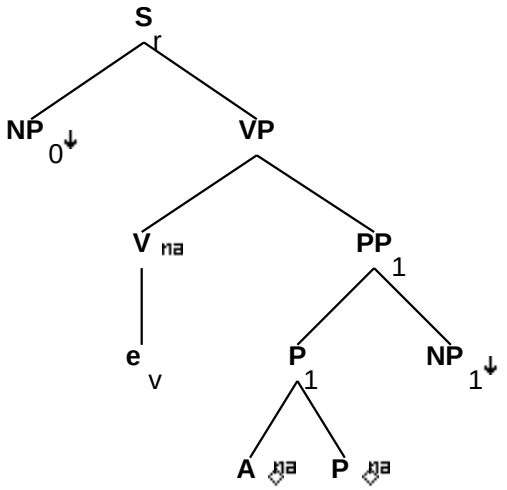
Family "Tnx0APnx1"

March 5, 2008

1 Tree "alphanx0APnx1"

1.1 graphe



1.2 comments

Declarative tree for predicative PPs. This tree family, like other predicative tree families, is anchored by the predicted object (here, the P), with the verb, if any, adjoining in.
EX: She is void of all hope.

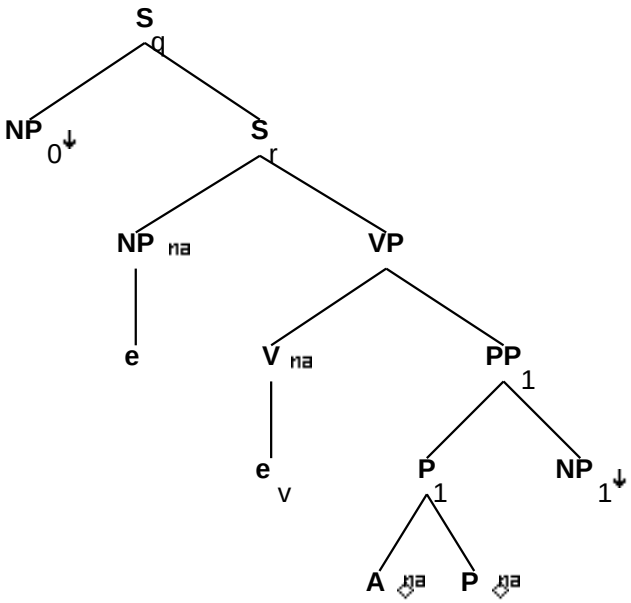
1.3 features

S_r.b:<inv> = -
S_r.b:<comp> = nil
S_r.b:<extracted> = -
S_r.b:<agr> = VP.t:<agr>
S_r.b:<mode> = VP.t:<mode>
S_r.b:<mainv> = VP.t:<mainv>
S_r.b:<tense> = VP.t:<tense>
S_r.b:<assign-comp> = VP.t:<assign-comp>
S_r.b:<assign-case> = VP.t:<assign-case>

```
S_r.b:<agr> = NP_0:<agr>
S_r.b:<assign-case> = NP_0:<case>
S_r.b:<control> = NP_0.t:<control>
S_r.b:<passive> = VP.t:<passive>
VP.t:<passive> = -

NP_0:<wh> = -

VP.b:<mode> = prep
VP.b:<assign-case> = acc

VP.b:<equiv> = PP_1.t:<equiv>
VP.b:<compar> = PP_1.t:<compar>

PP_1.b:<equiv> = P_1.t:<equiv>
PP_1.b:<compar> = P_1.t:<compar>
PP_1.b:<assign-case> = P_1.t:<assign-case>
PP_1.b:<assign-case> = NP_1.t:<case>
PP_1.b:<wh> = NP_1.t:<wh>

P_1.b:<equiv> = A.t:<equiv>
P_1.b:<compar> = A.t:<compar>
```

2 Tree "alphaW0nx0APnx1"

2.1 graphe



2.2 comments

wh subject extraction tree for predicative PPs. This tree does wh+ sentences only, no topicalization, since subject can not topicalize. This tree family, like other predicative tree families, is anchored by the predicted object (here, the P), with the verb, if any, adjoining in.
EX: who is void of all hope?

