TreeBanks and Wide-Coverage Parsing

Logische Grammatica's

Richard Moot

Question:

How can we parse arbitrary text?

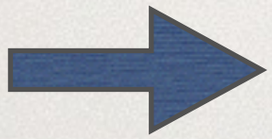
Two Solutions

Writing grammars by hand is very useful and very important, but not well-suited to wide-coverage parsing

1. Program a grammar by hand
2. Machine learning

Machine learning needs lots of examples: the more, the better. Where do we get these examples? Treebanks contain a lot of information we can exploit. So we use treebank transformations to make them as close as possible to what we really want.

Motivation:
Why Treebanks?

Treebank  Wide-coverage parser

Motivation:
Why Treebanks?

Treebank → Wide-coverage parser

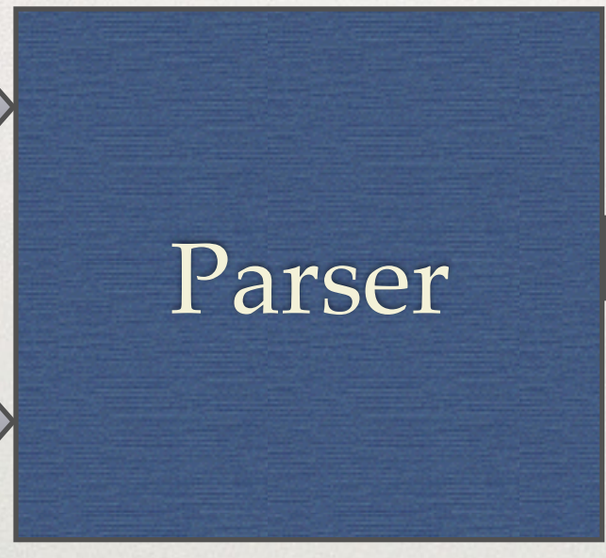
Type-logical treebank → Wide-coverage semantics

Why Are We Parsing?

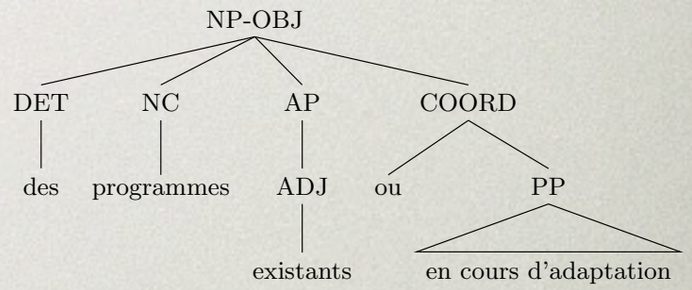
Why Are We Parsing?

grammar

string

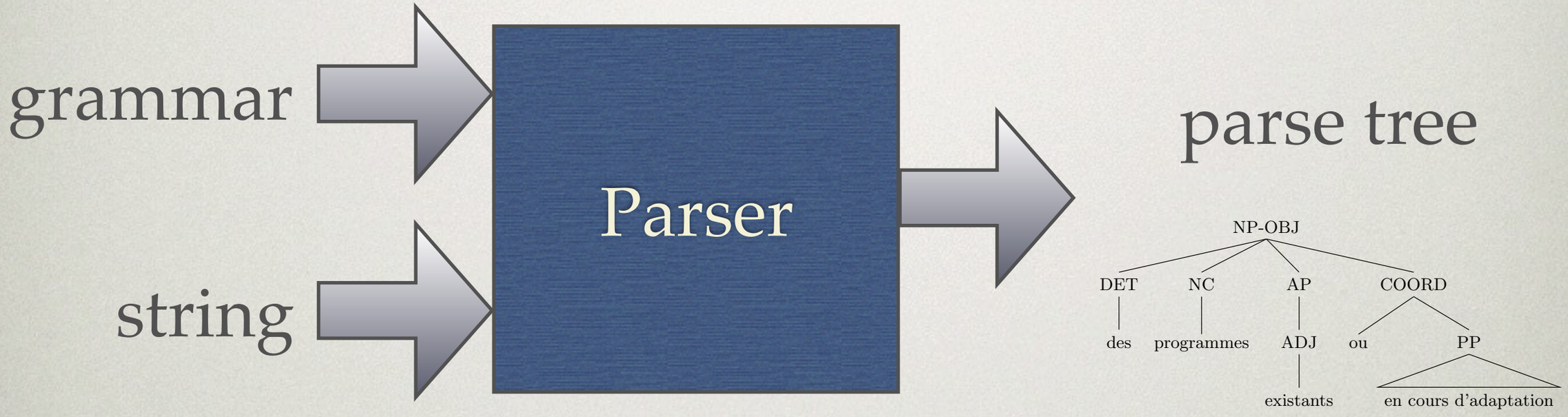


parse tree



or *no*

Why Are We Parsing?



Modern parsers do not ask the question: "is this sentence in the grammar and, if so, what is its analysis?" but the question: "if this sentence would be in the grammar, what would its (most likely) parse be?"
This is more forgiving to the people typing a string hoping to interact with the computer (less so for people interested mainly in language classes).
BONUS: better performance

Why Are We Parsing?

grammar

string

Parser

parse tree

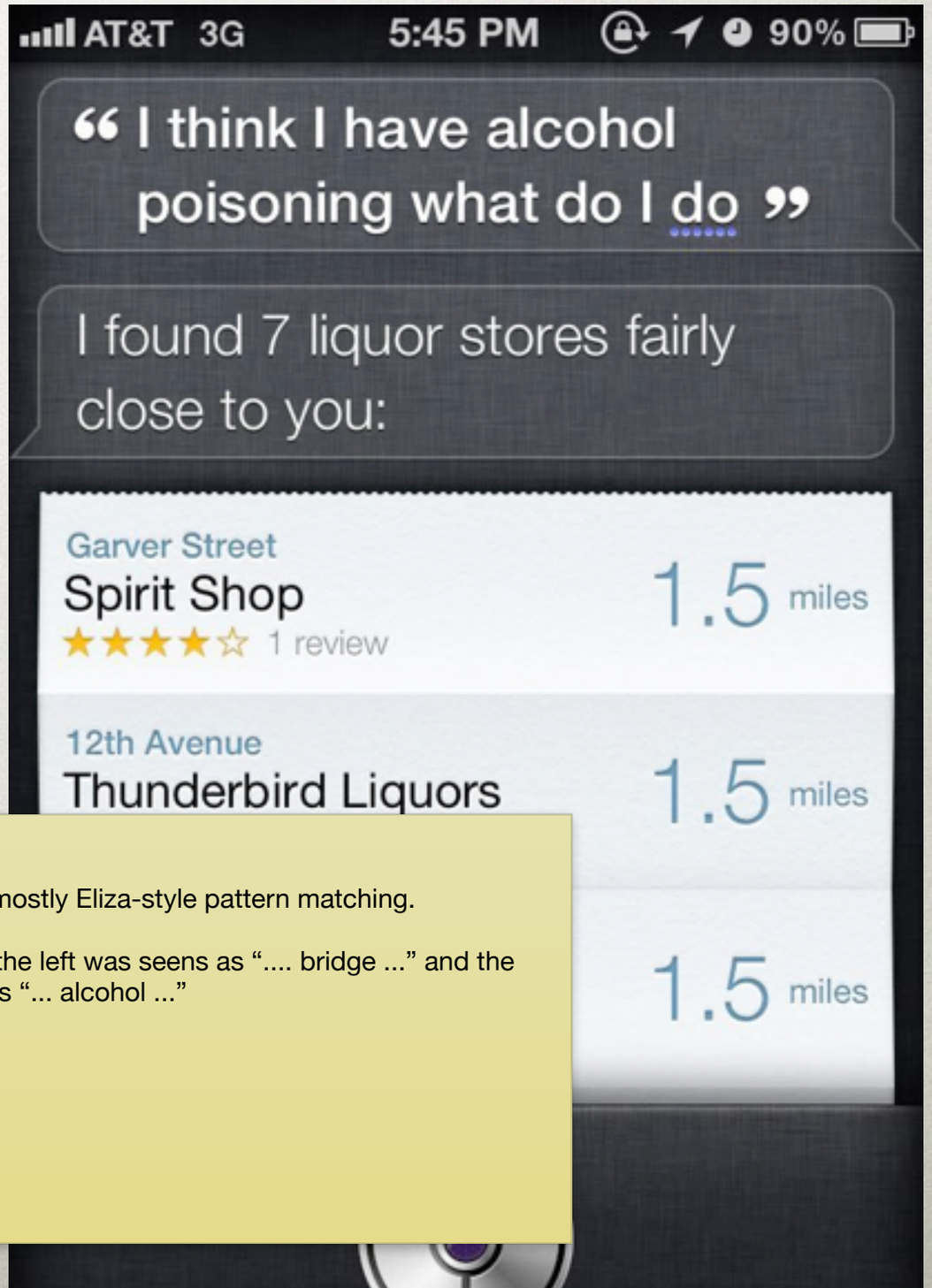
$$\dfrac{\dfrac{\dfrac{\text{de}}{(np\backslash s_{di})/(np\backslash s_i)}\,[Lex] \quad \dfrac{\dfrac{\text{s'}}{cl_r}\,[Lex] \quad \dfrac{\dfrac{\text{attaquer}}{(cl_r\backslash(np\backslash s_i))/pp_{\grave{a}}}\,[Lex] \quad \dfrac{}{\text{p}_0 \vdash pp_{\grave{a}}}\,[Hyp]_1}{\text{a} \circ \text{p}_0 \vdash cl_r\backslash(np\backslash s_i)}\,[/E]}{\text{s'} \circ (\text{a} \circ \text{p}_0) \vdash np\backslash s_i}\,[\backslash E]}{\text{de} \circ (\text{s'} \circ (\text{a} \circ \text{p}_0)) \vdash np\backslash s_i}\,[/E]$$

semantics

$$(\lambda x.\exists y.(x\ y)\ \lambda z.((\lambda v.\lambda w.\textbf{\textit{shot}}(v,w)\ z)\ \textbf{\textit{vito}}))$$

$e_1\ y_1$	
y_1 :	$e_2\ e_3\ x_3$
	$x_3 =$ **?**
	aider_à(e_2, x_0, x_3, e_3)
	partir(e_3, x_3)
demander(e_1, y_0, x_0, y_1)	

Question Answering Natural Language Interface



Siri seems to do mostly Eliza-style pattern matching.

The sentence on the left was seens as “.... bridge ...” and the one on the right as “... alcohol ...”

Question Answering Natural Language Interface

Wanted for general evilness; last seen at the tower of Barad-Dur; it's a giant eye, folks, kinda hard to miss

Watson does use deep parsing and rather successfully

Question Answering Natural Language Interface

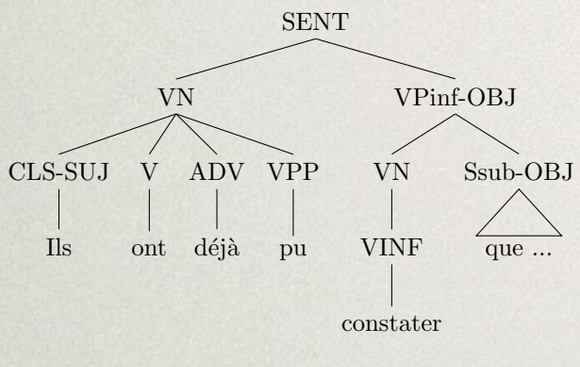
U.S. Cities

Its largest airport is named for a World War II hero; its second largest, for a World War II battle

Still, there seems to be room for improvement in the parser component: here the second sentence should be interpreted as “its second largest [airport] [is named] for a World War II battle.

Outline

Treebank



Grammar

Extraction

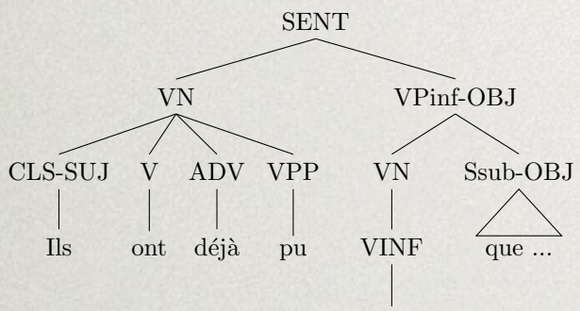
$$\dfrac{\dfrac{\text{de}}{(np\backslash s_{di})/(np\backslash s_i)}\,[Lex] \quad \dfrac{\dfrac{\text{s'}}{cl_r}\,[Lex] \quad \dfrac{\dfrac{\text{attaquer}}{(cl_r\backslash(np\backslash s_i))/pp_{\grave{a}}}\,[Lex] \quad \dfrac{}{\text{p}_0 \vdash pp_{\grave{a}}}\,[Hyp]_1}{\text{a} \circ \text{p}_0 \vdash cl_r\backslash(np\backslash s_i)}\,[/E]}{\text{s'} \circ (\text{a} \circ \text{p}_0) \vdash np\backslash s_i}\,[\backslash E]}{\text{de} \circ (\text{s'} \circ (\text{a} \circ \text{p}_0)) \vdash np\backslash s_i}\,[/E]$$

Applications

$e_1\ y_1$	
y_1 :	$e_2\ e_3\ x_3$
	$x_3 = \mathbf{?}$
	$\mathbf{aider\_à}(e_2, x_0, x_3, e_3)$
	$\mathbf{partir}(e_3, x_3)$
$\mathbf{demander}(e_1, y_0, x_0, y_1)$	

Outline

Treebank



Grammar

Extraction

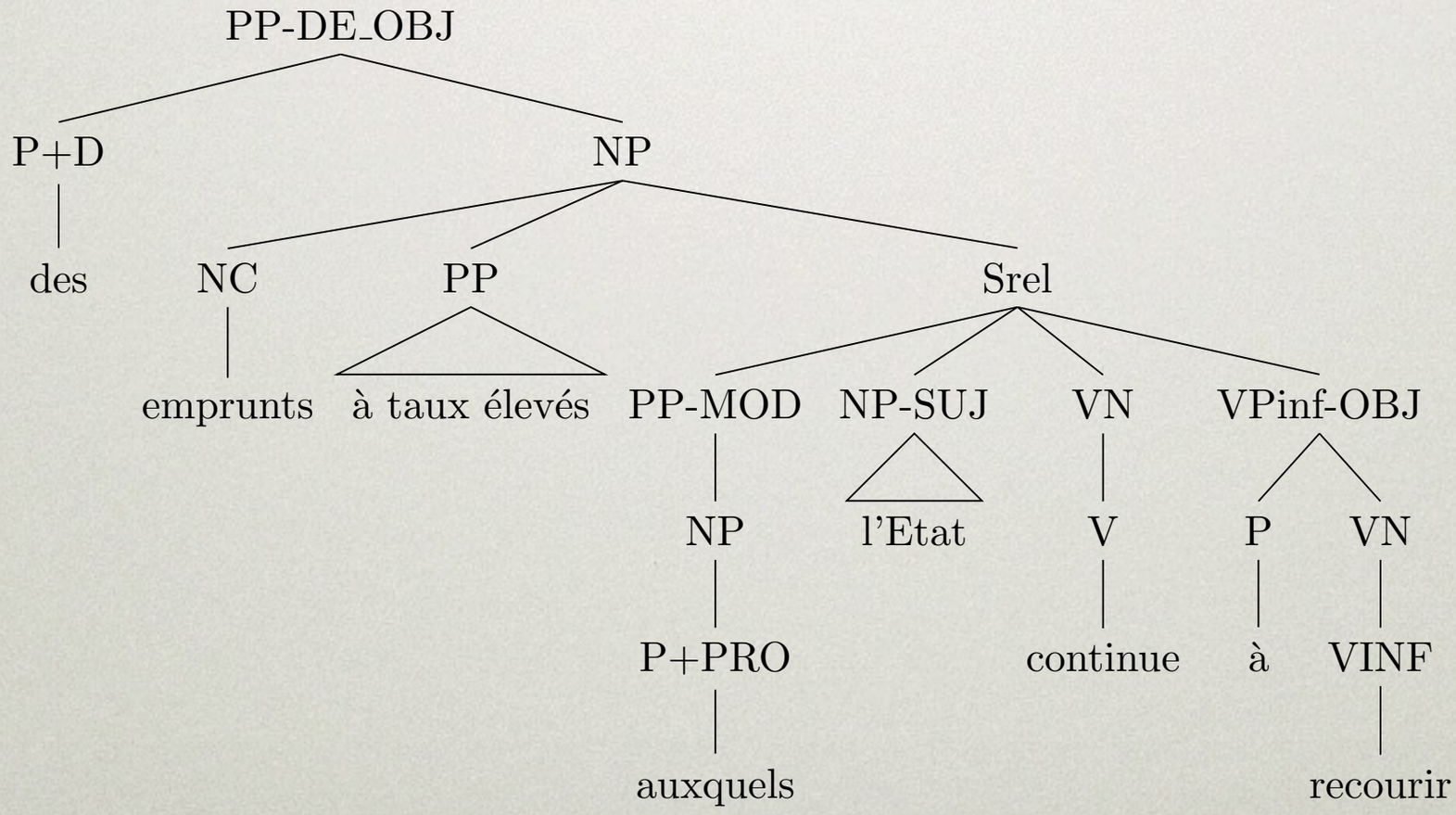
$$\dfrac{\dfrac{\text{de}}{(np\backslash s_{di})/(np\backslash s_i)}\,[Lex] \quad \dfrac{\dfrac{\text{s'}}{cl_r}\,[Lex] \quad \dfrac{\dfrac{\text{attaquer}}{(cl_r\backslash(np\backslash s_i))/pp_{\grave{a}}}\,[Lex] \quad \dfrac{}{\text{p}_0 \vdash pp_{\grave{a}}}\,[Hyp]_1}{\text{a} \circ \text{p}_0 \vdash cl_r\backslash(np\backslash s_i)}\,[/E]}{\text{s'} \circ (\text{a} \circ \text{p}_0) \vdash np\backslash s_i}\,[\backslash E]}{\text{de} \circ (\text{s'} \circ (\text{a} \circ \text{p}_0)) \vdash np\backslash s_i}\,[/E]$$

Applications

$$\begin{array}{|l|}\hline e_1\ y_1 \\ \hline y_1 : \begin{array}{|l|}\hline e_2\ e_3\ x_3 \\ \hline x_3 = \textbf{?} \\ \textbf{aider\_à}(e_2, x_0, x_3, e_3) \\ \textbf{partir}(e_3, x_3) \\ \hline \end{array} \\ \textbf{demander}(e_1, y_0, x_0, y_1) \\ \hline \end{array}$$

The French Treebank

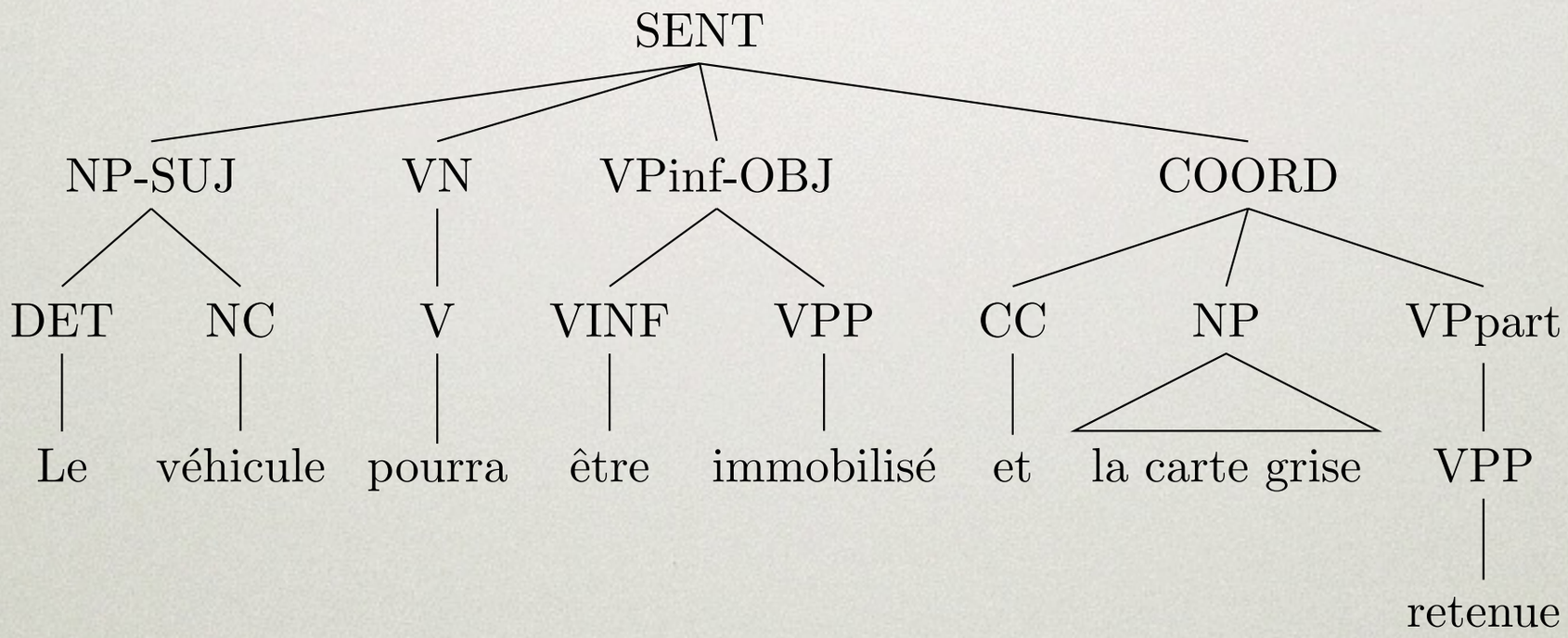
Example PP from the corpus (slightly simplified)



