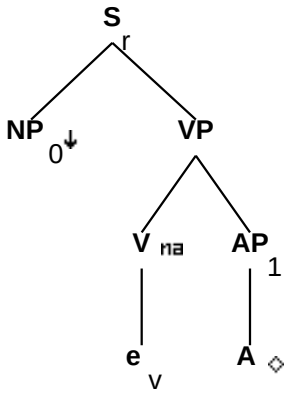
Family "Tnx0Ax1"

March 5, 2008

1 Tree "alphanx0Ax1"

1.1 graphe



1.2 comments

```
Predicative Adjective.
Small Clause construction.
Simple declarative.

Predicative 'be' is forced to adjoin in matrix clauses.
Adjunction of predicative 'be' is optional in embedded clauses

e.g.
     Ernest is stupid.
     Everyone considers Ernest stupid.
     Everyone thinks Ernest is stupid.
```

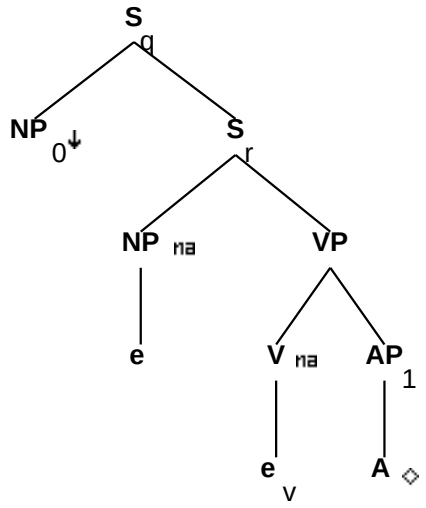
1.3 features

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

```
NP_0:<wh> = -

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


S_r.b:<agr> = VP.t:<agr>
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:<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:<passive> = VP.t:<passive>
VP.t:<passive> = -

AP_1.b:<equiv> = A.t:<equiv>
AP_1.b:<compar> = A.t:<compar>
AP_1.b:<wh> = A.t:<wh>
```

2 Tree "alphaW0nx0Ax1"

2.1 graphe



2.2 comments

Predicative Adjective
Small Clause construction
Wh-extraction on the subject

e.g.

Who is stupid?
Who does everyone consider stupid?
Who does everyone think is stupid?

2.3 features

S_q.b:<extracted> = +

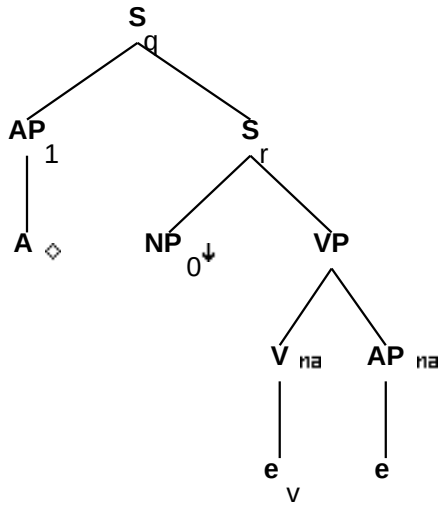
S_q.b:<inv> = S_r.t:<inv>
S_q.b:<wh> = NP_0.t:<wh>
S_r.t:<comp> = nil
S_r.b:<assign-comp> = VP.t:<assign-comp>

S_q.b:<comp> = nil
S_q.b:<mode> = S_r.t:<mode>
S_r.b:<mode> = VP.t:<mode>
S_r.b:<comp> = nil
S_r.b:<tense> = VP.t:<tense>
S_r.b:<inv> = -
S_r.b:<assign-case> = NP.t:<case>
S_r.b:<agr> = NP.t:<agr>
NP:<trace> = NP_0:<trace>
NP_0:<agr> = NP.t:<agr>
NP_0:<case> = NP.t:<case>
NP:<wh> = NP_0:<wh>
NP_0:<wh> = +
S_r.b:<agr> = VP.t:<agr>
S_r.b:<assign-case> = VP.t:<assign-case>
VP.b:<mode> = nom
VP.b:<assign-case> = acc
S_r.t:<conj> = nil
S_r.b:<assign-comp> = inf_nil/ind_nil/ecm

A.t:<compar> = AP_1.b:<compar>
A.t:<equiv> = AP_1.b:<equiv>
AP_1.b:<wh> = A.t:<wh>
VP.b:<compar> = -
VP.t:<passive> = -

3 Tree "alphaWA1nx0Ax1"

3.1 graphe



3.2 comments

```
Predicative Adjective
Small clause construction
Wh-extraction on the predicative adjective

e.g.

     What/How is Ernest?
     What/How does everyone consider Ernest?
     What/How does everyone think Ernest is?
```