2.3 features

S_q.b:<comp> = nil
S_q.b:<extracted> = +
S_q.b:<wh> = NP_0.t:<wh>
S_q.b:<inv> = S_r.t:<inv>
S_q.b:<mode> = S_r.t:<mode>

S_r.t:<comp> = nil
S_r.t:<conj> = nil

S_r.b:<inv> = -
S_r.b:<comp> = nil

S_r.b:<assign-comp> = inf_nil/ind_nil/ecm
S_r.b:<agr> = VP.t:<agr>
S_r.b:<mode> = VP.t:<mode>
S_r.b:<tense> = VP.t:<tense>
S_r.b:<assign-case> = VP.t:<assign-case>
S_r.b:<assign-comp> = VP.t:<assign-comp>

S_r.b:<agr> = NP.t:<agr>
S_r.b:<assign-case> = NP.t:<case>

VP.b:<mode> = prep
VP.b:<assign-case> = acc

VP.b:<equiv> = PP_1.t:<equiv>
VP.b:<compar> = PP_1.t:<compar>
VP.t:<passive> = -

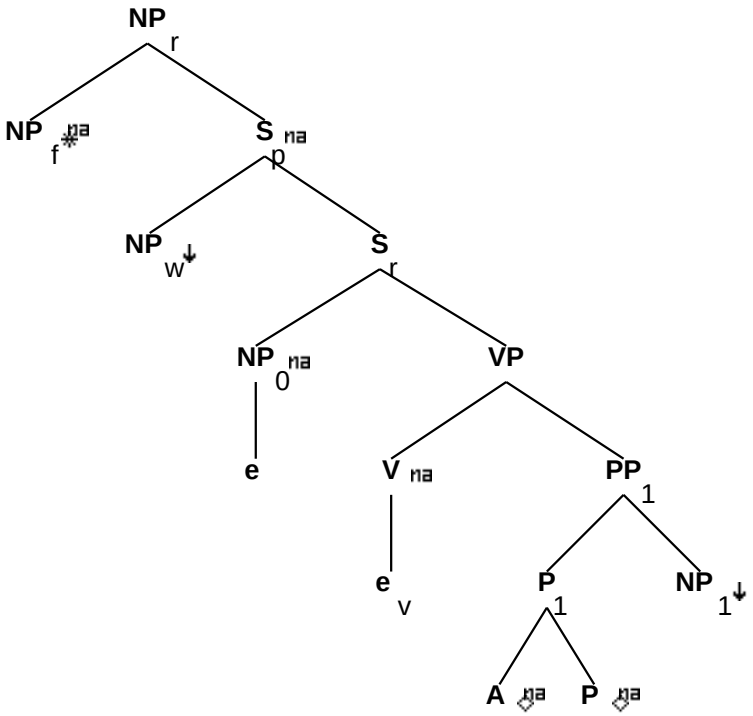
PP_1.b:<equiv> = P_1.t:<equiv>
PP_1.b:<compar> = P_1.t:<compar>
PP_1.b:<assign-case> = P_1.t:<assign-case>
PP_1.b:<assign-case> = NP_1.t:<case>
PP_1.b:<wh> = NP_1.t:<wh>

P_1.b:<equiv> = A.t:<equiv>
P_1.b:<compar> = A.t:<compar>

```
NP_0:<wh> = +
NP:<agr> = NP_0.t:<agr>
NP:<case> = NP_0.t:<case>
NP:<trace> = NP_0.t:<trace>
NP:<wh> = NP_0.t:<wh>
```

3 Tree "betaN0nx0APnx1"

3.1 graphe



3.2 comments

```
relative clause subject extraction tree for predicative PPs.
This tree family, like other predicative tree families, is anchored by the
predicted object (here, the P), with the verb, if any, adjoining in.
EX: the man who is void of all hope ...is feeding the pigeons.
```