≈ continue_to_resort_to(state,loans)

The French Treebank

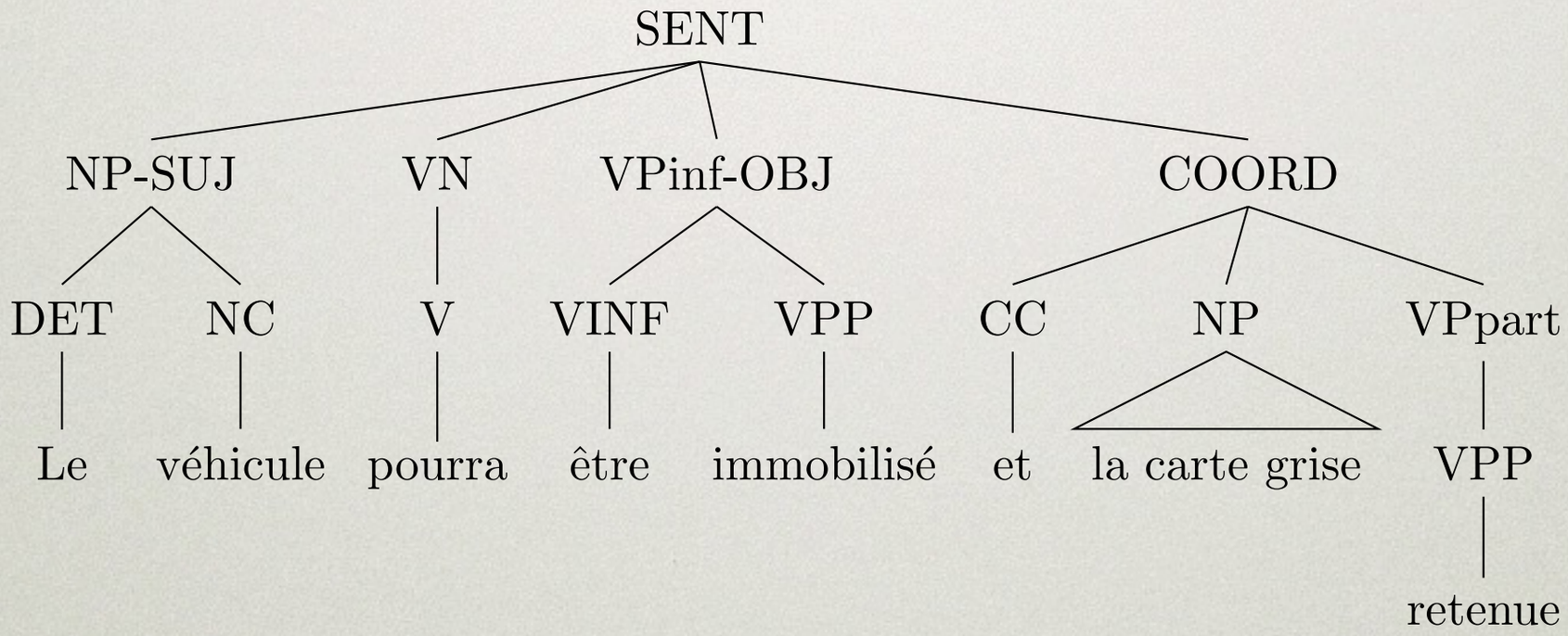
Example from the corpus (slightly simplified)



"The car could be immobilized and
the car registration (could be) taken"

The French Treebank

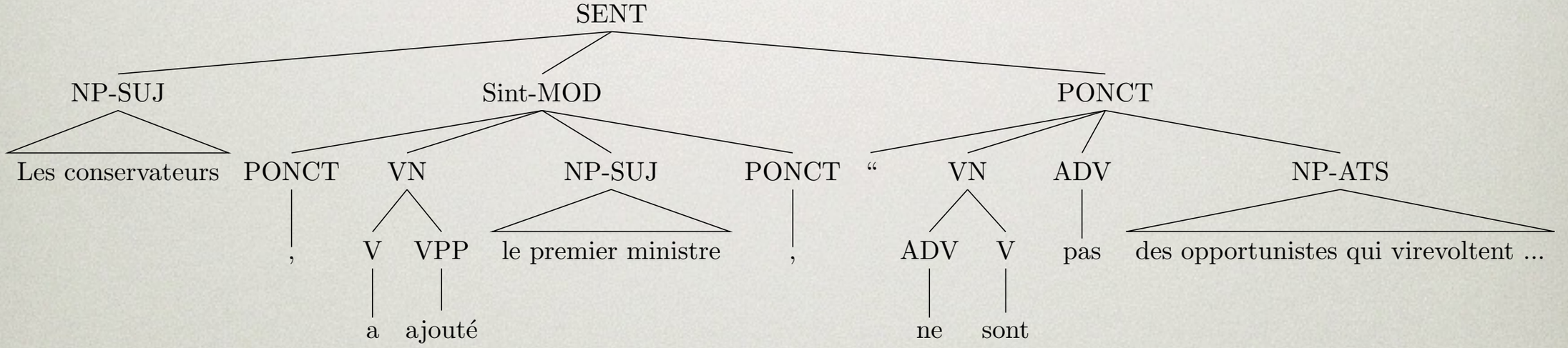
Example from the corpus (slightly simplified)



$\approx \exists x.\text{pouvoir}(\text{immobiliser}(x,\text{véhicule})) \wedge$
$\exists y.\text{pouvoir}(\text{retenir}(y,\text{cart\_grise}))$

The French Treebank

Example from the corpus (slightly simplified)



The conservative party, added the prime minister, “are not opportunists who flip-flop...”

FTB: The Good, The Bad and The Ugly

- Context-free grammar
- Verb group as a constituent
- Treatment of coordination

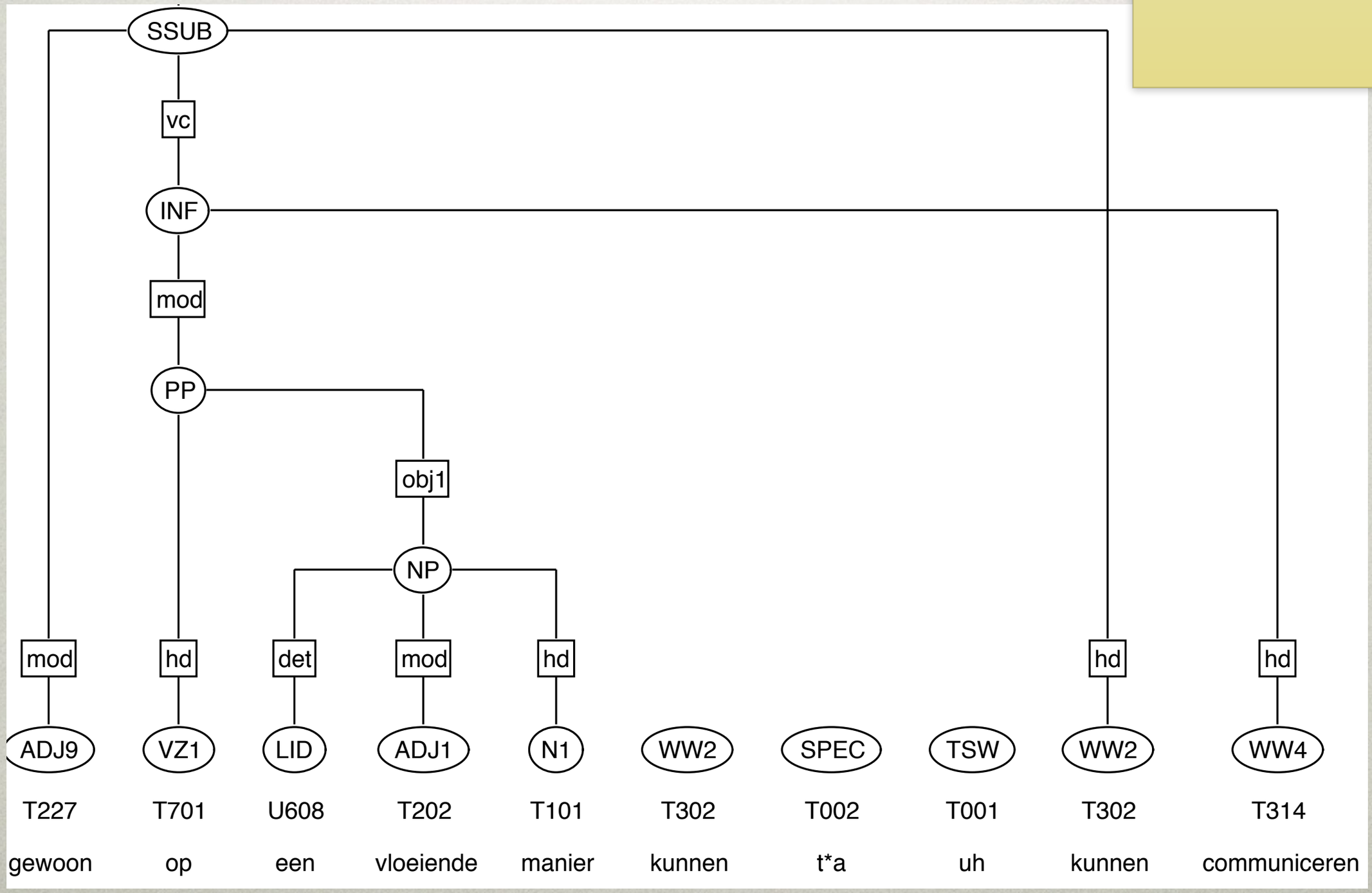
CGN and Lassy

- Spoken Dutch Corpus and Large-Scale Syntactic Annotation of Written Dutch.
- Both around 1 million words with similar annotation schemes.

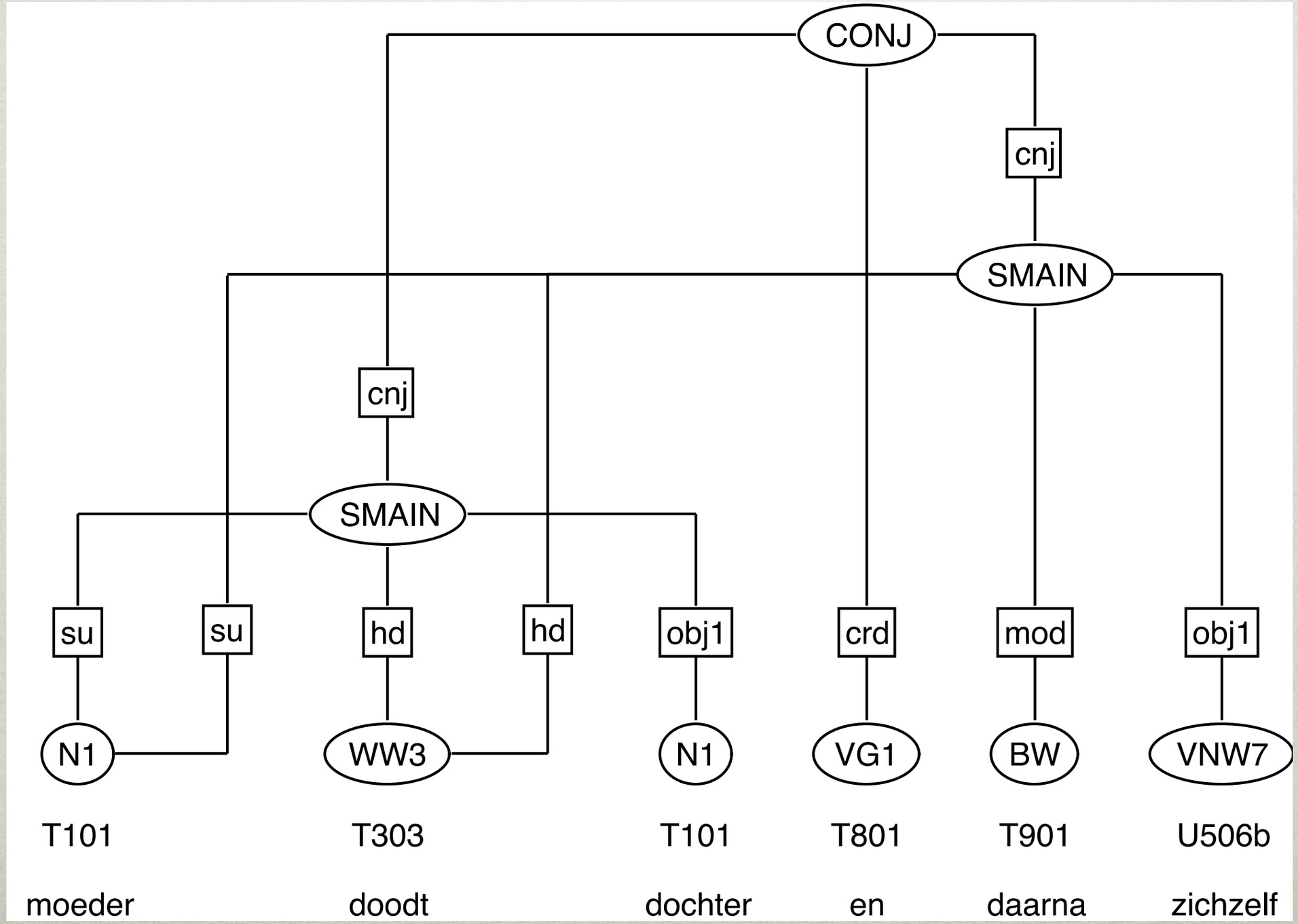
CGN and Lassy

disfluences: repeated “kunnen”, broken off word “t” and hesitation “uh”,

crossed dependency



CGN and Lassy



CGN and Lassy: The Good, The Bad, The Ugly

+ Long distance dependencies have been annotated

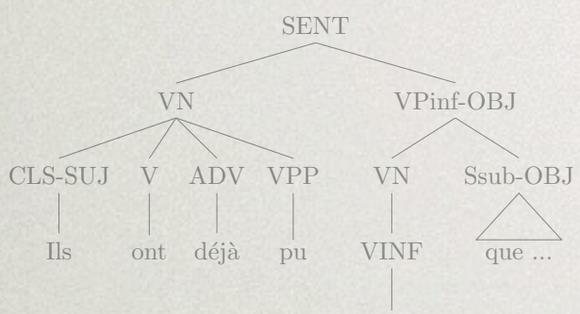
 - also used for coordination and auxiliaries

 - position of traces not indicated

+ Discontinuous dependencies

Outline

French Treebank



Grammar

Extraction

$$\dfrac{\text{de}}{(np\backslash s_{di})/(np\backslash s_i)}\,[Lex] \quad \dfrac{\dfrac{\text{s'}}{cl_r}\,[Lex] \quad \dfrac{\dfrac{\text{attaquer}}{(cl_r\backslash(np\backslash s_i))/pp_{\grave{a}}}\,[Lex] \quad \dfrac{}{\text{p}_0 \vdash pp_{\grave{a}}}\,[Hyp]_1}{\text{a} \circ \text{p}_0 \vdash cl_r\backslash(np\backslash s_i)}\,[/E]}{\text{s'} \circ (\text{a} \circ \text{p}_0) \vdash np\backslash s_i}\,[\backslash E]}{\text{de} \circ (\text{s'} \circ (\text{a} \circ \text{p}_0)) \vdash np\backslash s_i}\,[/E]$$

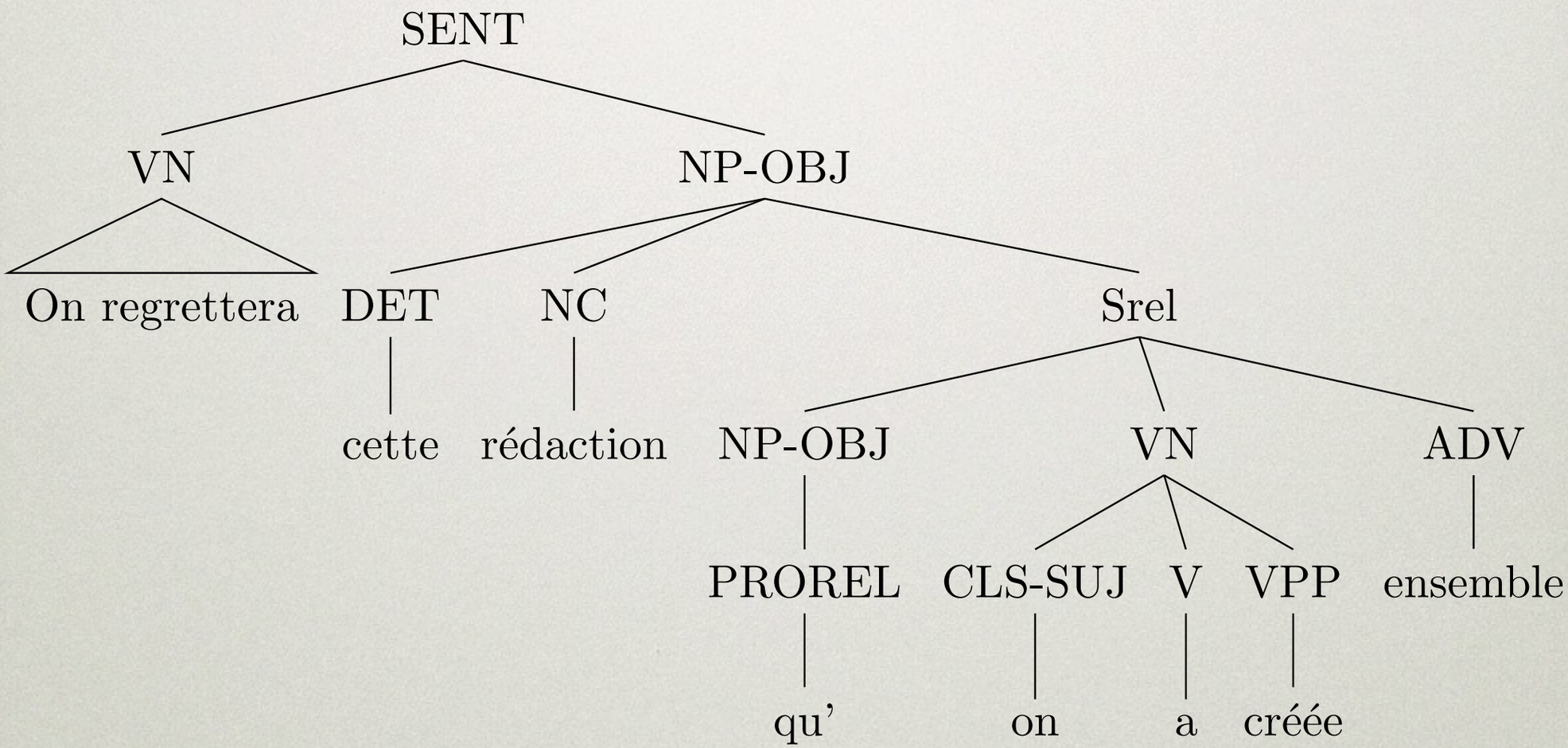
Applications

$$\begin{array}{|l|}\hline e_1\ y_1 \\ \hline y_1 : \begin{array}{|l|}\hline e_2\ e_3\ x_3 \\ \hline x_3 = \textbf{?} \\ \textbf{aider\_à}(e_2, x_0, x_3, e_3) \\ \textbf{partir}(e_3, x_3) \\ \hline \end{array} \\ \textbf{demander}(e_1, y_0, x_0, y_1) \\ \hline \end{array}$$