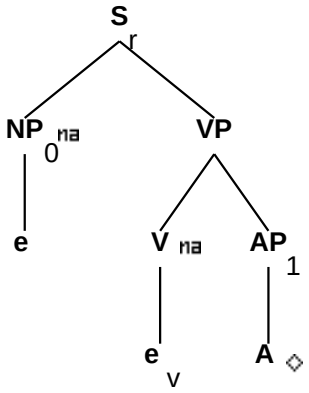
3.3 features

```
S_q.b:<extracted> = +

S_q.b:<inv> = S_r.t:<inv>
S_q.b:<inv> = S_q.b:<invlink>
AP_1.t:<wh> = S_q.b:<wh>
S_r.t:<comp> = nil
S_r.b:<assign-comp> = VP.t:<assign-comp>



S_q.b:<mode> = S_r.t:<mode>
S_q.b:<comp> = nil
S_r.b:<mode> = VP.t:<mode>
S_r.b:<comp> = nil
S_r.b:<inv> = -
```

NP_0:<agr> = S_r.b:<agr>
NP_0:<case> = S_r.b:<assign-case>
S_r.b:<tense> = VP.t:<tense>
S_r.b:<agr> = VP.t:<agr>
S_r.b:<assign-case> = VP.t:<assign-case>
AP.t:<trace> = AP_1.t:<trace>
AP.t:<wh> = AP_1.t:<wh>
AP_1.b:<wh> = A.t:<wh>
AP_1.b:<equiv> = A.t:<equiv>
AP_1.b:<compar> = A.t:<compar>
S_r.b:<mainv> = VP.t:<mainv>
VP.b:<mode> = nom
VP.b:<assign-case> = acc
S_r.t:<conj> = nil
S_r.b:<control> = NP_0.t:<control>
S_r.b:<passive> = VP.t:<passive>
VP.t:<passive> = -
VP.b:<compar> = -

4 Tree "alphaInx0Ax1"

4.1 graphe



4.2 comments

Predicative Adjective
Small clause construction
Imperative

e.g.

Be stupid.

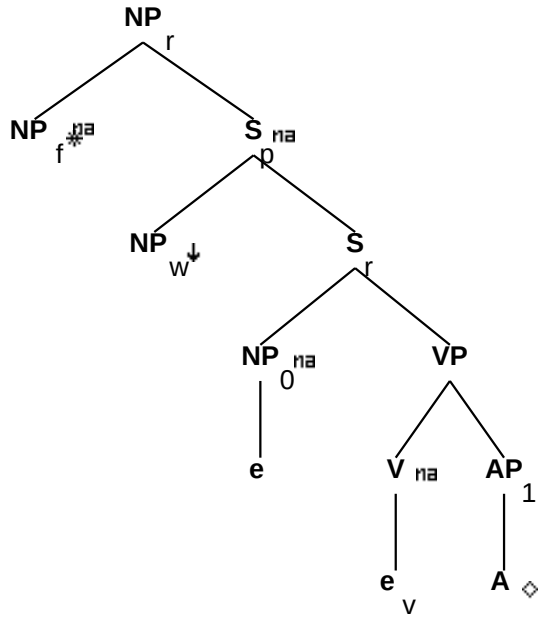
4.3 features

S_r.b:<extracted> = -

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



S_r.b:<mode> = imp
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> = -
NP_0:<agr pers> = 2
NP_0:<agr 3rdsing> = -
NP_0:<agr num> = plur/sing
NP_0:<case> = nom
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.t:<tense> = pres
VP.t:<mode> = base
VP.t:<neg> = -
VP.b:<mode> = nom
VP.b:<assign-case> = acc
A.t:<compar> = AP_1.b:<compar>
A.t:<equiv> = AP_1.b:<equiv>
AP_1.b:<wh> = A.t:<wh>
VP.b:<compar> = -
```

5 Tree "betaN0nx0Ax1"

5.1 graphe



5.2 comments

```
Predicative Adjective
Small clause construction
Relative clause on the subject


e.g.

     [the person] who is stupid
     [the most likely guy] to be stupid
```

5.3 features

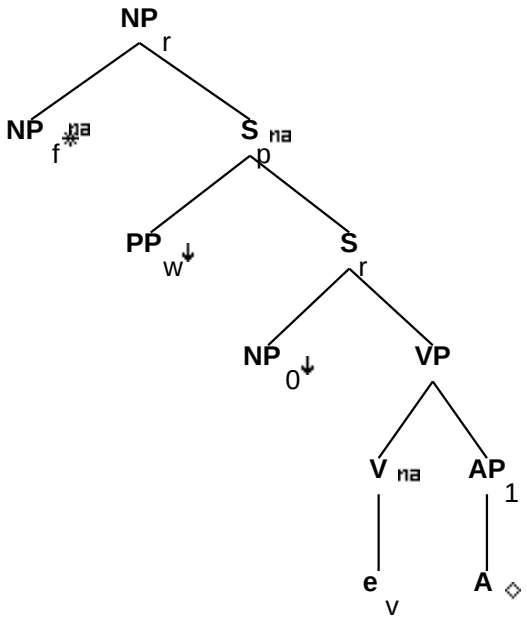
```
S_r.b:<assign-comp> = VP.t:<assign-comp>
```

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