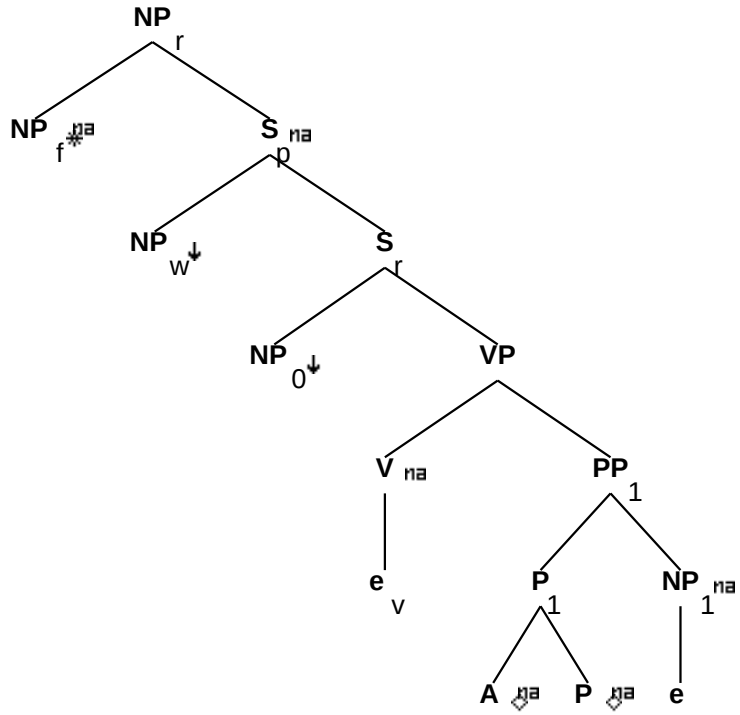
3.3 features

```
NP_r.b:<rel-clause> = +
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_r.b:<pron> = NP_f.t:<pron>
```

```
NP_f.b:<case> = nom/acc
NP_w.t:<wh> = +
NP_w.t:<agr> = NP_0.b:<agr>
NP_w.t:<case> = NP_0.b:<case>
NP_w.t:<trace> = NP_0.b:<trace>
S_r.t:<inv> = -
S_r.t:<conj> = nil
S_r.t:<comp> = nil
S_r.t:<mode> = ind/inf

S_r.b:<comp> = nil

S_r.b:<agr> = VP.t:<agr>
S_r.b:<mode> = VP.t:<mode>
S_r.b:<tense> = VP.t:<tense>
S_r.b:<mainv> = VP.t:<mainv>
S_r.b:<assign-comp> = VP.t:<assign-comp>
S_r.b:<assign-case> = VP.t:<assign-case>

S_r.b:<agr> = NP_0.t:<agr>
S_r.b:<assign-case> = NP_0.t:<case>
S_r.b:<passive> = VP.t:<passive>
VP.t:<passive> = -
VP.b:<mode> = prep
VP.b:<assign-case> = acc

VP.b:<equiv> = PP_1.t:<equiv>
VP.b:<compar> = PP_1.t:<compar>
PP_1.b:<equiv> = P_1.t:<equiv>
PP_1.b:<compar> = P_1.t:<compar>
PP_1.b:<assign-case> = P_1.t:<assign-case>
PP_1.b:<assign-case> = NP_1.t:<case>
P_1.b:<equiv> = A.t:<equiv>
P_1.b:<compar> = A.t:<compar>
```

4 Tree "betaN1nx0APnx1"

4.1 graphe



4.2 comments

relative clause object extraction tree for NP embedded in the predicative PP. This tree family (Tnx0Pnx1), like other predicative tree families, is anchored by the predicted object (here, the P), with the verb, if any, adjoining in. EX: the puzzle piece that these pieces are near to ... are missing

4.3 features

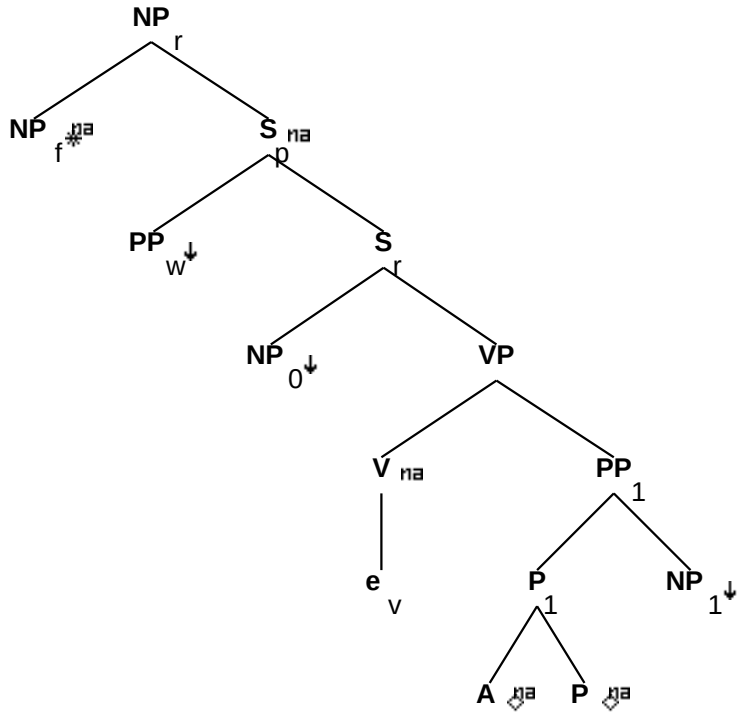
S_r.b:<assign-comp> = VP.t:<assign-comp>