Why AB grammars Are Not Enough

- getting the semantics right requires a somewhat richer system than AB grammars
  - introduction rules (“traces” or the original “slash categories” and their semantics)
  - structural rules (“movement” or “head wrap”, essentially restricted tree rewrite operations)

Introduction Rules: Example



Introduction Rules: Example

redaction	qu’	on	a	créée
n	(n\n)/(s/np)	np	(np\s)/(np\s_{ppart})	(np\s_{ppart})/np

Introduction Rules: Example

redaction	qu’	on	a	créée	
n	(n\n)/(s/np)	np	(np\s)/(np\s_{ppart})	(np\s_{ppart})/np	np

Introduction Rules: Example

redaction qu' on a créée

n (n\n)/(s/np) np (np\s)/(np\$s_{ppart}$) $\dfrac{(np\backslash s_{ppart})/np \qquad np}{np\backslash s_{ppart}}$ [/E]

Introduction Rules: Example

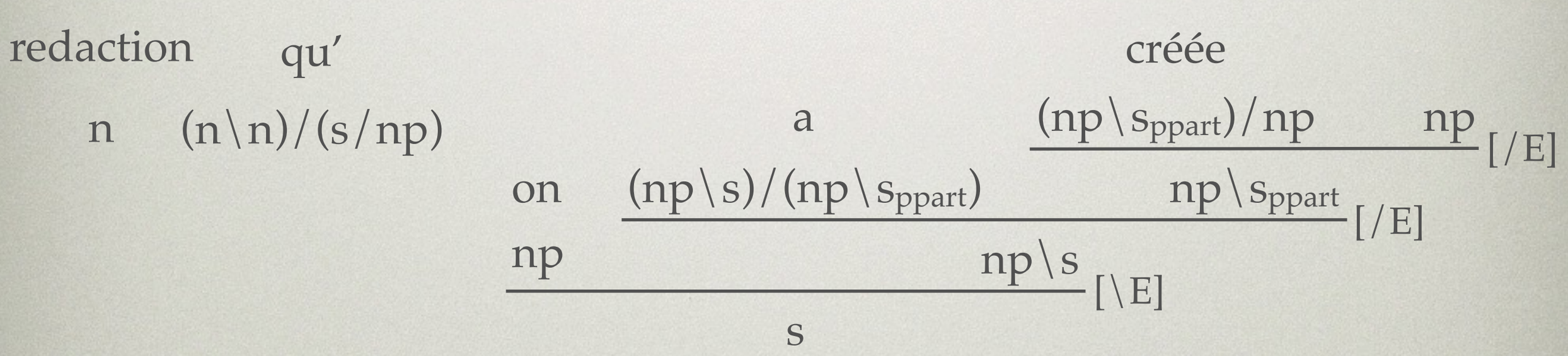
redaction qu' on créée

n (n\n)/(s/np) np a (np\s$_{ppart}$)/np np [/E]

(np\s)/(np\s$_{ppart}$) np\s$_{ppart}$ [/E]

np\s

Introduction Rules: Example

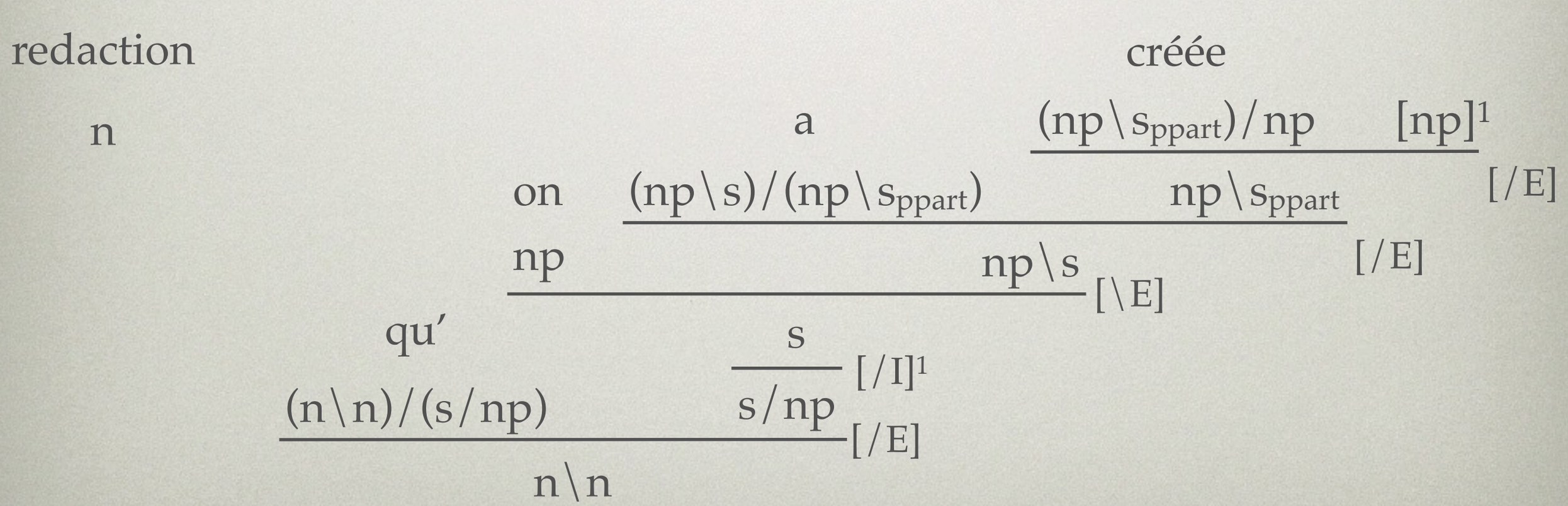


Introduction Rules: Example

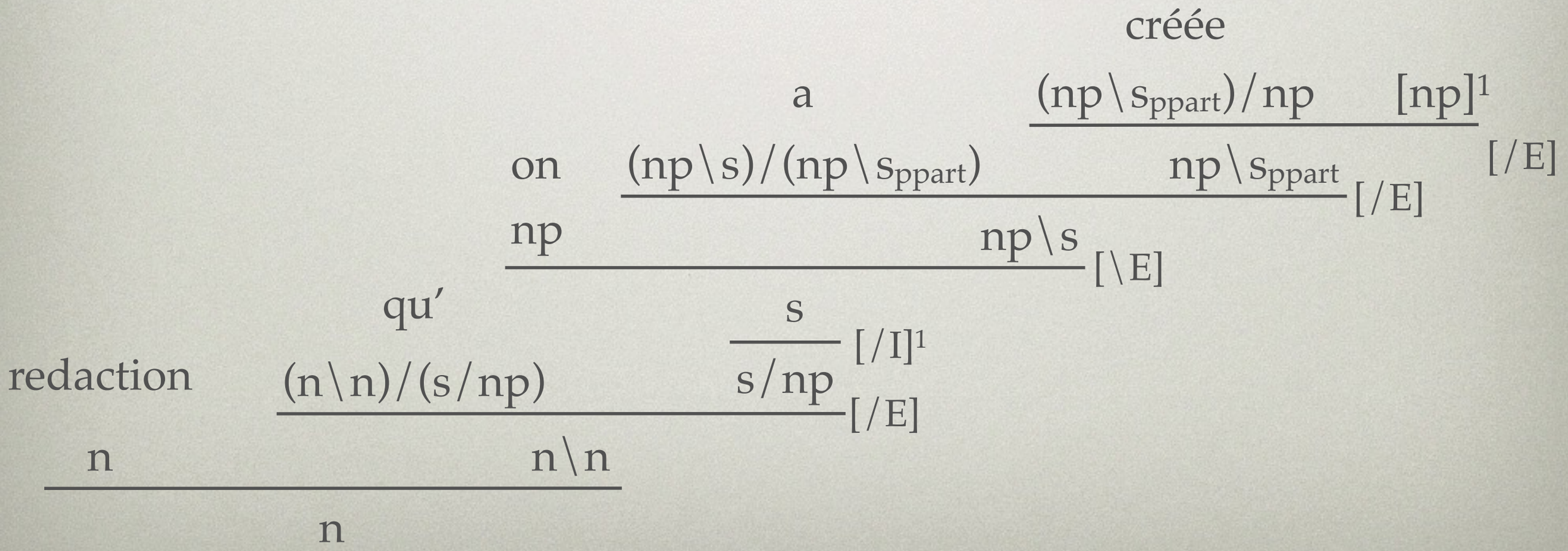
redaction qu' on a créée

n (n\n)/(s/np) np (np\s)/(np\s$_{ppart}$) [np]^1 (np\s$_{ppart}$)/np [/E] np\s$_{ppart}$ [/E] np\s [\E] s [/I]^1 s/np

Introduction Rules: Example



Introduction Rules: Example



Beyond AB grammarS
Introduction Rules

qu’	on	a	créée	ensemble
(n\n)/(s/np)	np	(np\s)/(np\s_{ppart})	(np\s_{ppart})/np	(np\s)\(np\s)

Beyond AB grammarS
Introduction Rules

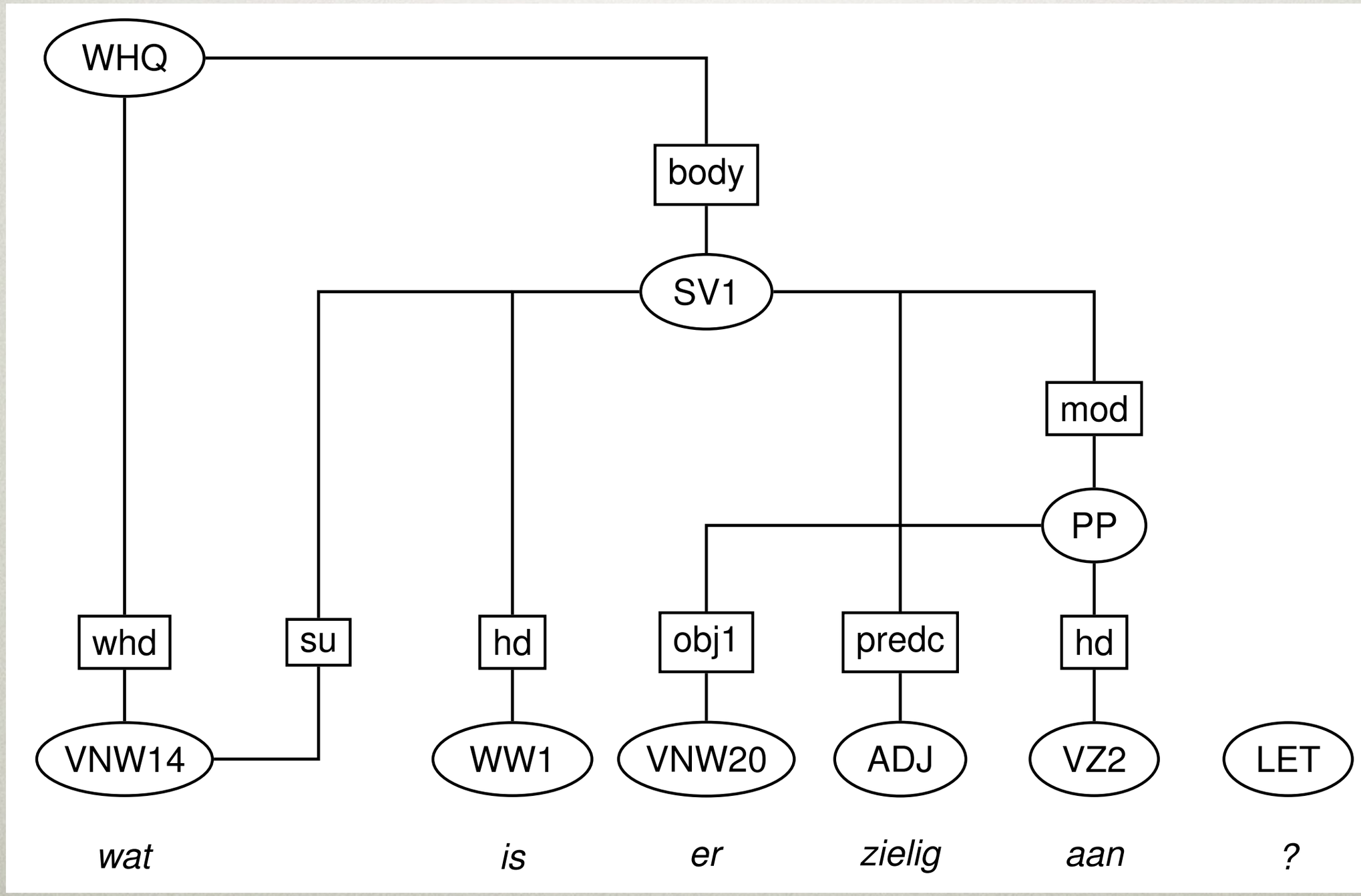
$$\frac{\ldots [B]^{i} \ldots}{A / \Diamond\Box B}\,[/I]^{i}$$

qu’	on	a	créée	ensemble
$(n\backslash n)/(s\Diamond\Box np)$	np	$(np\backslash s)/(np\backslash s_{ppart})$	$(np\backslash s_{ppart})/np$	$(np\backslash s)\backslash(np\backslash s)$

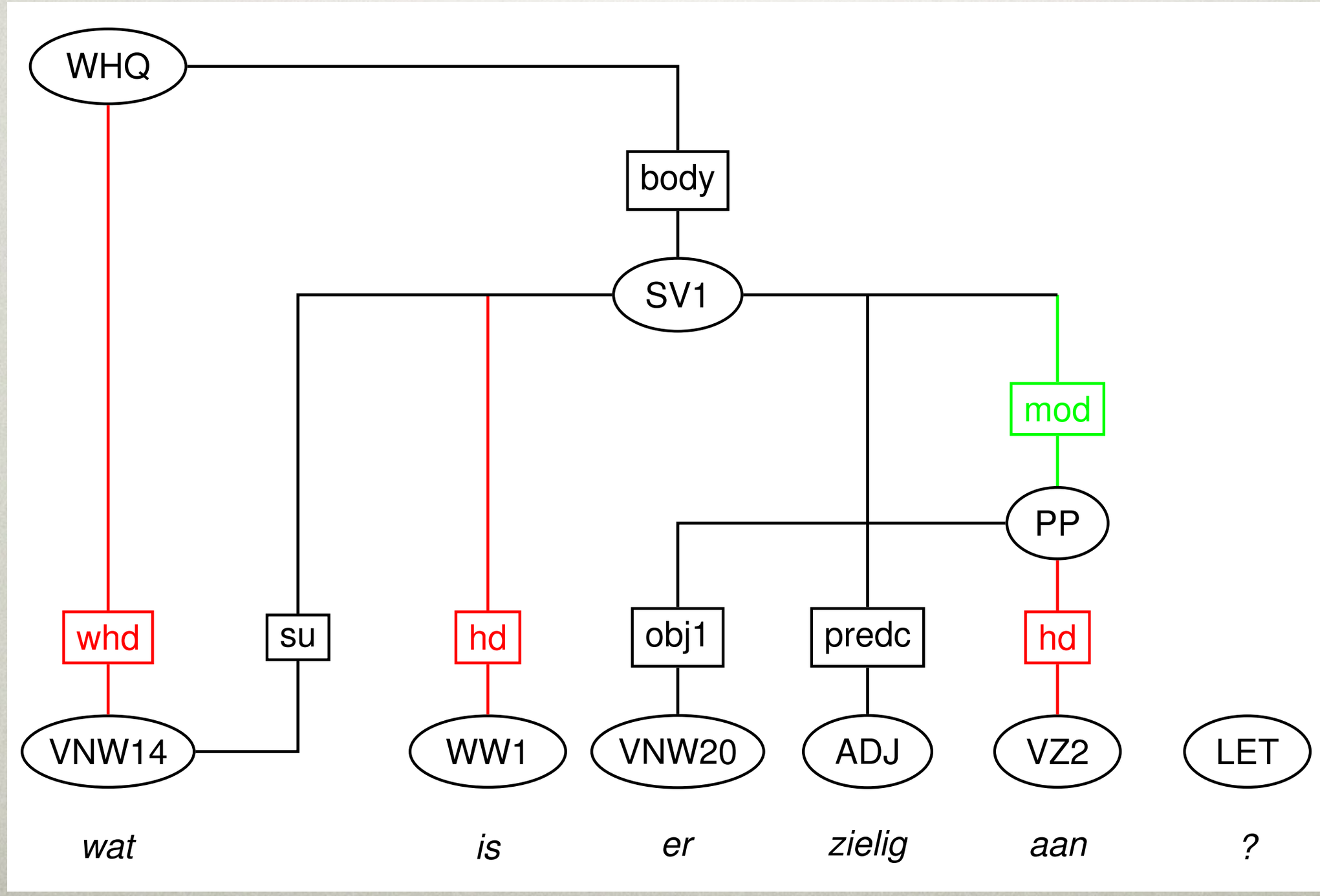
Treebank Extraction for French and Dutch

- A rather classical approach. The extraction algorithm is parametric wrt three functions:
  1. a function identifying heads/functors
  2. a function identifying modifiers
  3. a function given a formula for each syntactic category of the annotation

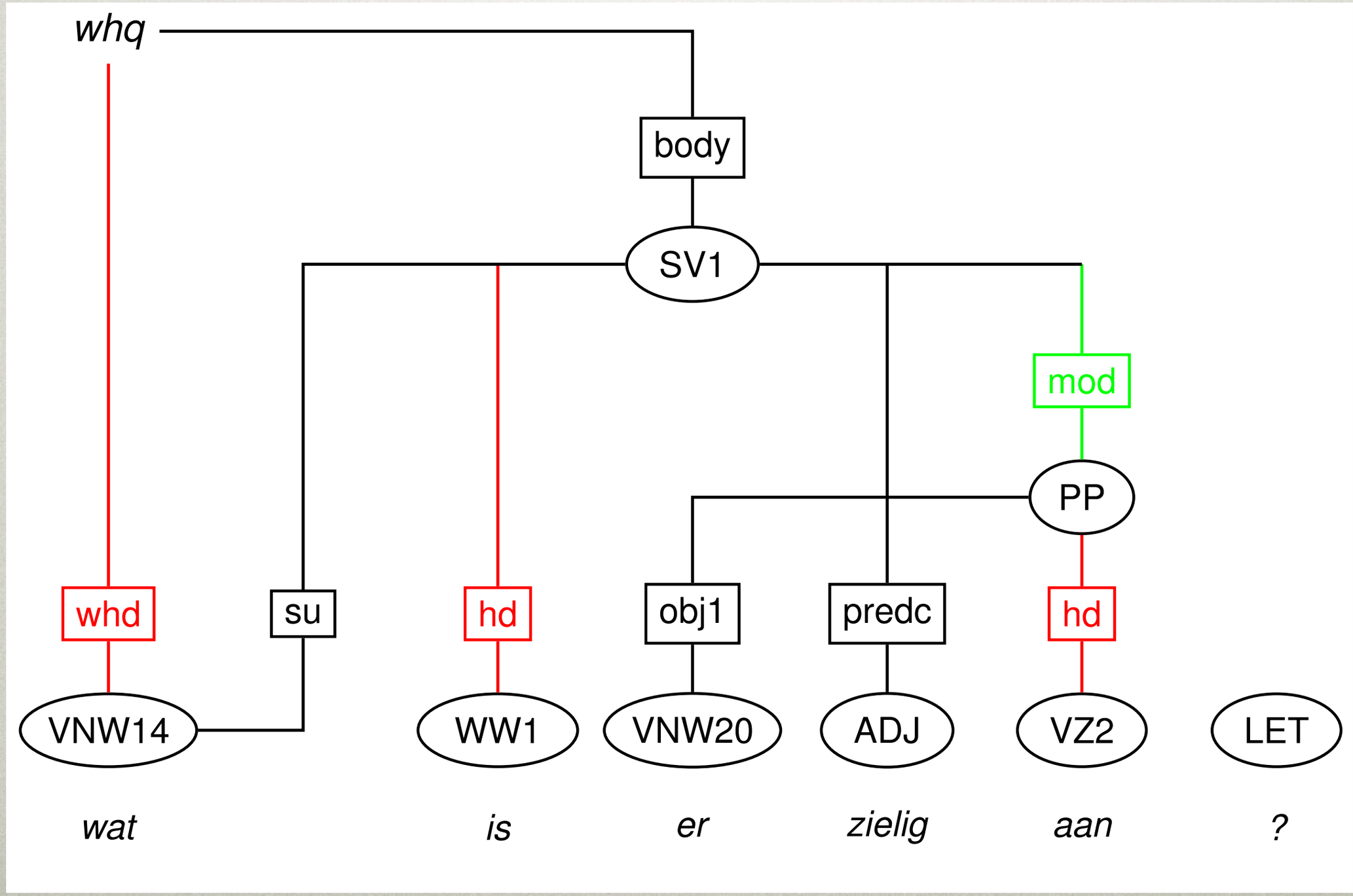
Treebank Extraction: Spoken Dutch Corpus