```
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:<passive> = VP.t:<passive>
VP.t:<passive> = -
VP.b:<mode> = nom
VP.b:<assign-case> = acc
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_w.t:<wh> = +
S_r.t:<comp> = nil
NP_r.b:<rel-clause> = +
NP_f.b:<case> = nom/acc
A.t:<compar> = AP_1.b:<compar>
A.t:<equiv> = AP_1.b:<equiv>
VP.b:<compar> = -
NP_r.b:<pron> = NP_f.t:<pron>
```

6 Tree "betaNpxnx0Ax1"

6.1 graphe



6.2 comments

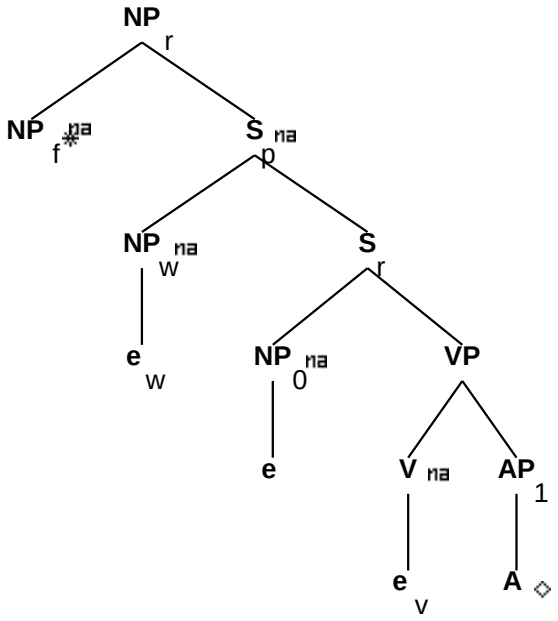
```
Predicative Adjective.
Small Clause construction.
Simple declarative.

Predicative 'be' is forced to adjoin in matrix clauses.
Adjunction of predicative 'be' is optional in embedded clauses

e.g.
     Ernest is stupid.
     Everyone considers Ernest stupid.
     Everyone thinks Ernest is stupid.
```

6.3 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> = nom
VP.b:<assign-case> = acc
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
A.t:<compar> = AP_1.b:<compar>
A.t:<equiv> = AP_1.b:<equiv>
VP.b:<compar> = -
NP_r.b:<pron> = NP_f.t:<pron>
```

7 Tree "betaNc0nx0Ax1"

7.1 graphe



7.2 comments

```
Predicative Adjective
Small clause construction
Relative clause on the subject


e.g.

     [the person] who is stupid
     [the most likely guy] to be stupid
```

7.3 features

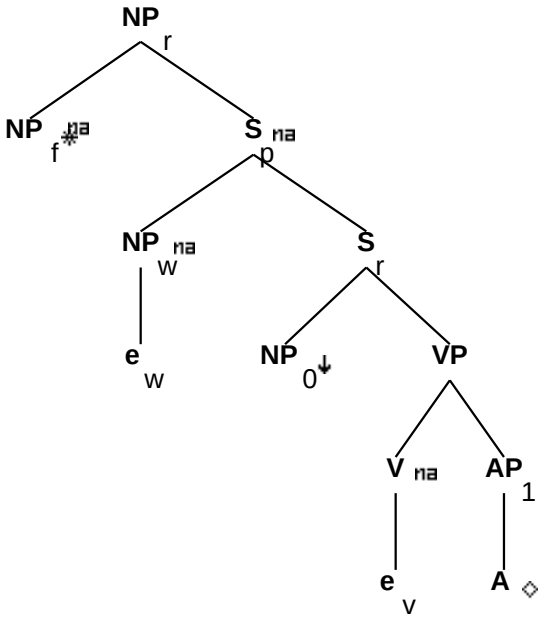
```
S_r.b:<assign-comp> = VP.t:<assign-comp>
```

```
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:<assign-case> = NP_0.t:<case>
S_r.b:<agr> = NP_0.t:<agr>
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:<passive> = VP.t:<passive>
VP.t:<passive> = -
VP.b:<mode> = nom
VP.b:<assign-case> = acc
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
S_r.t:<nocomp-mode> = inf/ger
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
A.t:<compar> = AP_1.b:<compar>
A.t:<equiv> = AP_1.b:<equiv>
VP.b:<compar> = -
NP_r.b:<pron> = NP_f.t:<pron>
```