S_r.t:<comp> = nil
S_r.b:<mode> = VP.t:<mode>
S_r.t:<mode> = ind/inf
S_r.b:<tense> = VP.t:<tense>
S_r.t:<inv> = -
S_r.t:<conj> = nil
S_r.b:<inv> = -
S_r.b:<assign-case> = NP_0:<case>
S_r.b:<agr> = NP_0:<agr>

```
S_r.b:<agr> = VP.t:<agr>
S_r.b:<tense> = VP.t:<tense>
S_r.b:<assign-case> = VP.t:<assign-case>
S_r.b:<mainv> = VP.t:<mainv>
S_r.b:<control> = NP_0.t:<control>
S_r.b:<passive> = VP.t:<passive>
VP.t:<passive> = -
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_r.b:<rel-clause> = +
NP_r.b:<pron> = NP_f.t:<pron>


NP_f.b:<case> = nom/acc
NP_w.t:<trace> = NP_1.b:<trace>
NP_w.t:<case> = NP_1.b:<case>
NP_w.t:<agr> = NP_1.b:<agr>
NP_w.t:<wh> = +
VP.b:<mode> = prep
VP.b:<assign-case> = acc

VP.b:<equiv> = PP_1.t:<equiv>
VP.b:<compar> = PP_1.t:<compar>
PP_1.b:<equiv> = P_1.t:<equiv>
PP_1.b:<compar> = P_1.t:<compar>
PP_1.b:<assign-case> = P_1.t:<assign-case>
PP_1.b:<assign-case> = NP_1.t:<case>
P_1.b:<equiv> = A.t:<equiv>
P_1.b:<compar> = A.t:<compar>
```

5 Tree "betaNpxnx0APnx1"

5.1 graphe



5.2 comments

Declarative tree for predicative PPs. This tree family, like other predicative tree families, is anchored by the predicted object (here, the P), with the verb, if any, adjoining in.
EX: She is void of all hope.

5.3 features

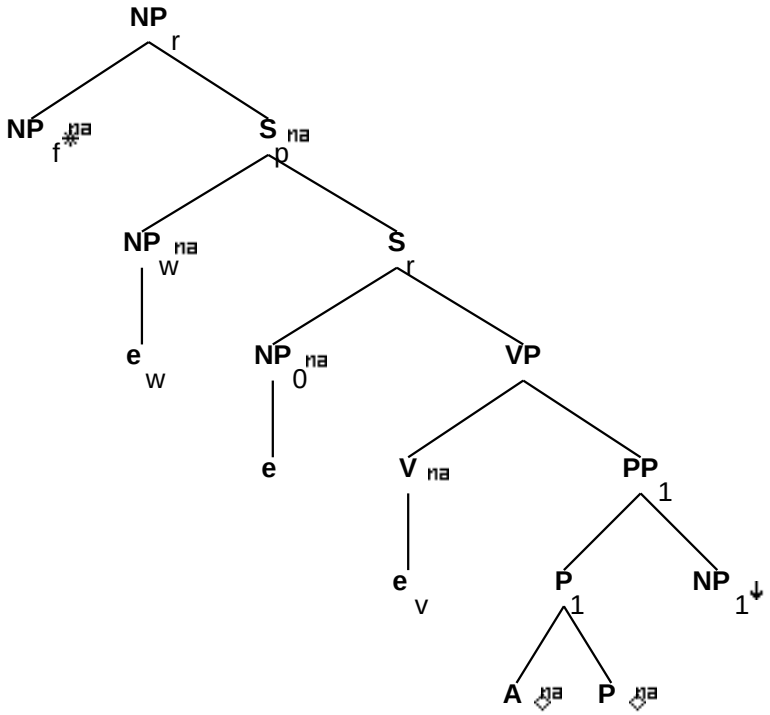
S_r.b:<extracted> = -
S_r.b:<inv> = -
S_r.b:<assign-comp> = VP.t:<assign-comp>