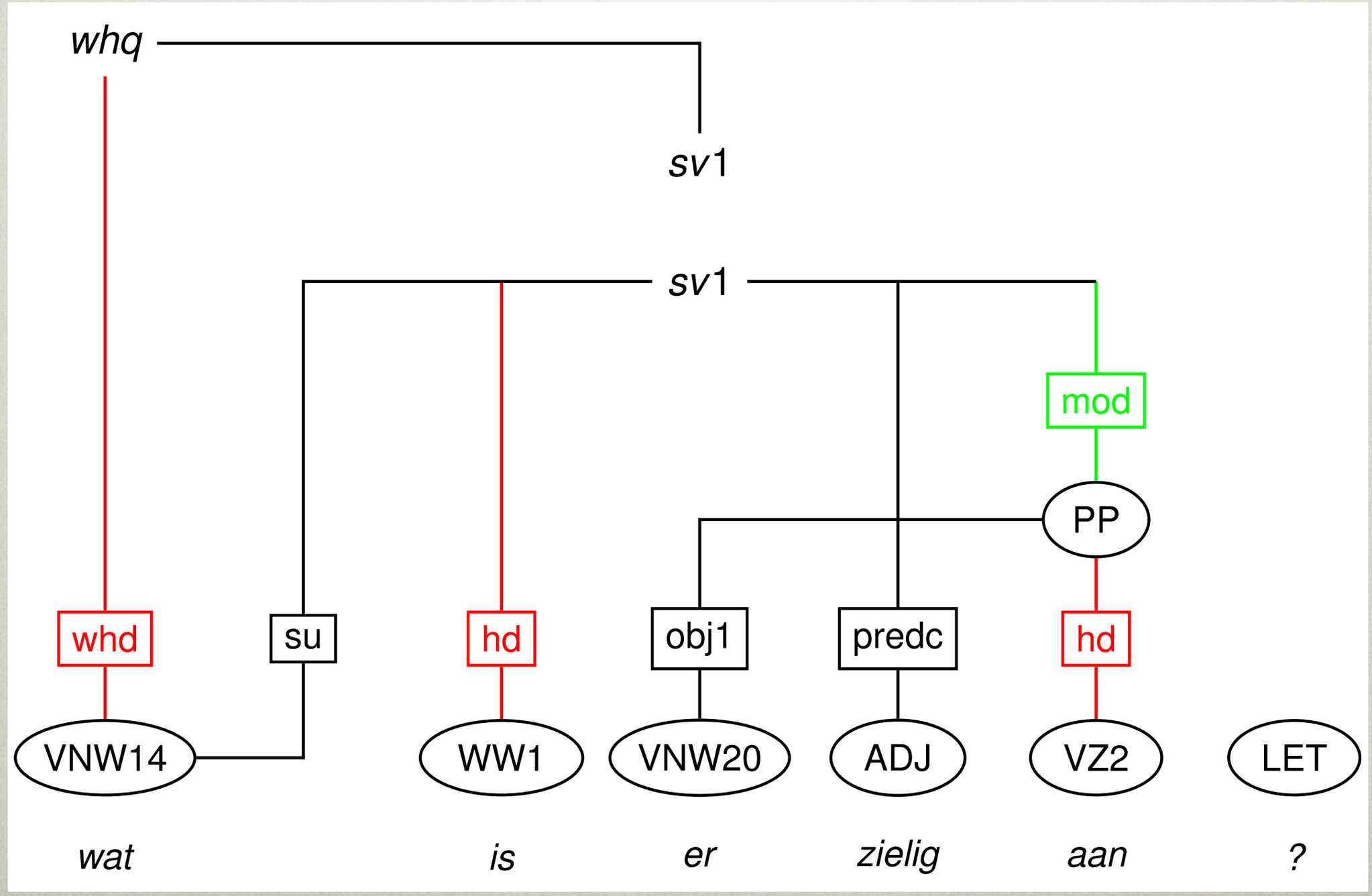
Treebank Extraction: Spoken Dutch Corpus



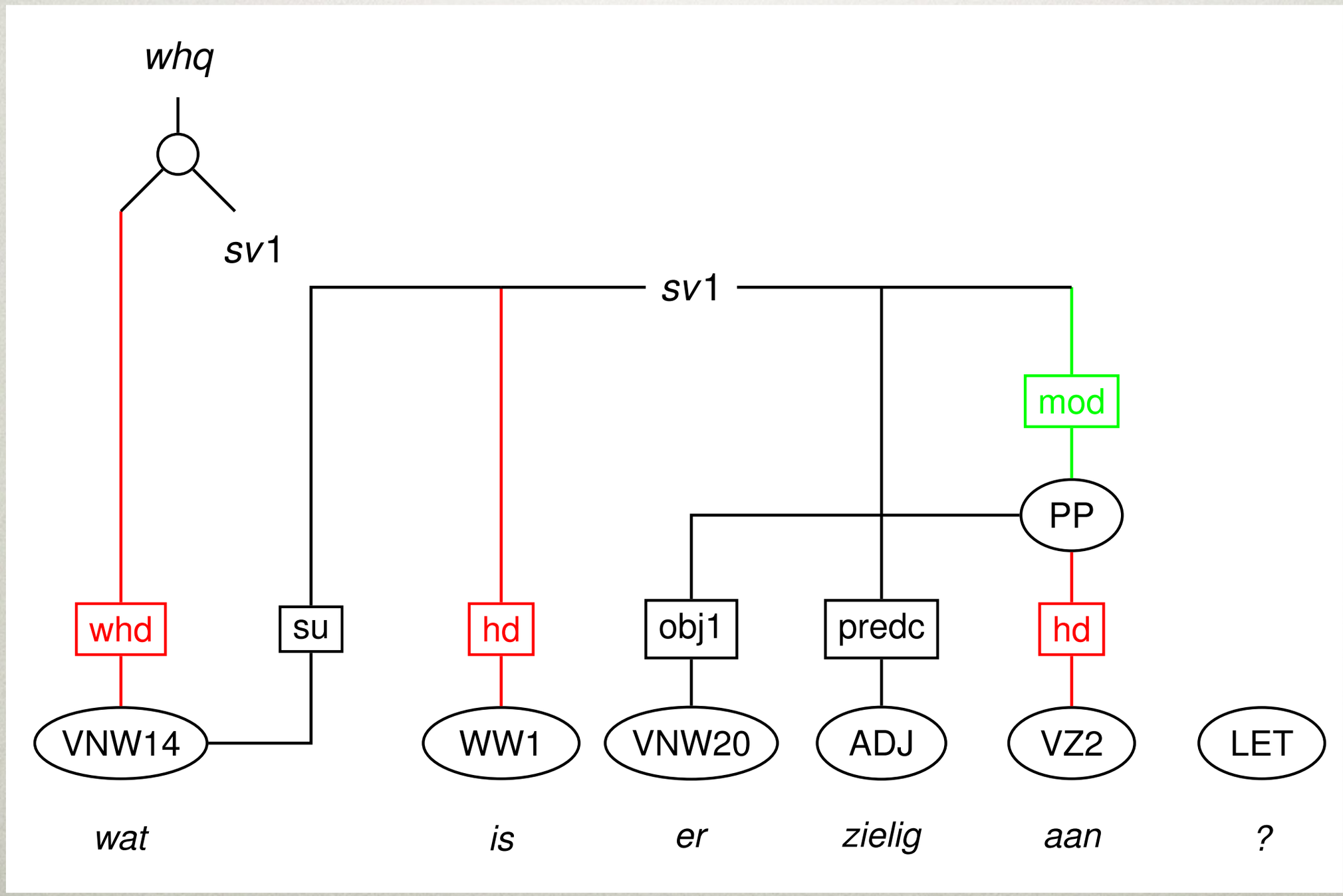
Treebank Extraction: Spoken Dutch Corpus



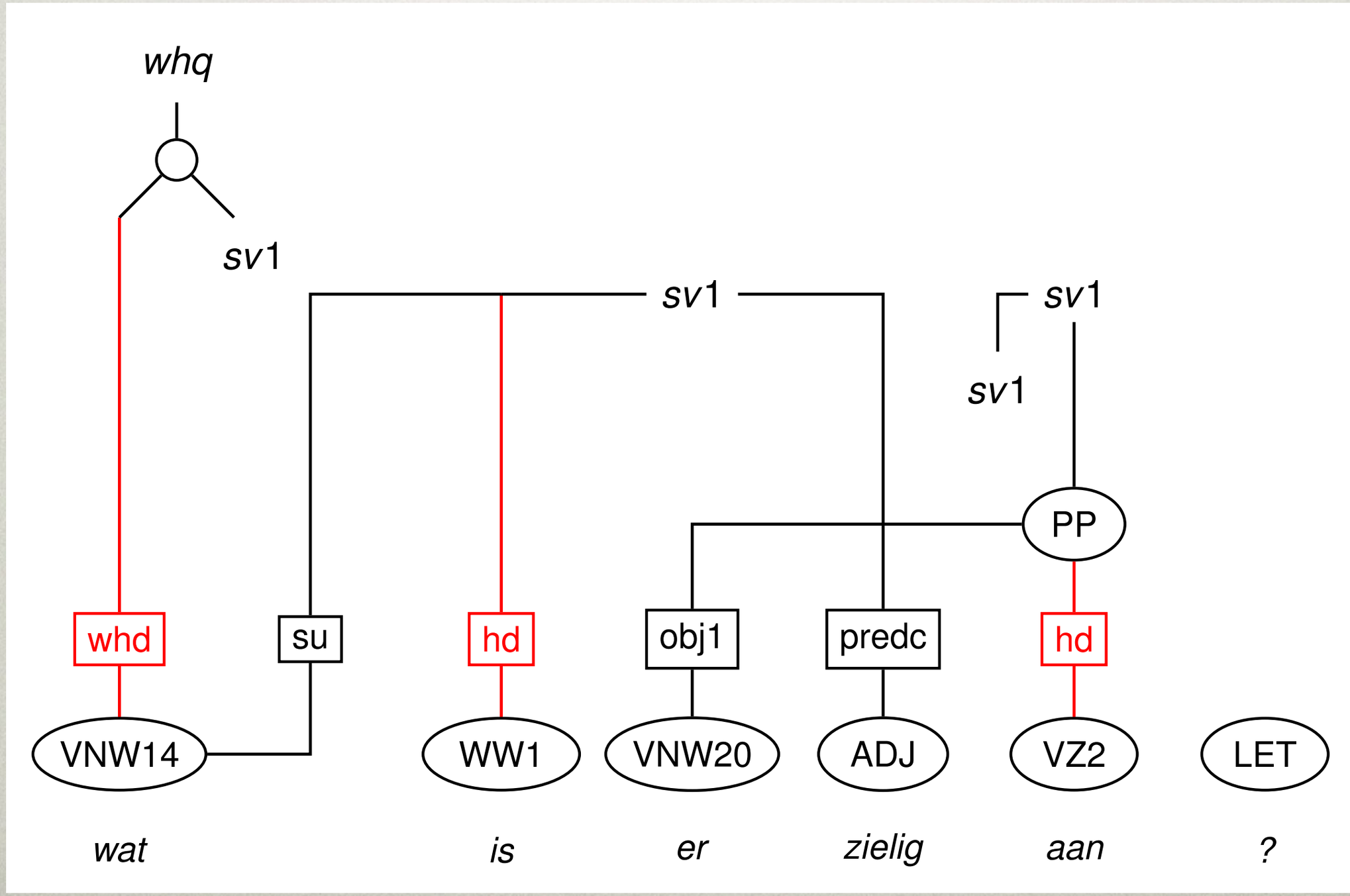
Treebank Extraction: Spoken Dutch Corpus



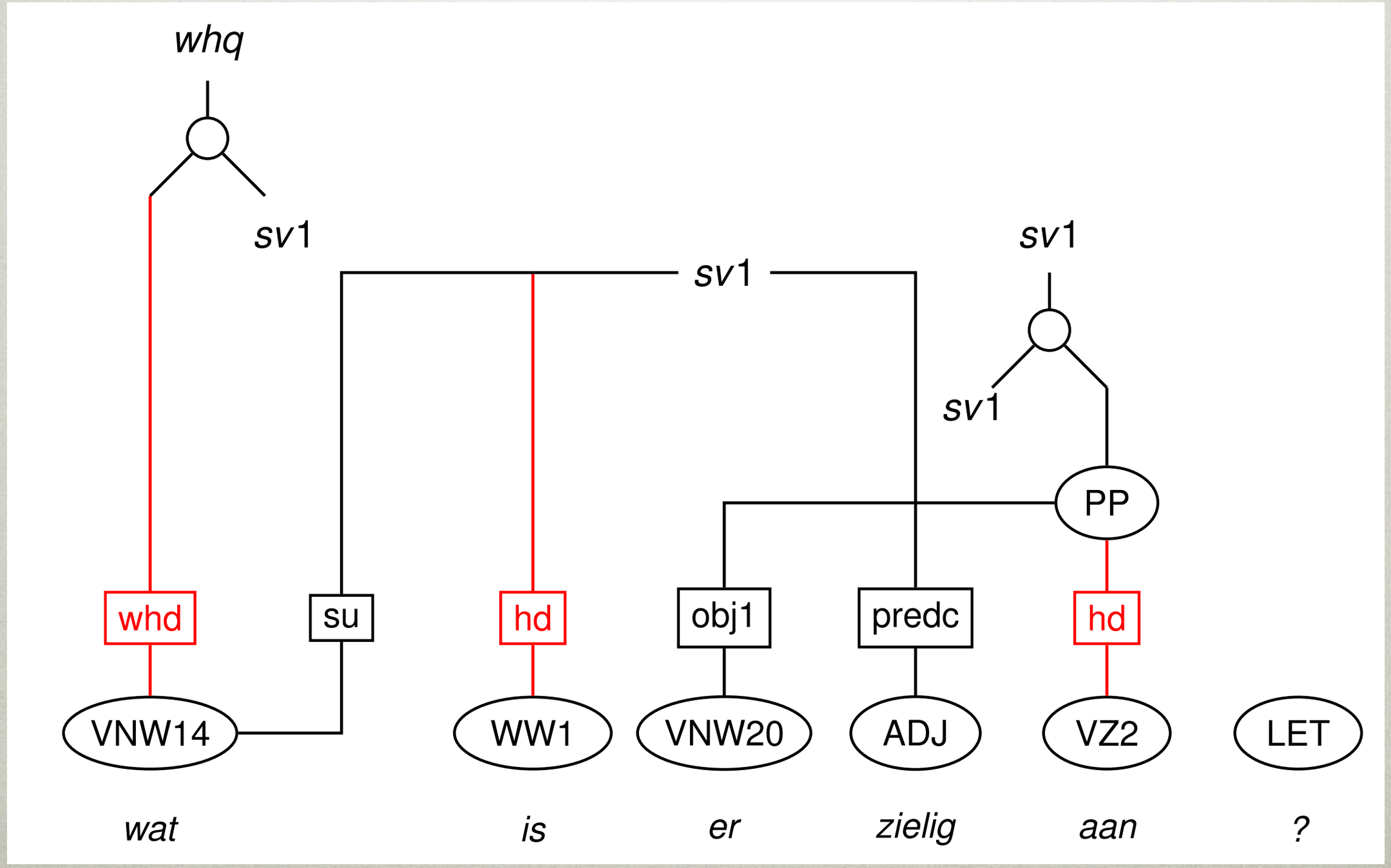
Treebank Extraction: Spoken Dutch Corpus



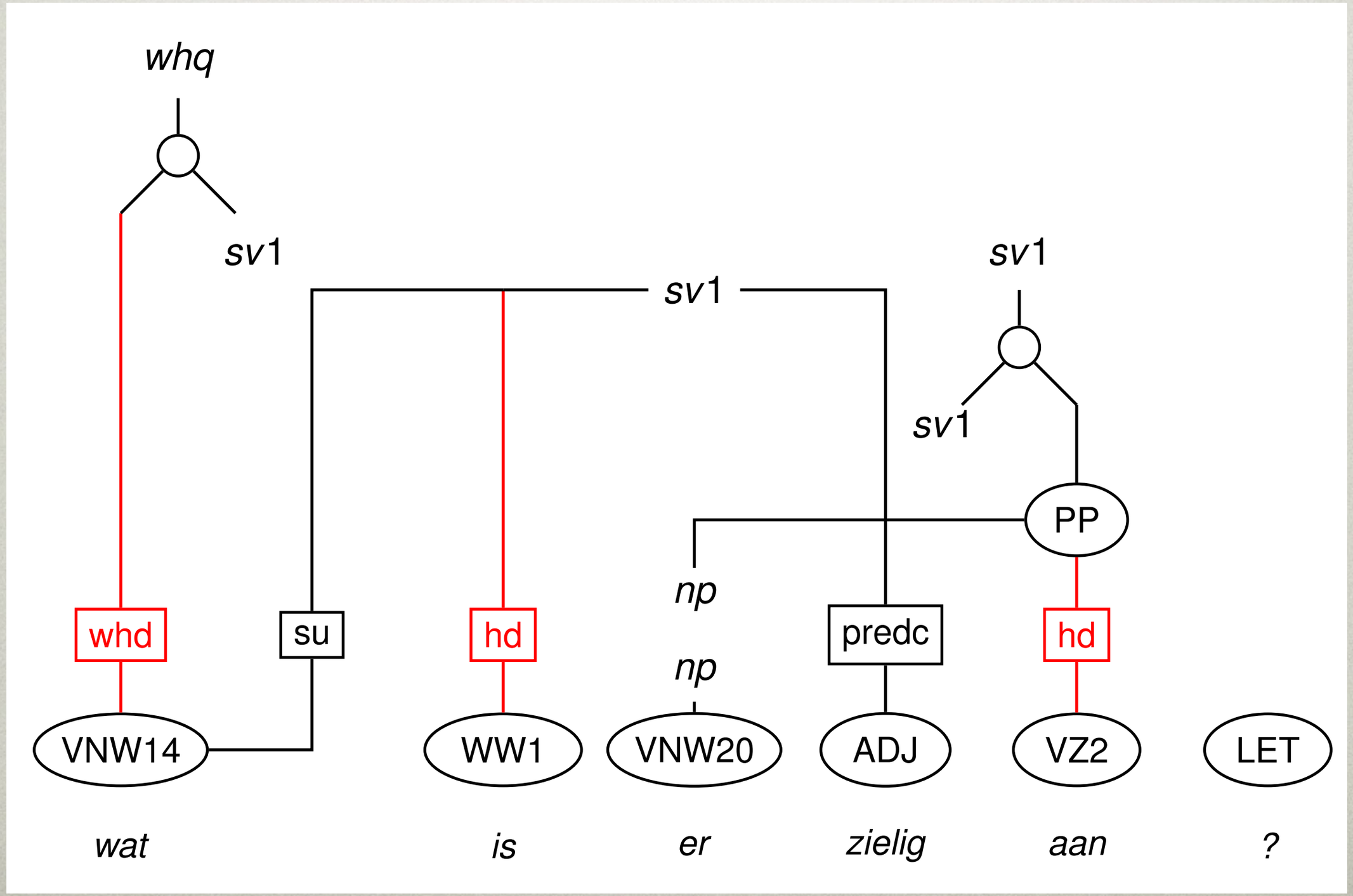
Treebank Extraction: Spoken Dutch Corpus



