr> = VP.t:<agr>
S_r.b:<conditional> = VP.t:<conditional>
S_r.b:<passive> = VP.t:<passive>
S_r.b:<perfect> = VP.t:<perfect>
S_r.b:<progressive> = VP.t:<progressive>
VP.b:<passive> = -
VP.b:<agr> = V.t:<agr>
VP.b:<mode> = V.t:<mode>
VP.b:<tense> = V.t:<tense>
VP.b:<assign-case> = V.t:<assign-case>
VP.b:<compar> = -
NP_0.b:<agr> = N:<agr>
NP_0.b:<wh> = N:<wh>
S_2:<extracted> = -
S_2:<mode> = ind
S_2:<assign-comp> = ind_nil
S_2:<comp> = that/nil
S_r.t:<inv> = -
S_r.t:<mode> = ind/inf
NP_r.b:<wh> = NP_f.t:<wh>
NP_r.b:<agr> = NP_f.t:<agr>
NP_r.b:<case> = NP_f.t:<case>
NP_f.b:<case> = acc/nom
S_r.t:<mode> = ind/inf
S_r.t:<nocomp-mode> = ind
VP.t:<assign-comp> = that/for/ind_nil
S_r.b:<nocomp-mode> = S_r.b:<mode>
NP_r.b:<rel-clause> = +
NP_f.b:<case> = nom/acc
P_1.t:<assign-case> = PP_1.b:<assign-case>
NP_1:<case> = PP_1.b:<assign-case>
PP_1.b:<wh> = NP_1:<wh>
NP_r.b:<pron> = NP_f.t:<pron>

V.b:<mode> = V_1.b:<mode>
VP.b:<mode> = VP_1.t:<mode>
VP_1.b:<mode> = V_1.t:<mode>
```