S_r.b:<mode> = VP.t:<mode>
S_r.b:<mainv> = VP.t:<mainv>
S_r.b:<comp> = nil
S_r.b:<tense> = VP.t:<tense>
NP_0:<agr> = S_r.b:<agr>
NP_0:<case> = S_r.b:<assign-case>
NP_0:<wh> = -

```
S_r.b:<agr> = VP.t:<agr>
S_r.b:<assign-case> = VP.t:<assign-case>
S_r.b:<passive> = VP.t:<passive>
VP.t:<passive> = -
VP.b:<mode> = prep
VP.b:<assign-case> = acc
PP_1.b:<assign-case> = P_1.t:<assign-case>
PP_1.b:<assign-case> = NP_1.t:<case>
S_r.b:<control> = NP_0.t:<control>
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
NP_r.b:<pron> = NP_f.t:<pron>

VP.b:<equiv> = PP_1.t:<equiv>
VP.b:<compar> = PP_1.t:<compar>
PP_1.b:<equiv> = P_1.t:<equiv>
PP_1.b:<compar> = P_1.t:<compar>
P_1.b:<equiv> = A.t:<equiv>
P_1.b:<compar> = A.t:<compar>
```

6 Tree "betaNc0nx0APnx1"

6.1 graphe



6.2 comments

relative clause subject extraction tree for predicative PPs.
This tree family, like other predicative tree families, is anchored by the predicted object (here, the P), with the verb, if any, adjoining in.
EX: the man who is void of all hope ...is feeding the pigeons.

6.3 features

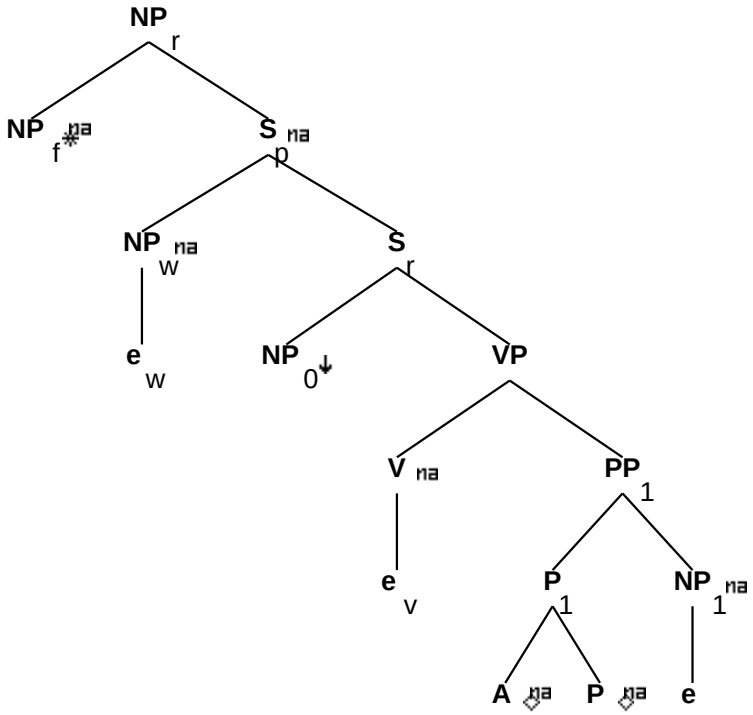
S_r.b:<assign-comp> = VP.t:<assign-comp>