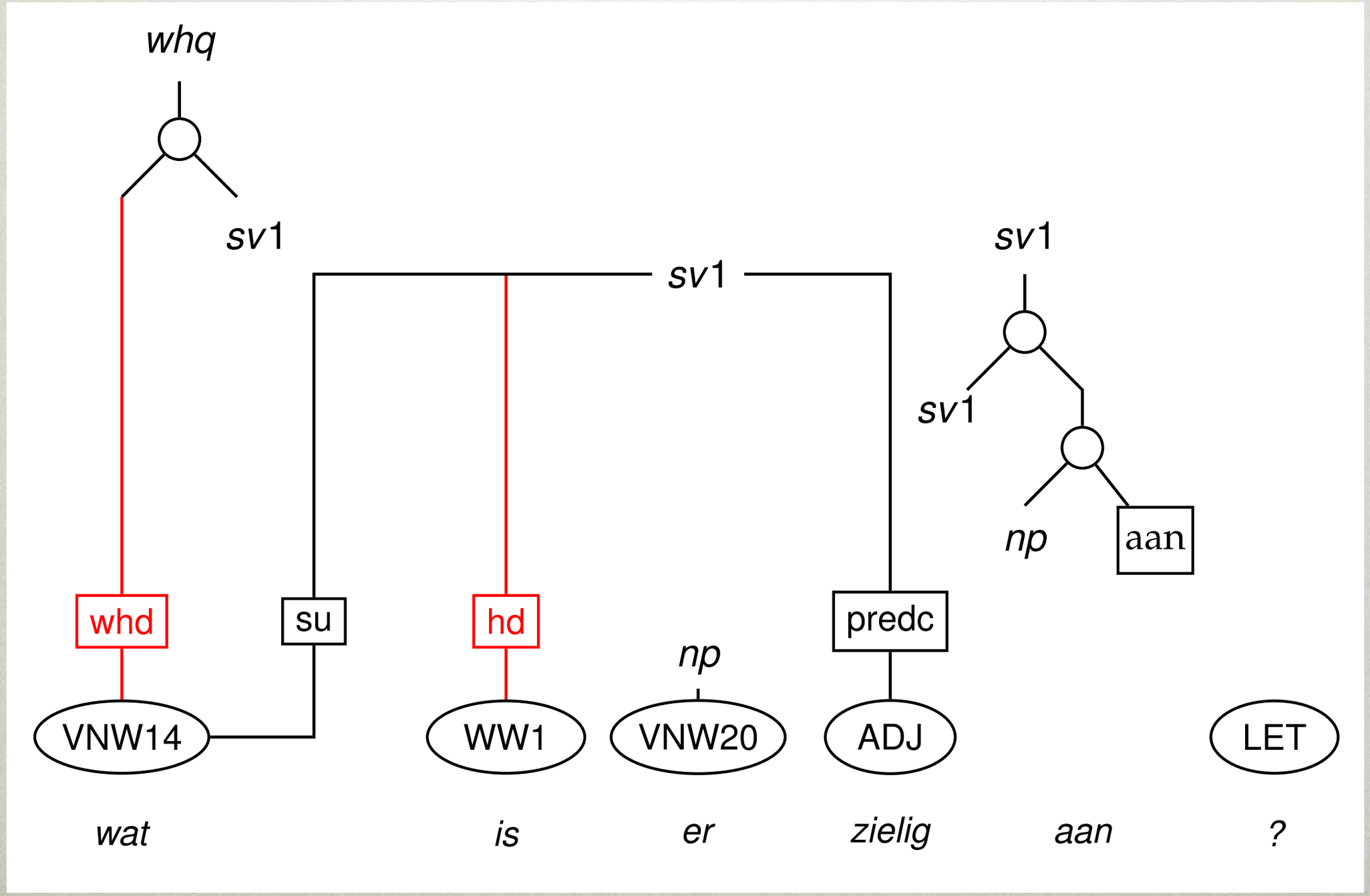
Treebank Extraction: Spoken Dutch Corpus



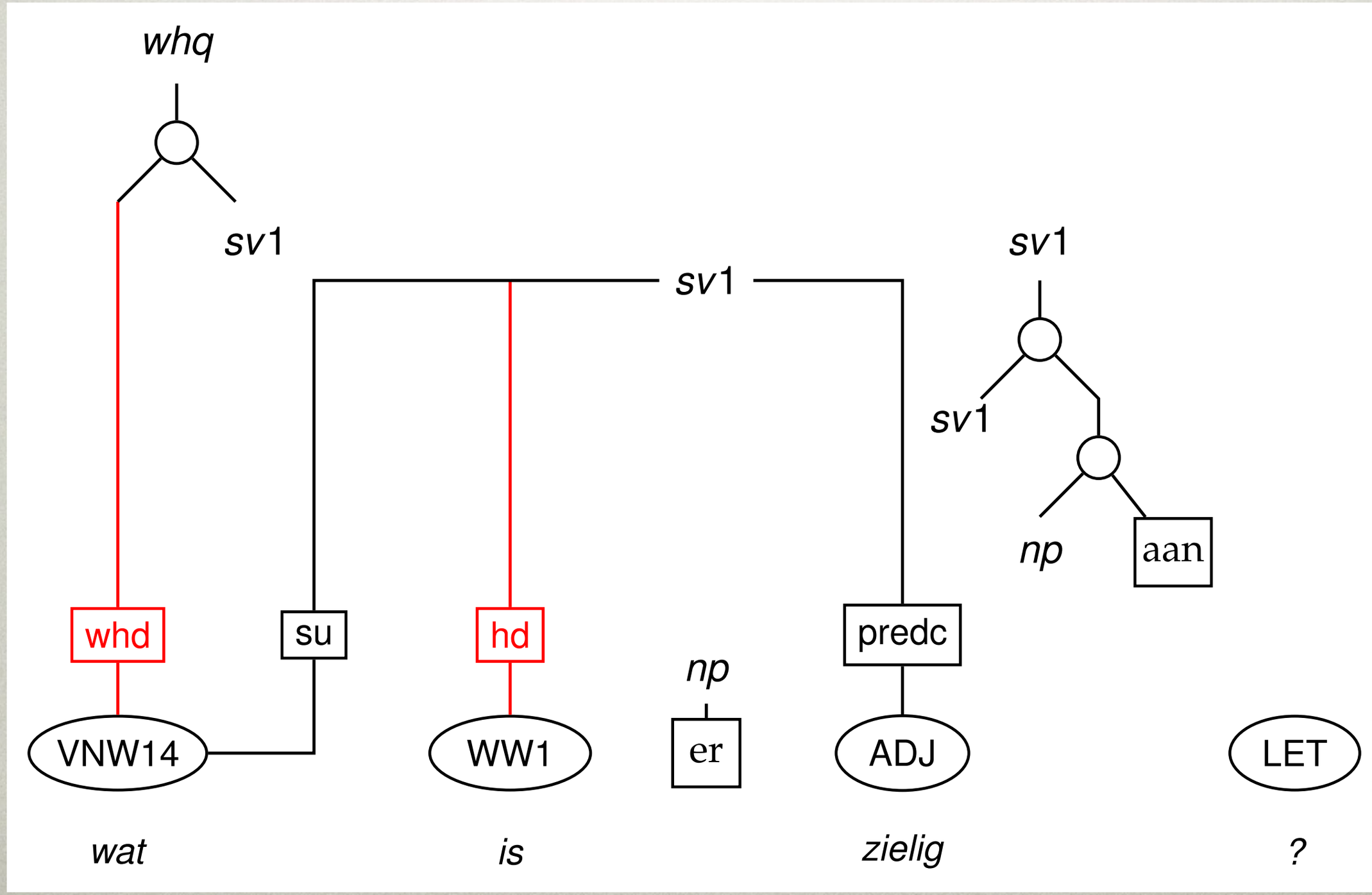
Treebank Extraction: Spoken Dutch Corpus



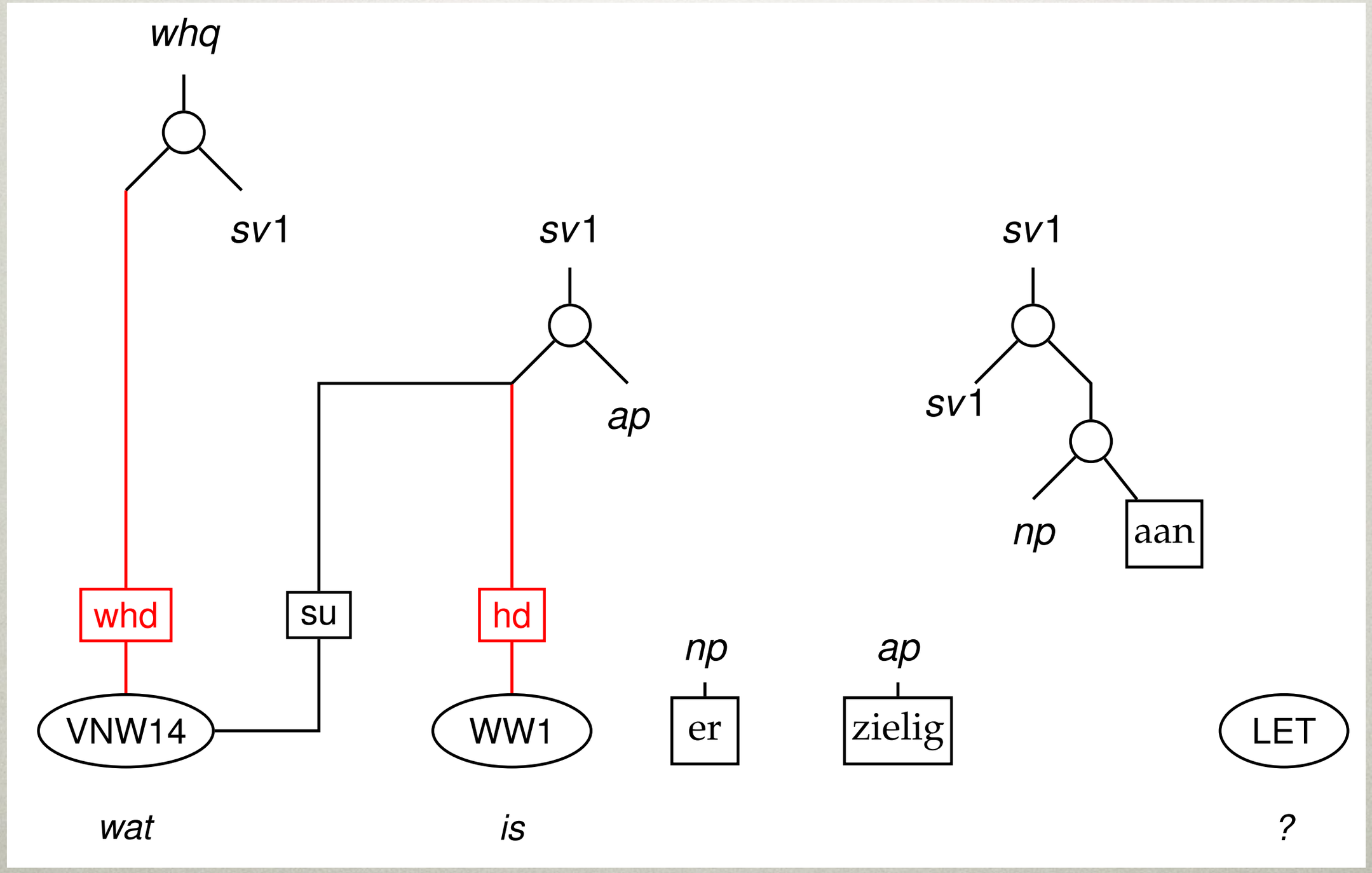
Treebank Extraction: Spoken Dutch Corpus



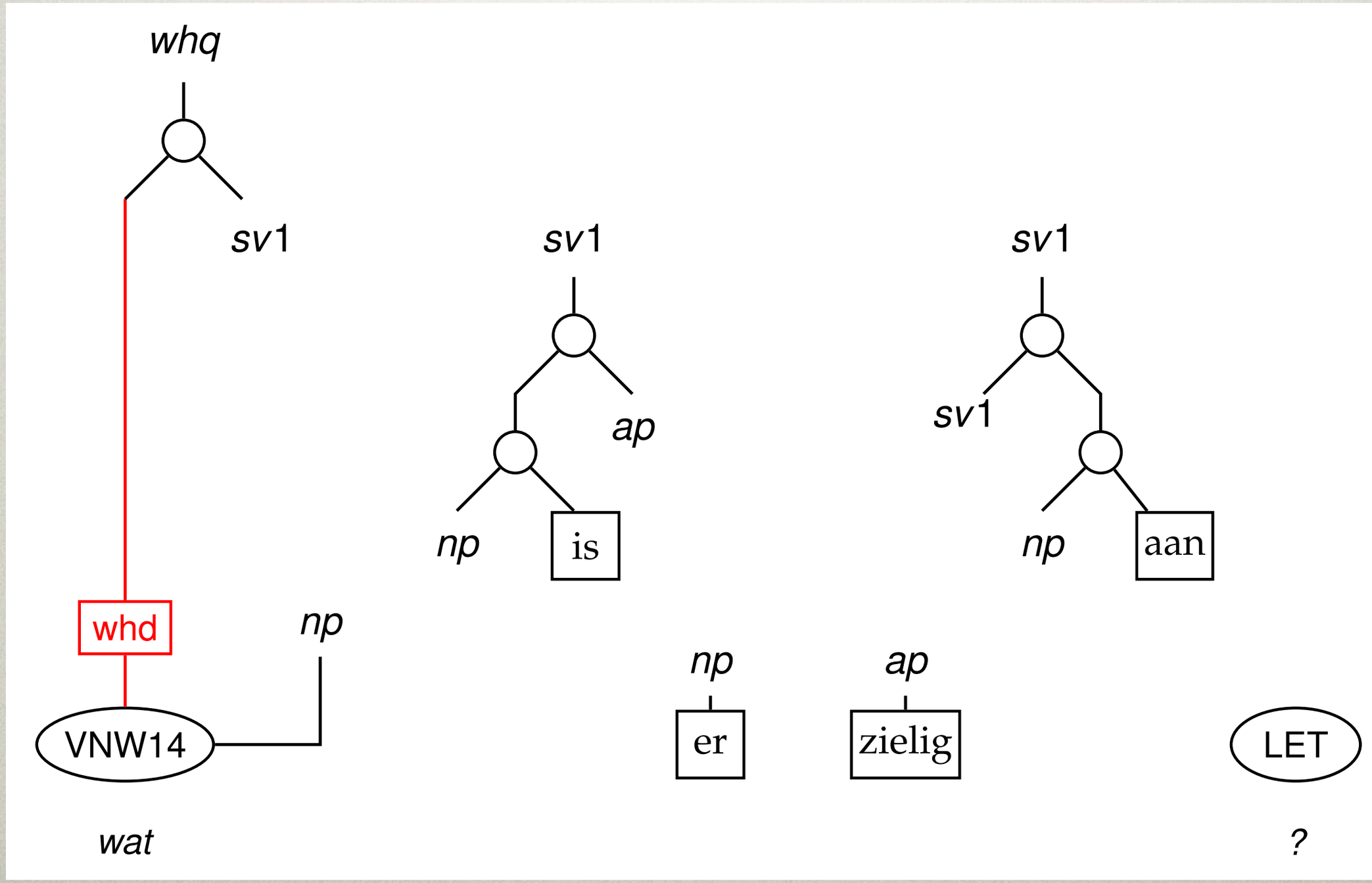
Treebank Extraction: Spoken Dutch Corpus



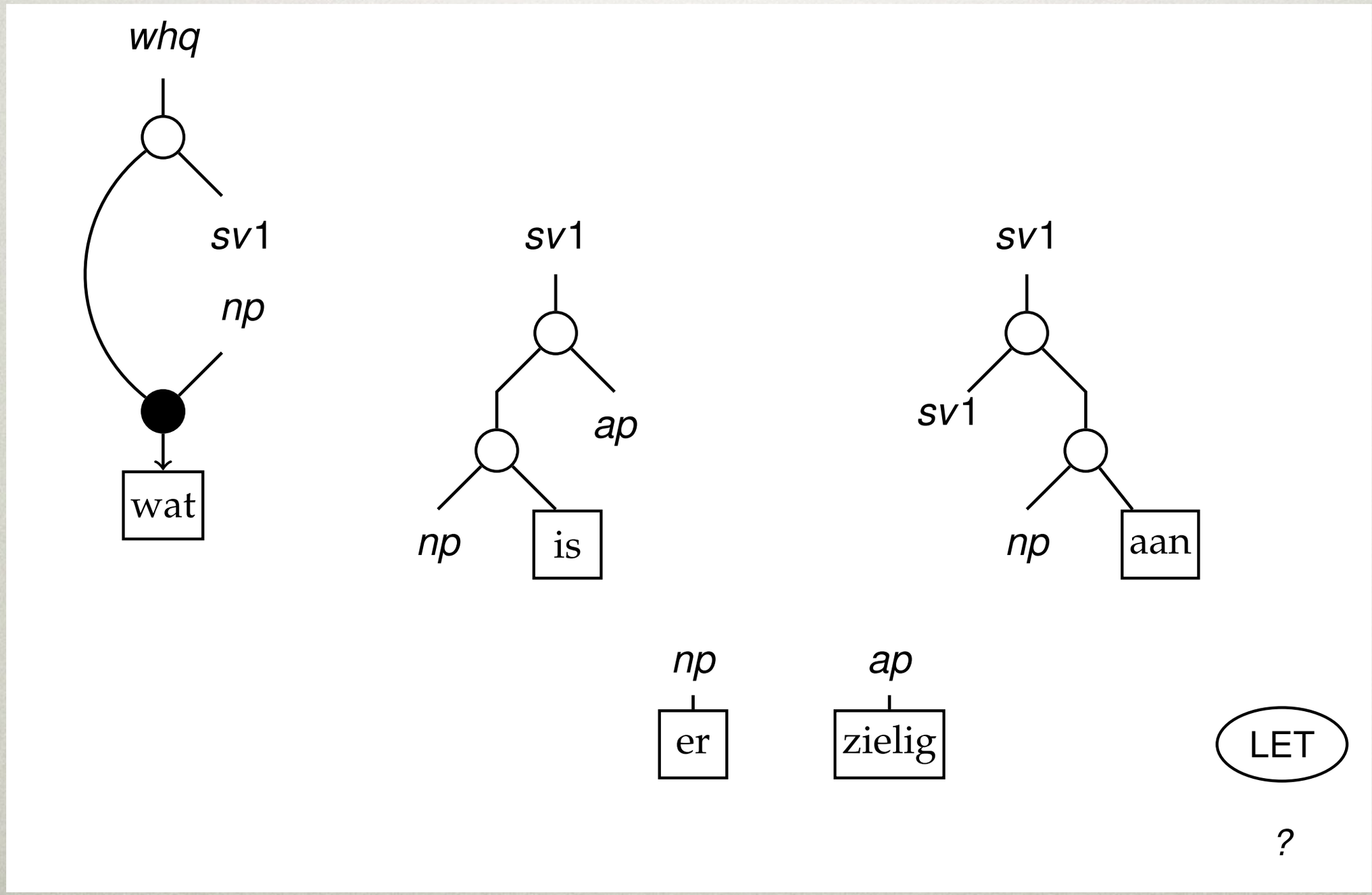
Treebank Extraction: Spoken Dutch Corpus