8 Tree "betaNcnx0Ax1"

8.1 graphe



8.2 comments

```
Predicative Adjective.
Small Clause construction.
Simple declarative.

Predicative 'be' is forced to adjoin in matrix clauses.
Adjunction of predicative 'be' is optional in embedded clauses

e.g.
     Ernest is stupid.
     Everyone considers Ernest stupid.
     Everyone thinks Ernest is stupid.
```

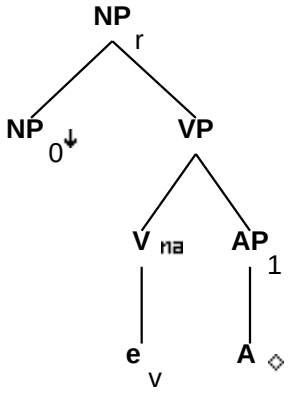
8.3 features

```
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> = nom
VP.b:<assign-case> = acc
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
A.t:<compar> = AP_1.b:<compar>
A.t:<equiv> = AP_1.b:<equiv>
VP.b:<compar> = -
```

NP_r.b:<pron> = NP_f.t:<pron>

9 Tree "alphaGnx0Ax1"

9.1 graphe



9.2 comments

Predicative adjective
Small clause construction
Gerund NP

...Ernest('s) being stupid...

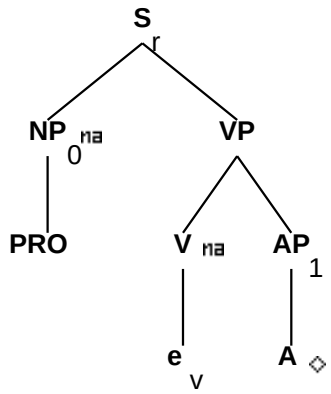
9.3 features

NP_0:<wh> = NP_r.b:<wh>
VP.t:<mode> = ger
NP_r.b:<case> = nom/acc
NP_r.b:<agr num> = sing
NP_r.b:<agr pers> = 3
NP_r.b:<agr 3rdsing> = +
VP.b:<mode> = nom
VP.b:<assign-case> = acc

NP_r.b:<gerund> = +
A.t:<compar> = AP_1.b:<compar>
A.t:<equiv> = AP_1.b:<equiv>
VP.b:<compar> = -
NP_0:<case> = acc/gen

10 Tree "alphanx0Ax1-PRO"

10.1 graphe



10.2 comments

```
Predicative Adjective.
Small Clause construction.
PRO subject

Ernest doesn't want [PRO to be stupid].
While [PRO being stupid] Ernest shot his eye out.
```

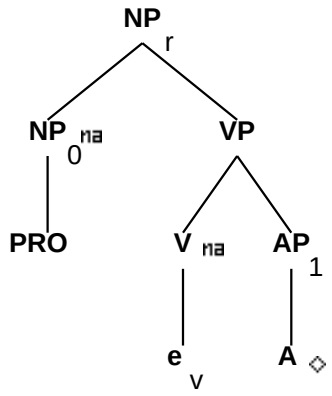
10.3 features

```
S_r.b:<inv> = -
S_r.b:<comp> = nil
S_r.b:<extracted> = -
NP_0:<wh> = -
VP.b:<compar> = -
VP.b:<mode> = nom
S_r.b:<agr> = VP.t:<agr>
S_r.b:<agr> = NP_0:<agr>
S_r.b:<control> = NP_0.t:<control>
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:<passive> = VP.t:<passive>
VP.t:<passive> = -
AP_1.b:<equiv> = A.t:<equiv>
AP_1.b:<compar> = A.t:<compar>
AP_1.b:<wh> = A.t:<wh>
VP.t:<mode> = inf/ger
NP_0.t:<wh> = -
NP_0.t:<case> = none
```

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

11 Tree "alphaGnx0Ax1-PRO"

11.1 graphe



11.2 comments

Predicative adjective
Small clause construction
Gerund NP w/ PRO subject

[PRO being silly] was all that Ernest knew how to do.

11.3 features

NP_0:<wh> = NP_r.b:<wh>
VP.t:<mode> = ger
NP_r.b:<case> = nom/acc
NP_r.b:<agr num> = sing
NP_r.b:<agr pers> = 3
NP_r.b:<agr 3rdsing> = +
VP.b:<mode> = nom
NP_r.b:<gerund> = +
A.t:<compar> = AP_1.b:<compar>
A.t:<equiv> = AP_1.b:<equiv>
VP.b:<compar> = -
NP_0.t:<wh> = -
NP_0.t:<case> = none