S_r.b:<mode> = VP.t:<mode>
S_r.b:<comp> = nil
S_r.b:<tense> = VP.t:<tense>
S_r.t:<inv> = -
S_r.b:<agr> = VP.t:<agr>
S_r.b:<assign-case> = VP.t:<assign-case>
S_r.b:<mainv> = VP.t:<mainv>
S_r.b:<agr> = NP_0.t:<agr>

```
S_r.b:<assign-case> = NP_0.t:<case>
S_r.b:<passive> = VP.t:<passive>
VP.t:<passive> = -
VP.b:<mode> = prep
VP.b:<assign-case> = acc
PP_1.b:<assign-case> = P_1.t:<assign-case>
PP_1.b:<assign-case> = NP_1.t:<case>
NP_r.b:<wh> = NP_f.t:<wh>
NP_r.b:<agr> = NP_f.t:<agr>
NP_r.b:<case> = NP_f.t:<case>
S_r.t:<conj> = nil

NP_w.t:<trace> = NP_0.b:<trace>
NP_w.t:<case> = NP_0.b:<case>
NP_w.t:<agr> = NP_0.b:<agr>
NP_r.b:<rel-clause> = +
S_r.t:<mode> = inf/ger/ind/prep
S_r.t:<nocomp-mode> = inf/ger/prep
VP.t:<assign-comp> = that/ind_nil/inf_nil/ecm
S_r.b:<nocomp-mode> = S_r.b:<mode>
NP_f.b:<case> = nom/acc
NP_r.b:<pron> = NP_f.t:<pron>

VP.b:<equiv> = PP_1.t:<equiv>
VP.b:<compar> = PP_1.t:<compar>
PP_1.b:<equiv> = P_1.t:<equiv>
PP_1.b:<compar> = P_1.t:<compar>
P_1.b:<equiv> = A.t:<equiv>
P_1.b:<compar> = A.t:<compar>
```

7 Tree "betaNc1nx0APnx1"

7.1 graphe



7.2 comments

relative clause object extraction tree for NP embedded in the predicative PP. This tree family (Tnx0Pnx1), like other predicative tree families, is anchored by the predicted object (here, the P), with the verb, if any, adjoining in. EX: the puzzle piece that these pieces are near to ... are missing

7.3 features

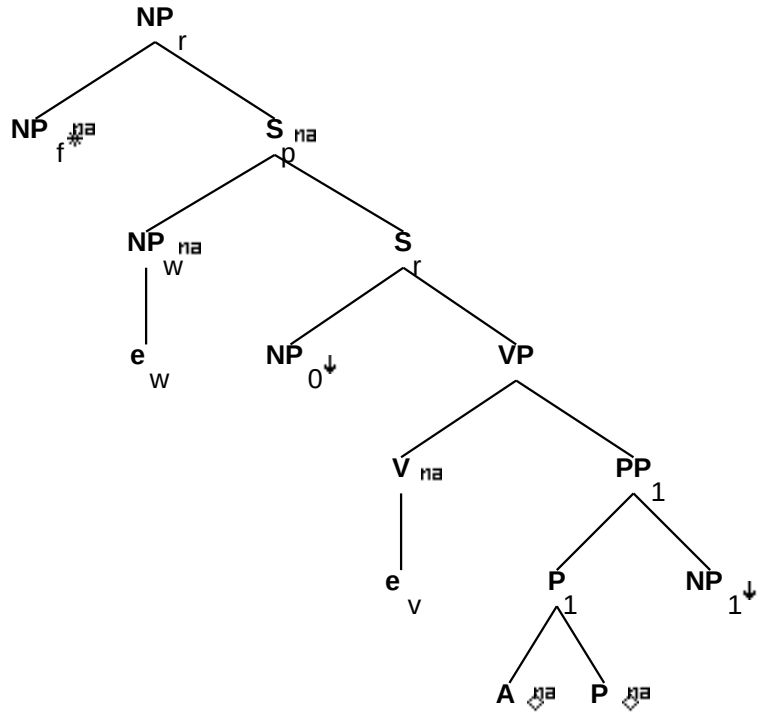
S_r.b:<assign-comp> = VP.t:<assign-comp>

S_r.b:<mode> = VP.t:<mode>
S_r.b:<tense> = VP.t:<tense>
S_r.t:<inv> = -
S_r.b:<inv> = -
NP_0:<agr> = S_r.b:<agr>
NP_0:<case> = S_r.b:<assign-case>
S_r.b:<agr> = VP.t:<agr>
S_r.b:<tense> = VP.t:<tense>

```
S_r.b:<assign-case> = VP.t:<assign-case>
S_r.b:<mainv> = VP.t:<mainv>
S_r.b:<control> = NP_0.t:<control>
S_r.b:<passive> = VP.t:<passive>
VP.t:<passive> = -
VP.b:<mode> = prep
VP.b:<assign-case> = acc
PP_1.b:<assign-case> = P_1.t:<assign-case>
PP_1.b:<assign-case> = NP_1.t:<case>
NP_r.b:<wh> = NP_f.t:<wh>
NP_r.b:<agr> = NP_f.t:<agr>
NP_r.b:<case> = NP_f.t:<case>
S_r.t:<conj> = nil

NP_w.t:<trace> = NP_1.b:<trace>
NP_w.t:<case> = NP_1.b:<case>
NP_w.t:<agr> = NP_1.b:<agr>
NP_r.b:<rel-clause> = +
S_r.t:<mode> = inf/ind
S_r.t:<nocomp-mode> = ind
VP.t:<assign-comp> = that/for/ind_nil
S_r.b:<nocomp-mode> = S_r.b:<mode>
NP_f.b:<case> = nom/acc
NP_r.b:<pron> = NP_f.t:<pron>

VP.b:<equiv> = PP_1.t:<equiv>
VP.b:<compar> = PP_1.t:<compar>
PP_1.b:<equiv> = P_1.t:<equiv>
PP_1.b:<compar> = P_1.t:<compar>
P_1.b:<equiv> = A.t:<equiv>
P_1.b:<compar> = A.t:<compar>
```