Treebank Extraction: Spoken Dutch Corpus

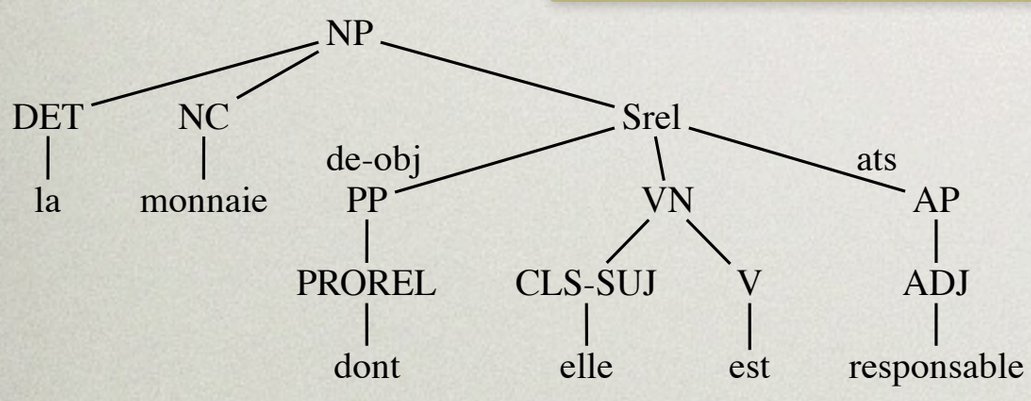


Treebank Extraction: Spoken Dutch Corpus

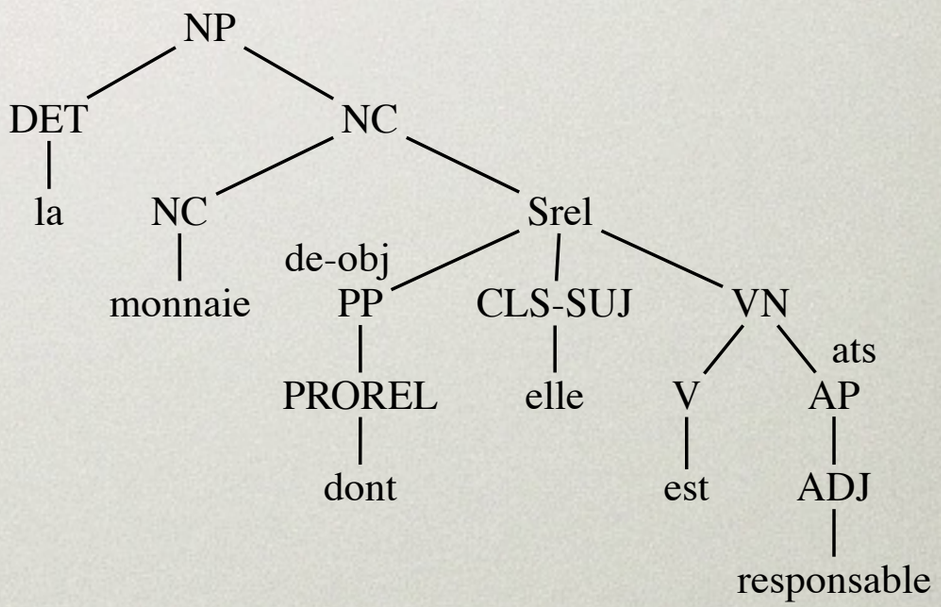


Treebank Extraction

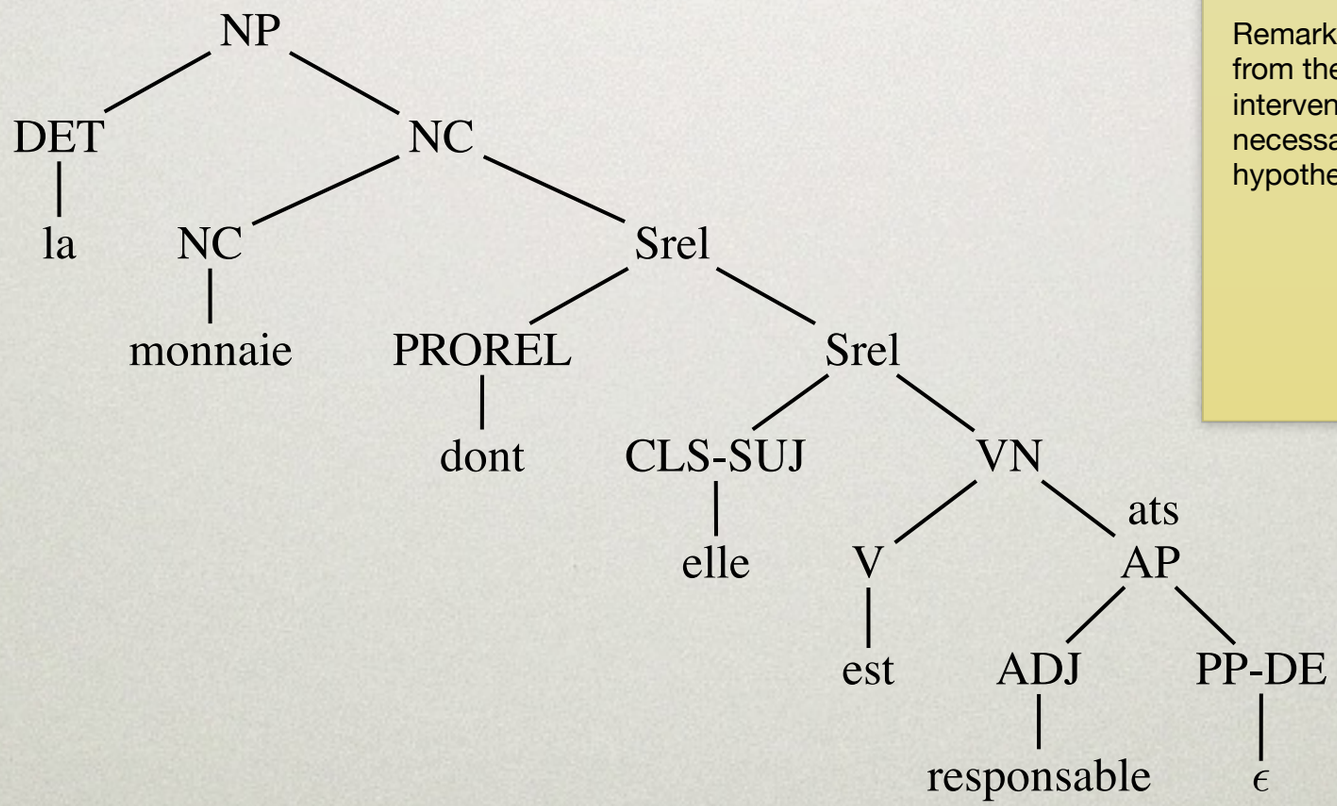
Sentence as we find it in the corpus. “dont” is a relative pronoun like “que” but which selects a sentence missing de “de” preposition (instead of a sentence missing an np like “que”)



Note how “dont” is annotated as a “de-obj” argument, which is useful.



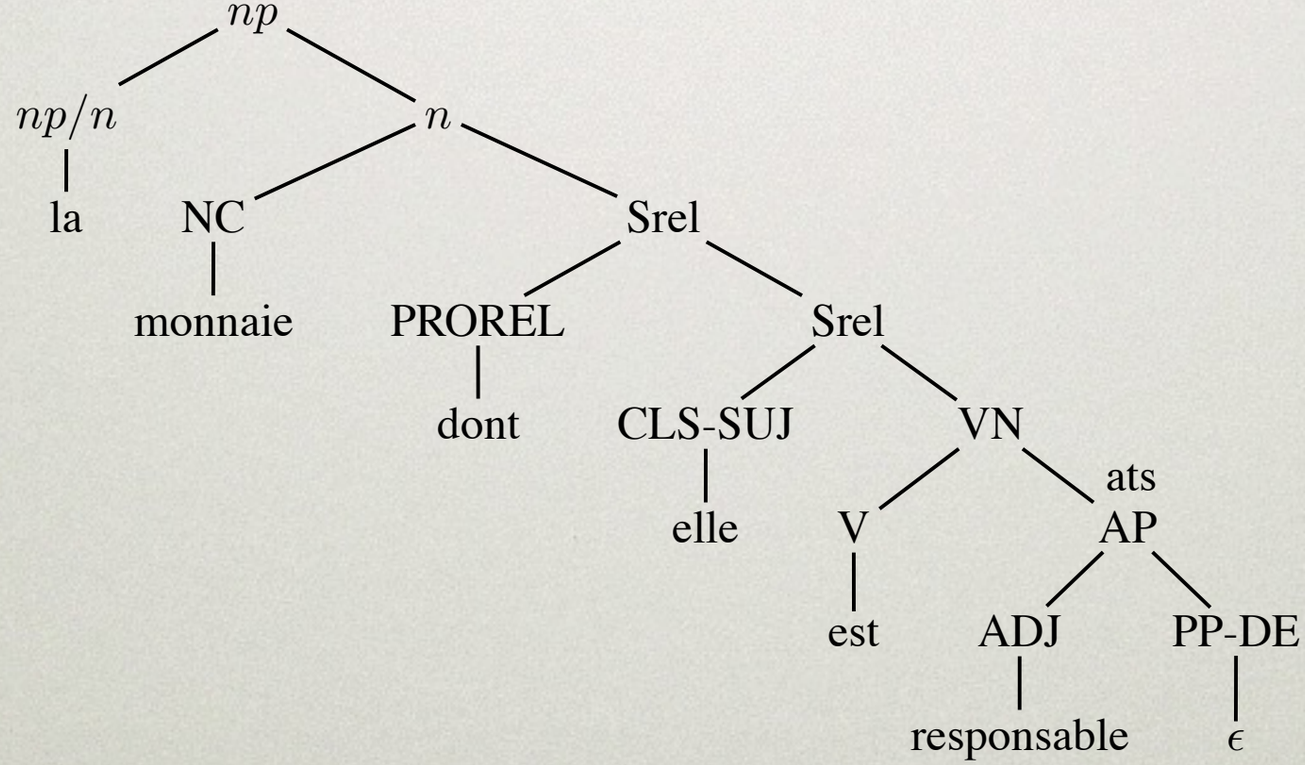
Treebank Extraction



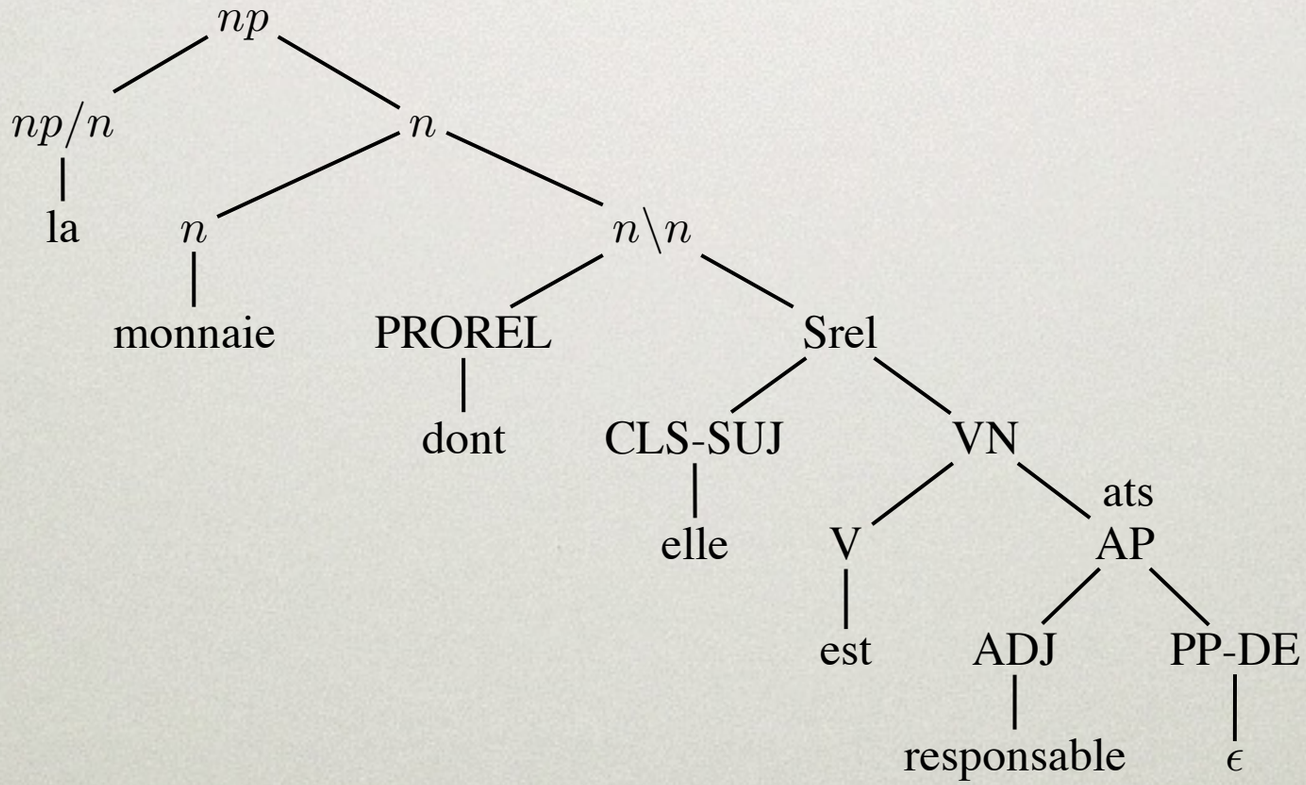
However, the “de” preposition belongs to “responsable” (some adjectives select for prepositions: “responsable de X” functions as an adjective just as “responsable”)

Remark however, that there is no way to derive this from the annotation as it is given. Manual intervention (or at least verification!) is unfortunately necessary to assure the correct placement of the hypothetical preposition.