8 Tree "betaNcnx0APnx1"

8.1 graphe



8.2 comments

Declarative tree for predicative PPs. This tree family, like other predicative tree families, is anchored by the predicted object (here, the P), with the verb, if any, adjoining in.
EX: She is void of all hope.

8.3 features

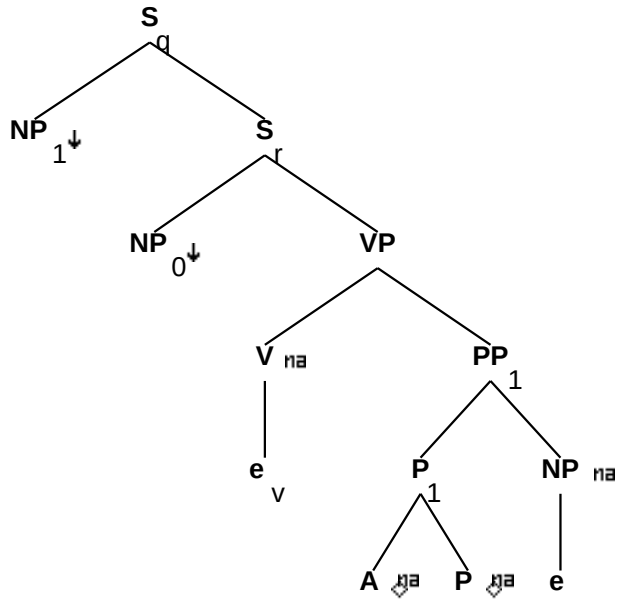
S_r.b:<extracted> = -
S_r.b:<inv> = -
S_r.b:<assign-comp> = VP.t:<assign-comp>

S_r.b:<mode> = VP.t:<mode>
S_r.b:<mainv> = VP.t:<mainv>
S_r.b:<comp> = nil
S_r.b:<tense> = VP.t:<tense>
NP_0:<agr> = S_r.b:<agr>
NP_0:<case> = S_r.b:<assign-case>
NP_0:<wh> = -

```
S_r.b:<agr> = VP.t:<agr>
S_r.b:<assign-case> = VP.t:<assign-case>
S_r.b:<passive> = VP.t:<passive>
VP.t:<passive> = -
VP.b:<mode> = prep
VP.b:<assign-case> = acc
PP_1.b:<assign-case> = P_1.t:<assign-case>
PP_1.b:<assign-case> = NP_1.t:<case>
S_r.b:<control> = NP_0.t:<control>
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:<inv> = -
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
NP_r.b:<pron> = NP_f.t:<pron>

VP.b:<equiv> = PP_1.t:<equiv>
VP.b:<compar> = PP_1.t:<compar>
PP_1.b:<equiv> = P_1.t:<equiv>
PP_1.b:<compar> = P_1.t:<compar>
P_1.b:<equiv> = A.t:<equiv>
P_1.b:<compar> = A.t:<compar>
```

9 Tree "alphaW1nx0APnx1"

9.1 graphe



9.2 comments

wh object extraction tree for predicative PPs. This tree does wh+ sentences only, no topicalization, since subject can not topicalize. This tree family, like other predicative tree families, is anchored by the predicted object (here, the A and P), with the verb, if any, adjoining in.

EX: I know how to get to the mall and the movie theater. What is Bill's house nearer to?

9.3 features

S_q.b:<extracted> = +

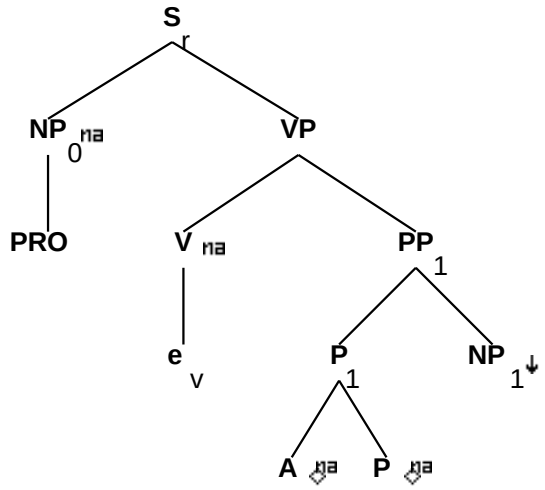
S_q.b:<inv> = S_r.t:<inv>
S_q.b:<inv> = S_q.b:<invlink>
S_q.b:<wh> = NP_1.t:<wh>
S_r.t:<comp> = nil
S_r.b:<assign-comp> = VP.t:<assign-comp>

S_q.b:<comp> = nil
S_q.b:<mode> = S_r.t:<mode>
S_r.b:<mode> = VP.t:<mode>
S_r.b:<mainv> = VP.t:<mainv>
S_r.b:<comp> = nil

```
S_r.b:<tense> = VP.t:<tense>
S_r.b:<inv> = -
NP:<trace> = NP_1.t:<trace>
NP:<agr> = NP_1.t:<agr>
NP:<case> = NP_1.t:<case>
NP:<wh> = NP_1.t:<wh>
S_r.b:<agr> = VP.t:<agr>
S_r.b:<assign-case> = VP.t:<assign-case>
S_r.b:<agr> = NP_0.t:<agr>
S_r.b:<assign-case> = NP_0.t:<case>
S_r.b:<control> = NP_0.t:<control>
S_r.b:<passive> = VP.t:<passive>
VP.t:<passive> = -
VP.b:<mode> = prep
VP.b:<assign-case> = acc
PP_1.b:<assign-case> = P_1.t:<assign-case>
PP_1.b:<assign-case> = NP.t:<case>
PP_1.b:<wh> = NP.t:<wh>
S_r.t:<conj> = nil
VP.b:<equiv> = PP_1.t:<equiv>
VP.b:<compar> = PP_1.t:<compar>
PP_1.b:<equiv> = P_1.t:<equiv>
PP_1.b:<compar> = P_1.t:<compar>
P_1.b:<equiv> = A.t:<equiv>
P_1.b:<compar> = A.t:<compar>
```

10 Tree "alphanx0APnx1-PRO"

10.1 graphe



10.2 comments

Predicative PPs (Adj and Prep) w/ PRO subject. This tree family, like other predicative tree families, is anchored by the predicated object (here, the multiword P), with the verb, if any, adjoining in.

Mary doesn't want [PRO to be void of all hope].

10.3 features

S_r.b:<inv> = -
S_r.b:<comp> = nil
S_r.b:<extracted> = -
S_r.b:<agr> = VP.t:<agr>
S_r.b:<mode> = VP.t:<mode>
S_r.b:<mainv> = VP.t:<mainv>
S_r.b:<tense> = VP.t:<tense>
S_r.b:<assign-comp> = VP.t:<assign-comp>
S_r.b:<agr> = NP_0:<agr>
S_r.b:<control> = NP_0.t:<control>
S_r.b:<assign-case> = NP_0.t:<case>
S_r.b:<passive> = VP.t:<passive>
NP_0:<wh> = -
NP_0.t:<case> = none
VP.t:<passive> = -
VP.t:<mode> = inf/ger
VP.b:<mode> = prep
VP.b:<assign-case> = acc
VP.b:<equiv> = PP_1.t:<equiv>
VP.b:<compar> = PP_1.t:<compar>
PP_1.b:<equiv> = P_1.t:<equiv>
PP_1.b:<compar> = P_1.t:<compar>
PP_1.b:<assign-case> = P_1.t:<assign-case>
PP_1.b:<assign-case> = NP_1.t:<case>
PP_1.b:<wh> = NP_1.t:<wh>
P_1.b:<equiv> = A.t:<equiv>
P_1.b:<compar> = A.t:<compar>