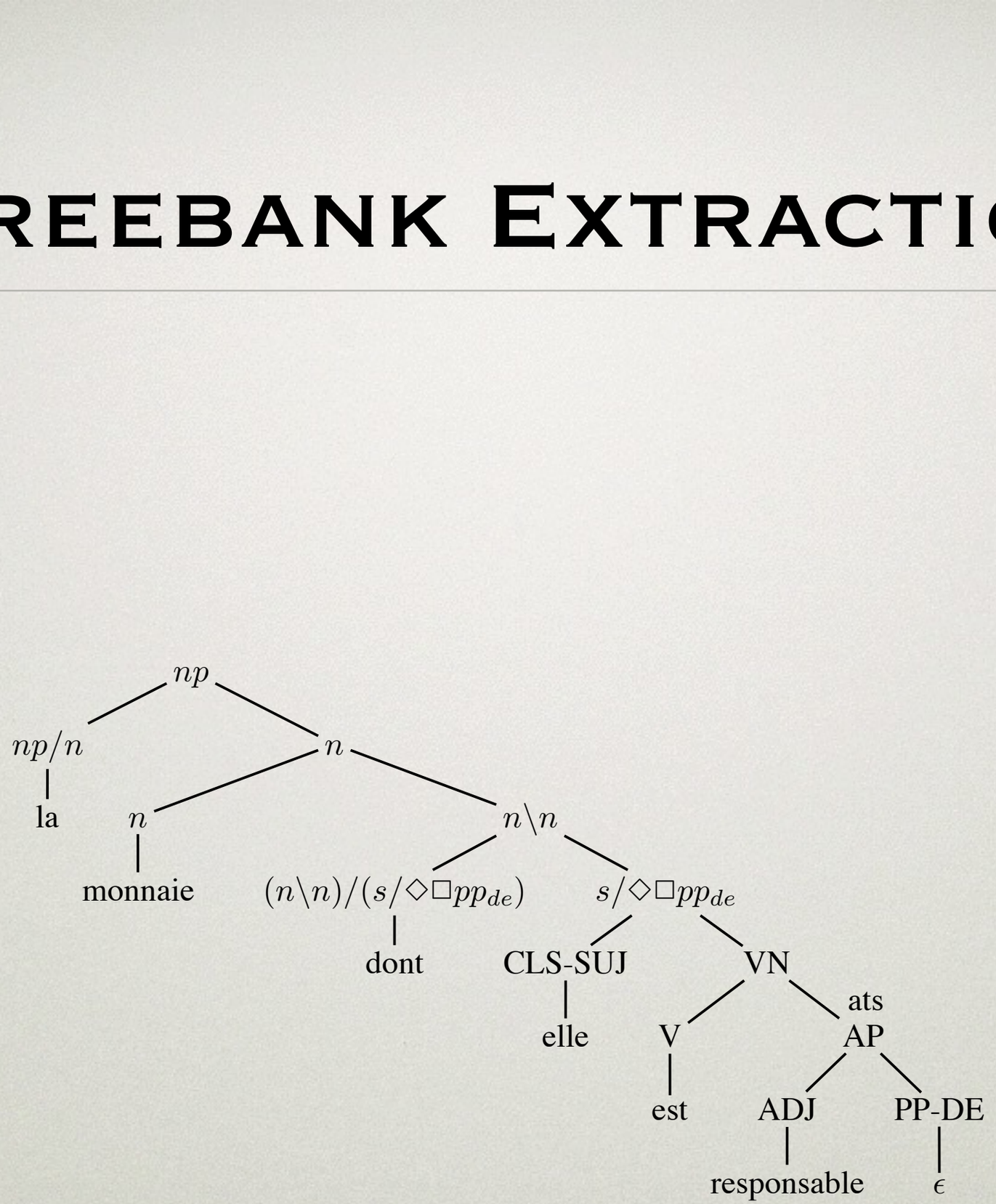
Treebank Extraction



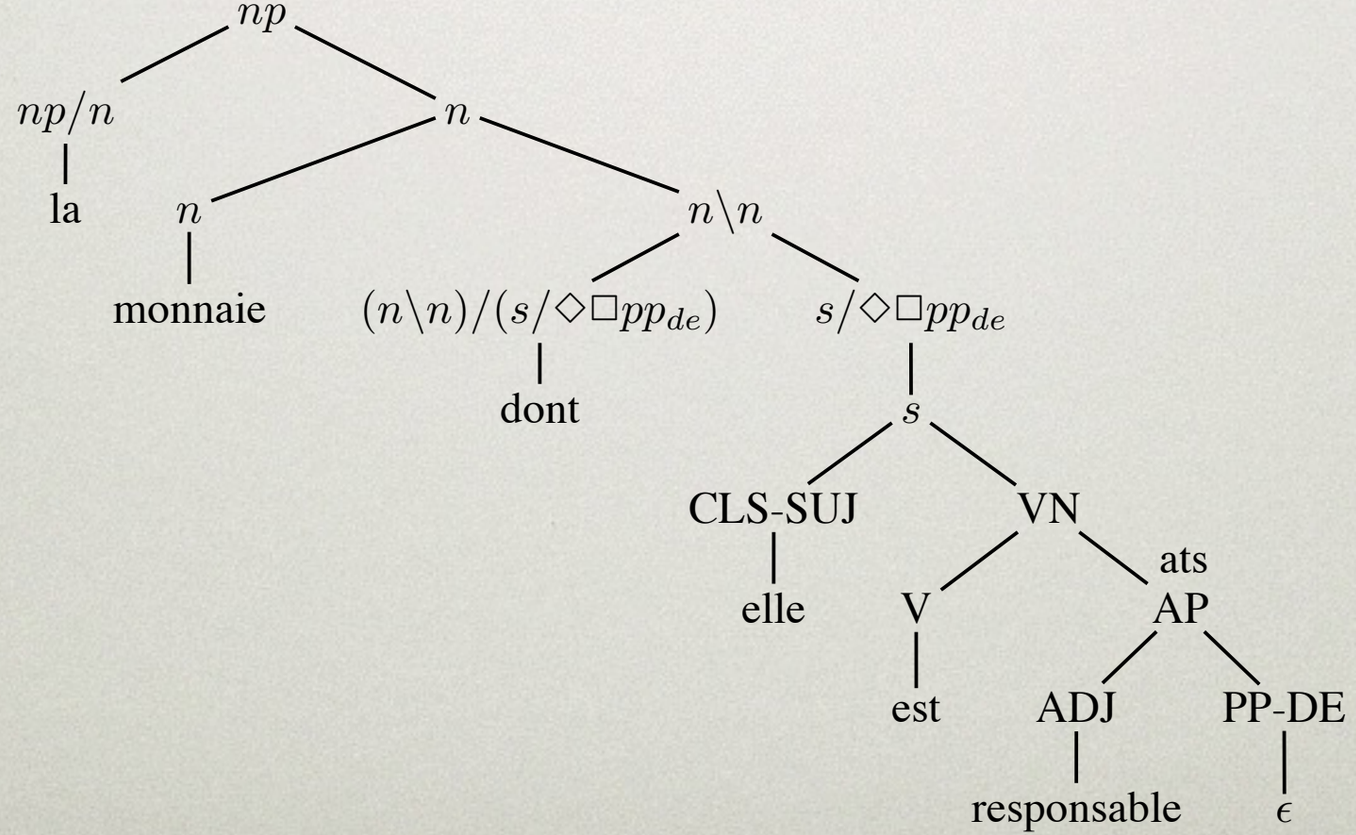
Treebank Extraction



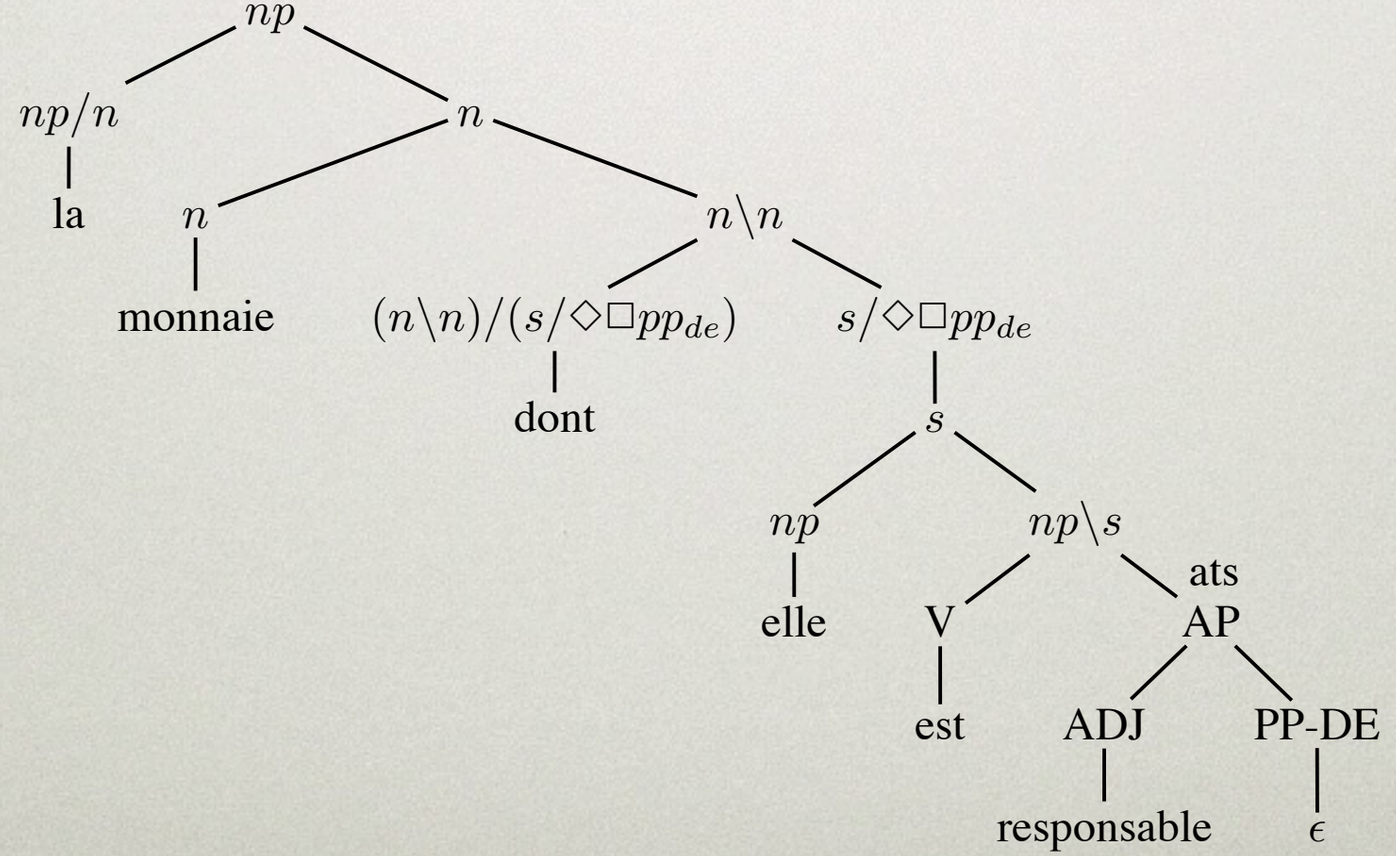
Treebank Extraction



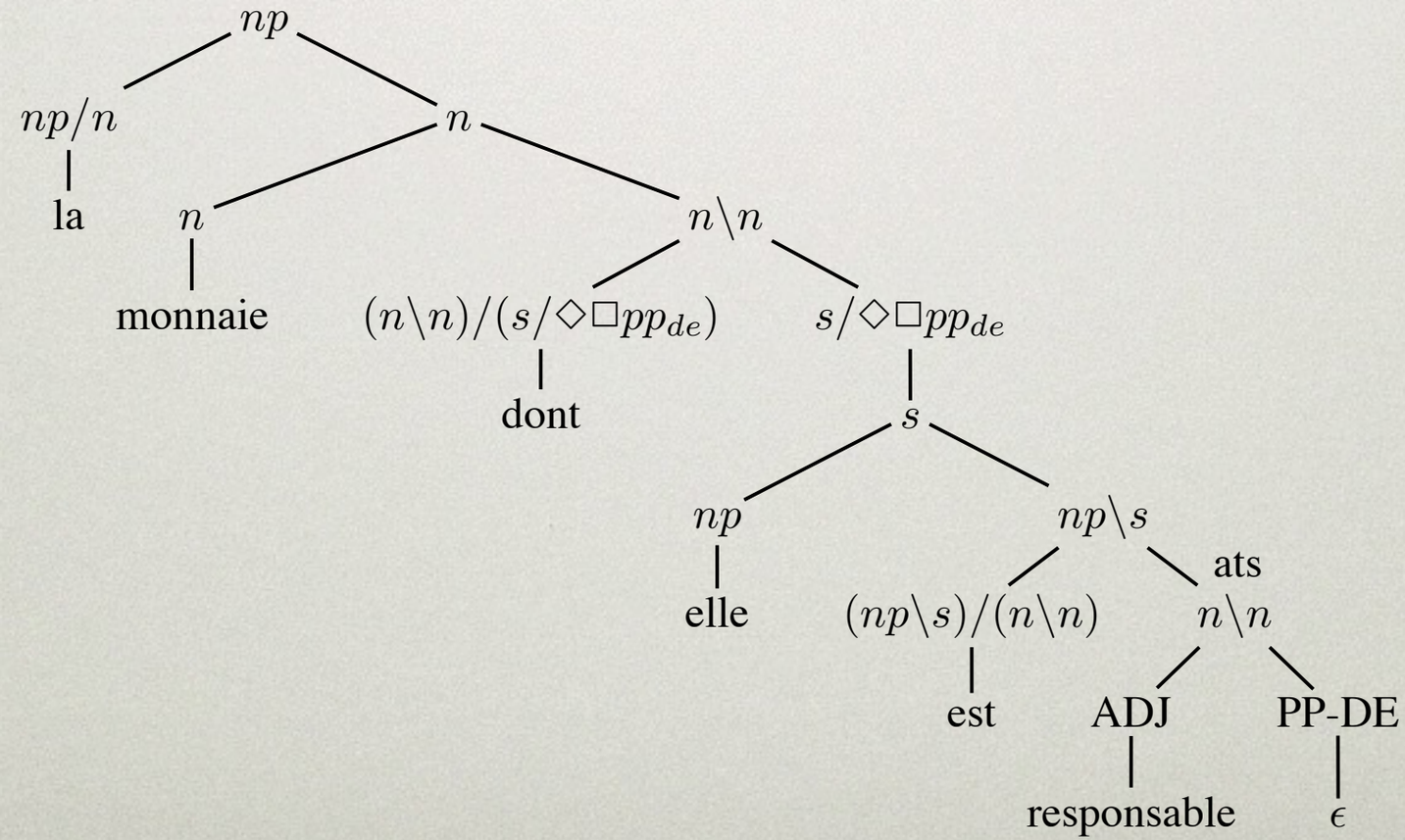
Treebank Extraction



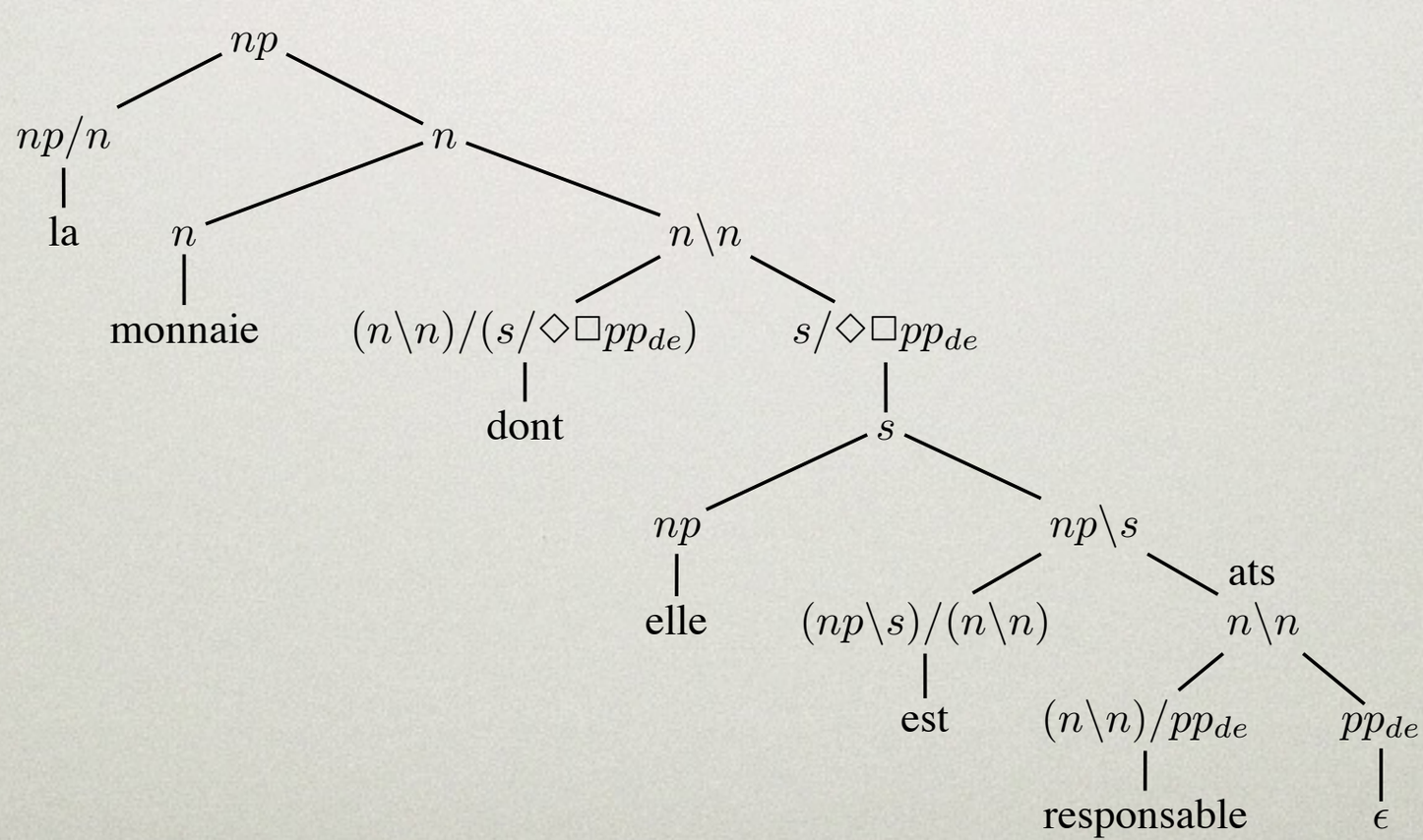
Treebank Extraction



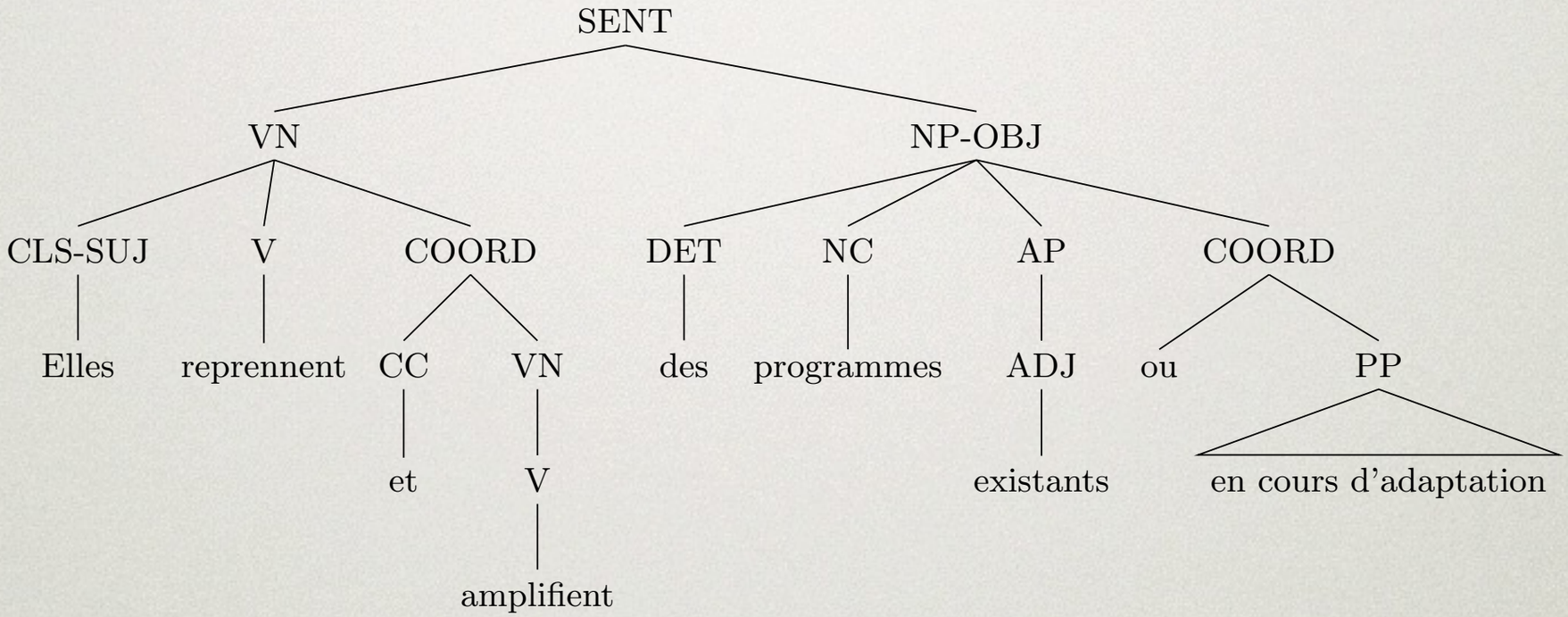
Treebank Extraction



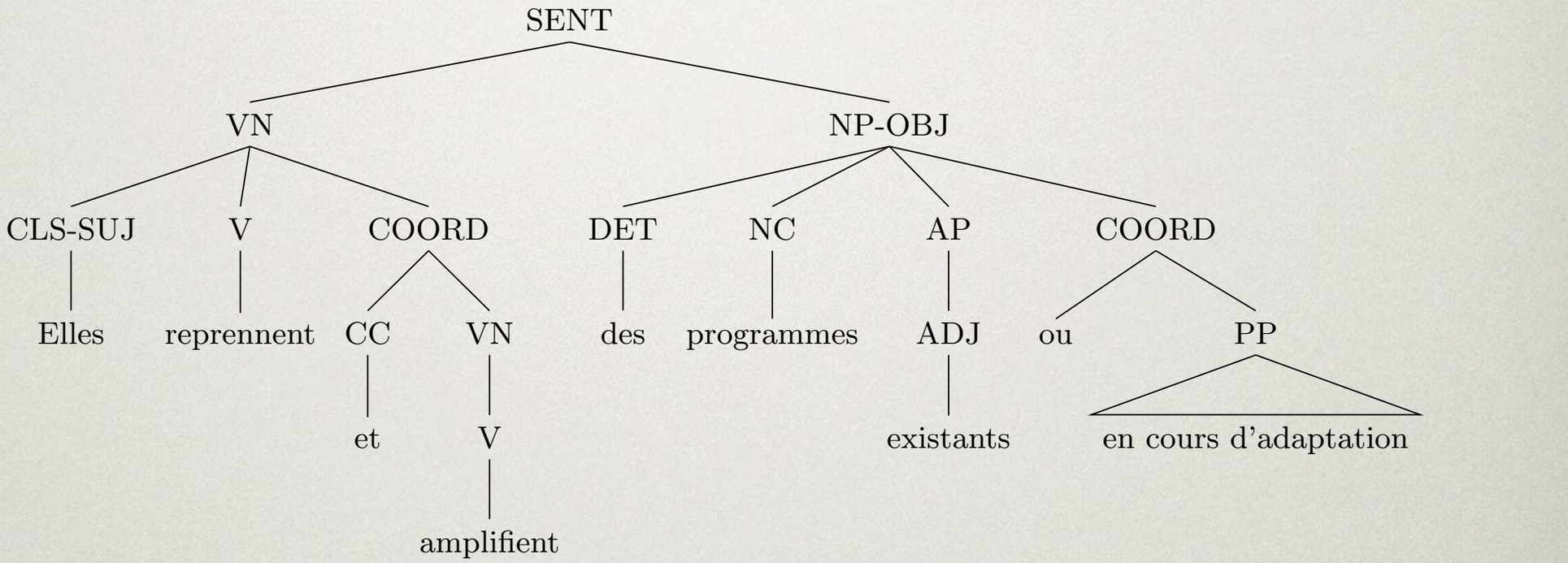
Treebank Extraction



Treebank Extraction: Coordination



Treebank Extraction: Coordination



$$\dfrac{\dfrac{\dfrac{existants}{n\backslash n}\,Lex \quad \dfrac{\dfrac{ou}{((n\backslash n)\backslash(n\backslash n))/(n\backslash n)}\,Lex \quad \dfrac{en\ cours...}{n\backslash n}\,Lex}{(n\backslash n)\backslash(n\backslash n)}\,/E}{n\backslash n}\,\backslash E$$

Statistics About the Extracted French Treebank

- 15,590 sentences 425,918 words
- 45,270 distinct lexical entries
- 883 different formulas (659 occurring more than once — comparison: CGN treebank has 4,768 formulas)
- By comparison: 12,617 CFG rules

How to Improve an Extracted Treebank

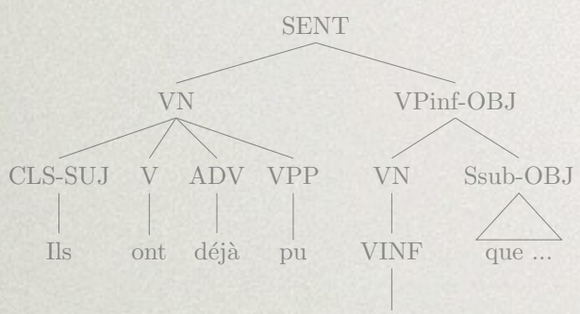
Detective work:

try to identify the problem and look for clues to the underlying cause, eg.:

- annotation error
- need for a more sophisticated extraction strategy
- need to add information by hand

Outline

French Treebank



Grammar

Extraction

$$\dfrac{\dfrac{\text{de}}{(np\backslash s_{di})/(np\backslash s_i)}\,[Lex] \quad \dfrac{\dfrac{\text{s'}}{cl_r}\,[Lex] \quad \dfrac{\dfrac{\text{attaquer}}{(cl_r\backslash(np\backslash s_i))/pp_{\grave{a}}}\,[Lex] \quad \dfrac{}{\text{p}_0 \vdash pp_{\grave{a}}}\,[Hyp]_1}{\text{a} \circ \text{p}_0 \vdash cl_r\backslash(np\backslash s_i)}\,[/E]}{\text{s'} \circ (\text{a} \circ \text{p}_0) \vdash np\backslash s_i}\,[\backslash E]}{\text{de} \circ (\text{s'} \circ (\text{a} \circ \text{p}_0)) \vdash np\backslash s_i}\,[/E]$$

Applications

$$\begin{array}{|l|l|}\hline e_1\ y_1 & \\ \hline y_1 : & \begin{array}{|l|}\hline e_2\ e_3\ x_3 \\ \hline x_3 = \textbf{?} \\ \textbf{aider\_à}(e_2, x_0, x_3, e_3) \\ \textbf{partir}(e_3, x_3) \\ \hline \end{array} \\ \textbf{demander}(e_1, y_0, x_0, y_1) & \\ \hline \end{array}$$

Applications

- Wide-coverage parsing for French
- Wide-coverage semantics for French
- Wide-coverage semantics and parsing for Dutch?

Wide-Coverage Parsing

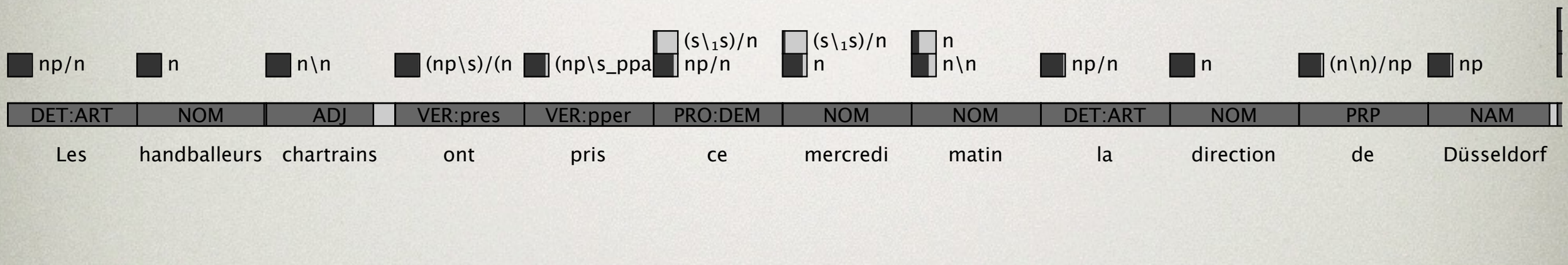
- How can we parse very big grammars efficiently?
- Bottlenecks: lexicon size, grammatical combinatorics

Lexicon Size

- Many frequent words occur with very many different formulas
- Classic solution: supertagging

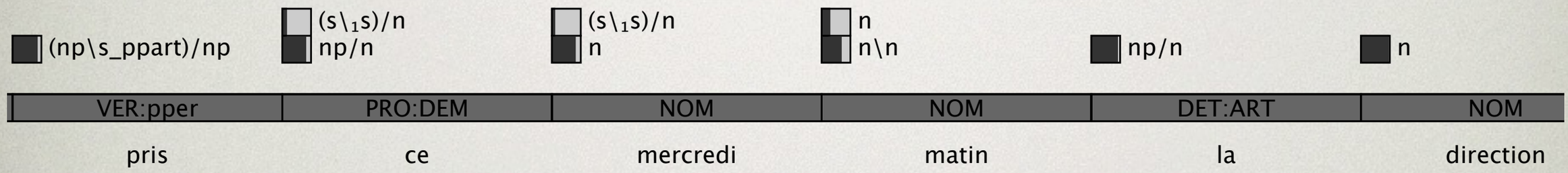
est - “is”	
$(np\backslash s)/np$	23,2 %
$(np\backslash s)/(n\backslash n)$	20,6 %
$(np\backslash s)/(np\backslash s_{pass})$	16,8 %
$(cl_r\backslash(np\backslash s))/(cl_r\backslash(np\backslash s_{ppart}))$	10,8 %
$(np\backslash s)/pp$	8,1 %
$(np\backslash s)/(np\backslash s_{ppart})$	6,3 %
$(np\backslash s)/(np\backslash s_{infX})$	2,8 %
$((np\backslash s)/s_q)/(n\backslash n)$	2,2 %

What Supertagging Does



- Supertagging = statistical finite state approximation of lexical lookup
- Assigns each word the contextually most likely (set of) formulas

What Supertagging Does



- Supertagging = statistical finite state approximation of lexical lookup
- Assigns each word the contextually most likely (set of) formulas

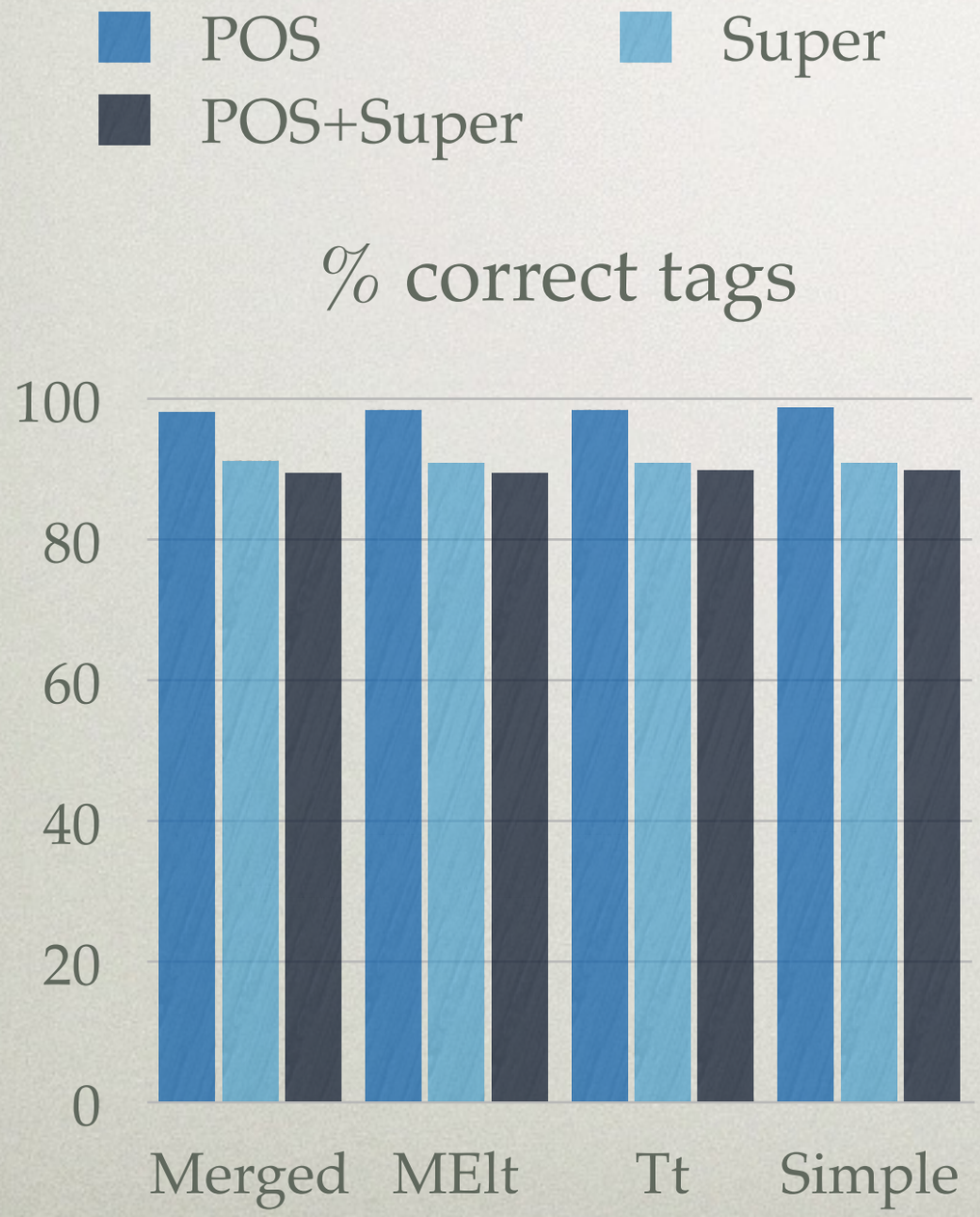
Supertagger Performance

Corpus	POS	Super	0,1	0,01	F/w
FTB	97,8 %	90,6 %	96,4 %	98,4 %	2,3

Supertagger Performance

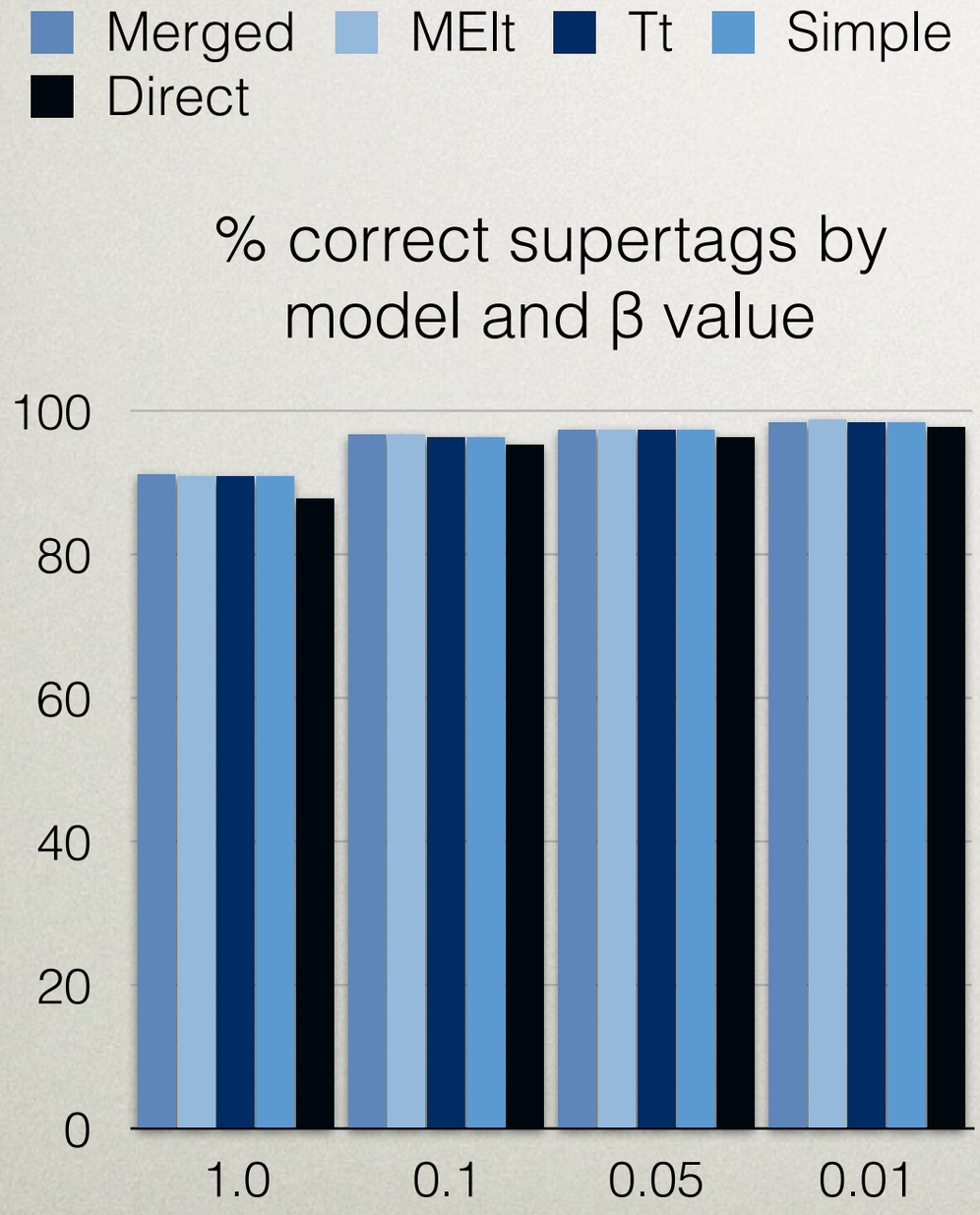
Corpus	POS	Super	0,1	0,01	F/w
FTB	97,8 %	90,6 %	96,4 %	98,4 %	2,3
Le Monde 2010	97,3 %	89,9 %	95,8 %	97,9 %	2,2
Sequoia / Annodis	97,3 %	88,1 %	94,8 %	97,6 %	2,4
Itipy / Forbes	95,7 %	86,7 %	93,8 %	97,1 %	2,6

Detailed Performance



POS-merged	98.0
Super-merged	90.4
POS+Super	88.7

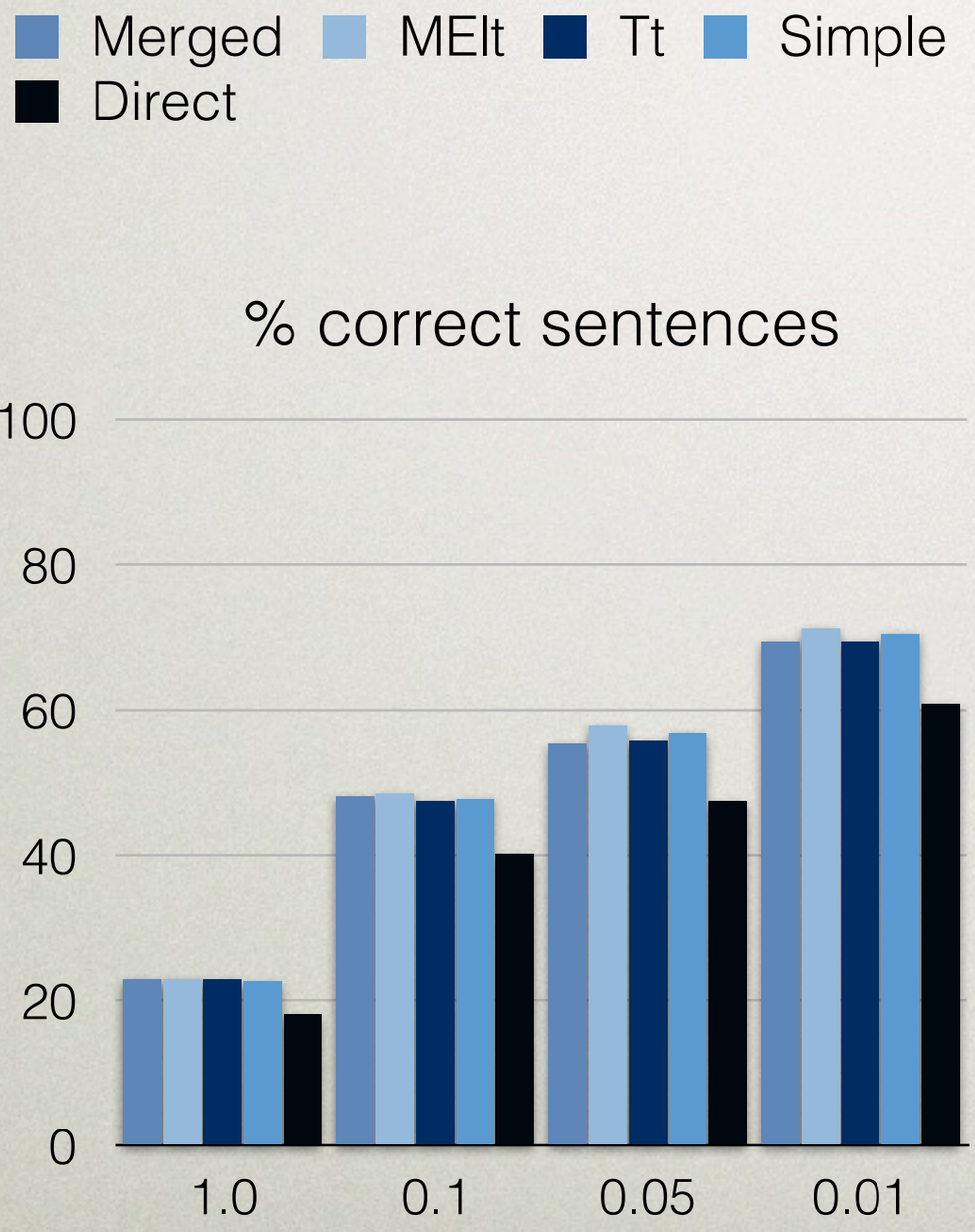
Detailed Performance



% correct supertags by model and β value

- Results with the use of different values of β.
- In a sense, the β value allows us to trade coverage for efficiency: at higher values of β, we parse more sentences, but we do so more slowly.

Detailed Performance



- Finally, here is the percentage of sentences which are assigned the correct sequence of supertags for the different settings of β and the different POS models.
- Note that the number of sentences for which a parse is found is actually better (close to 90% at β=0.01)

Chart-Parsing Type-Logical Grammars

- Prototype exhaustive chart parser for Type-Logical Grammars, strongly inspired by Shieber e.a. (1995)
- Fine-tuned for grammars in the restricted form as produced by the extraction algorithm
- LaTeX output

Chart-Parsing Type-Logical Grammars

Chart-Parsing Type-Logical Grammars

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\text{Afin} \vdash ((s/(np\backslash s_{main}))/np)/(np\backslash s_{inf})}\,[Lex] \quad \dfrac{}{\text{de} \vdash (np\backslash s_{inf})/(np\backslash s_{inf})}\,[Lex]}{\text{de} \circ (\text{raconter} \circ (\text{l'} \circ (\text{histoire} \circ (\text{qu'} \circ (\text{ils} \circ (\text{ont} \circ \text{découverte})))))) \vdash np\backslash s_{inf}}\,[/E]}{\text{Afin} \circ (\text{de} \circ (\text{raconter} \circ (\text{l'} \circ (\text{histoire} \circ (\text{qu'} \circ (\text{ils} \circ (\text{ont} \circ \text{découverte})))))) \vdash (s/(np\backslash s_{main}))/np}\,[/E] \quad \dfrac{}{\text{raconter} \vdash (np\backslash s_{inf})/np}\,[Lex]}{\text{l'} \circ (\text{histoire} \circ (\text{qu'} \circ (\text{ils} \circ (\text{ont} \circ \text{découverte})))) \vdash np}\,[/E] \quad \dfrac{}{\text{l'} \vdash np/n}\,[Lex]}{\text{histoire} \circ (\text{qu'} \circ (\text{ils} \circ (\text{ont} \circ \text{découverte}))) \vdash n}\,[/E] \quad \dfrac{}{\text{histoire} \vdash n}\,[Lex]}{\text{qu'} \circ (\text{ils} \circ (\text{ont} \circ \text{découverte})) \vdash n\backslash n}\,[\backslash E] \quad \dfrac{}{\text{qu'} \vdash (n\backslash n)/(s_{main}/\Diamond_1\Box_1^{\downarrow}np)}\,[Lex]}{\text{ils} \circ (\text{ont} \circ \text{découverte}) \vdash s_{main}}\,[E\ end] \quad \dfrac{}{\text{ils} \vdash np}\,[Lex]}{\text{ont} \circ \text{découverte} \vdash np\backslash s_{main}}\,[\backslash E] \quad \dfrac{}{\text{ont} \vdash (np\backslash s_{main})/(np\backslash s_{ppart})}\,[Lex]}{\text{découverte} \vdash np\backslash s_{ppart}}\,[/E] \quad \dfrac{}{\text{qu'} \vdash (n\backslash n)/(s/\Diamond_1\Box_1^{\downarrow}np)}\,[Lex]}{\text{découverte} \vdash (np\backslash s_{ppart})/np}\,[E\ start] \quad \dfrac{}{\text{découverte} \vdash (np\backslash s_{ppart})/np}\,[Lex]$$

Chart-Parsing Type-Logical Grammars

$$\dfrac{\dfrac{\dfrac{}{\text{histoire} \vdash n}\,[Lex] \quad \dfrac{}{\text{qu'} \vdash (n\backslash n)/(s_{main}/\Diamond_1 \square_1^{\downarrow} np)}\,[Lex] \quad \dfrac{\dfrac{}{\text{ils} \vdash np}\,[Lex] \quad \dfrac{\dfrac{}{\text{ont} \vdash (np\backslash s_{main})/(np\backslash s_{ppart})}\,[Lex] \quad \dfrac{\dfrac{}{\text{qu'} \vdash (n\backslash n)/(s/\Diamond_1 \square_1^{\downarrow} np)}\,[Lex] \quad \dfrac{}{\text{découverte} \vdash (np\backslash s_{ppart})/np}\,[Lex]}{\text{découverte} \vdash np\backslash s_{ppart}}\,[/E]}{\text{ont} \circ \text{découverte} \vdash np\backslash s_{main}}\,[\backslash E]}{\text{ils} \circ (\text{ont} \circ \text{découverte}) \vdash s_{main}}\,[E\ end]}{\text{qu'} \circ (\text{ils} \circ (\text{ont} \circ \text{découverte})) \vdash n\backslash n}\,[\backslash E]}{\text{histoire} \circ (\text{qu'} \circ (\text{ils} \circ (\text{ont} \circ \text{découverte}))) \vdash n}\,[/E]$$

Semantics

- Type-logical grammar proofs are a subset of intuitionistic proofs, which correspond to lambda-terms.
- Lexical substitution follow by beta normalization gives us the full sentence meaning

Lexical Semantics

- Montague-style semantics, where the meaning of *love* is **love'**
- But which has the advantage of being scaleable, since many lexical entries follow a specific pattern
- Uses DRT

Semantics

Example entries (slightly simplified)

marché: $\lambda x.$ | x |
| --- |
| **marché(x)** |

Marie: $\lambda P.$ | y | $\oplus$ **(P y)**
| --- | --- |
| **nommé(y,Marie)** | |

chaque: $\lambda P\ \lambda Q.($	z	$\oplus$ **(P z))** $\rightarrow$ **(Q z)**

Semantics

Example entries (slightly simplified)

aime: (np\s)/np $\lambda y.\ \lambda x.$

aimer(x,y)

semble: $(np\backslash s)/(np\backslash s_{inf})$ $\lambda P.\ \lambda x.$

s
sembler(s) s: (P x)

veut: $(np\backslash s)/(np\backslash s_{inf})$ $\lambda P.\ \lambda x.$

s
vouloir(x,s) s: (P x)

Semantics

- 484 words in the lexicon, with idiosyncratic properties
- 348 lexical schemata, eg. (np\s)/np for word w means semantics w' ($\equiv \lambda y.\lambda x.$ $w'(x,y)$)

Conclusions

- The Type-Logical Treebank is an essential component of a wide-coverage French parser
- Because Type-Logical Grammars factorize grammars in a way permitting an easy connection to formal semantics, we can use the treebank for wide-coverage *semantics* as well