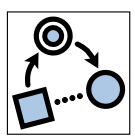
Grammar Implementation with Lexicalized Tree Adjoining Grammars and Frame Semantics

Grammar implementation with XMG: Syntax

Laura Kallmeyer, Timm Lichte, Rainer Osswald & Simon Petitjean

University of Düsseldorf

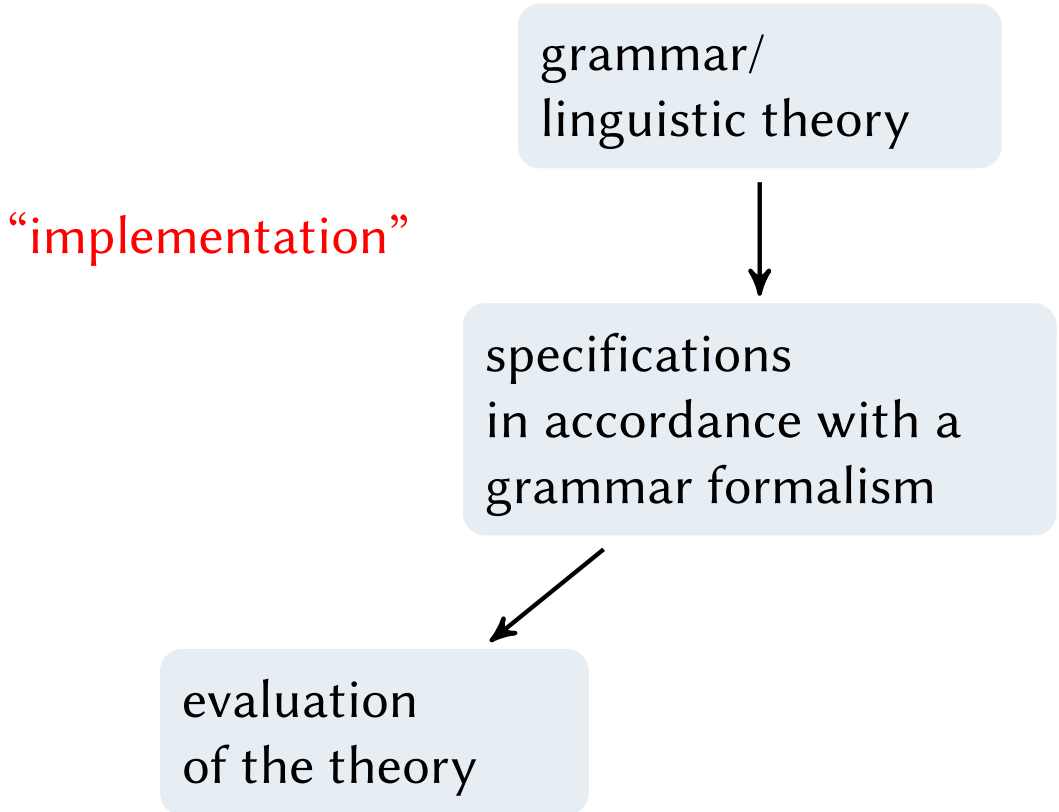
DGfS Fall School, September 19, 2017



Recapitulation of yesterday

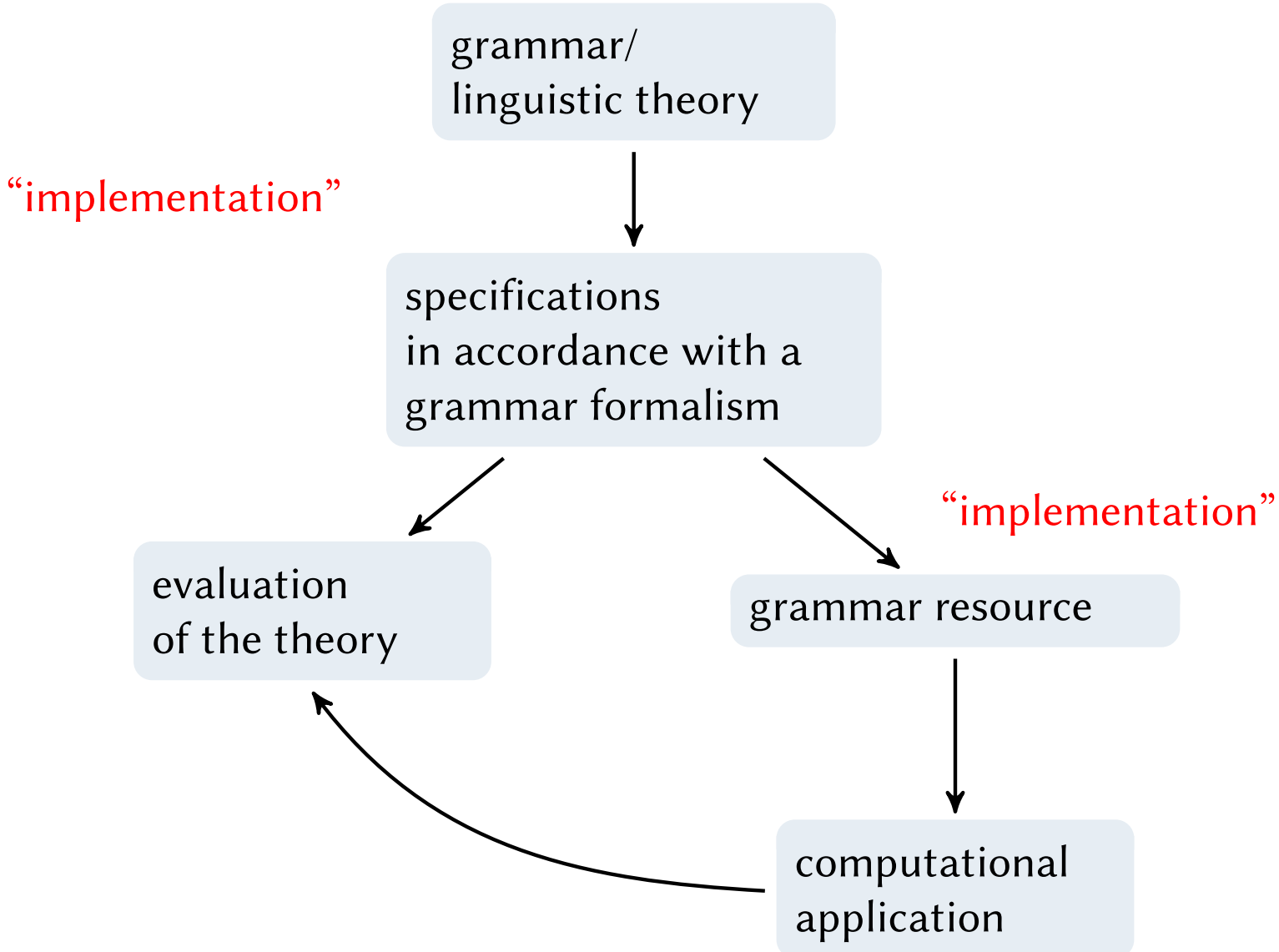
- two meanings of “implementation”
- two techniques of grammar implementation: metarules versus metagrammar
- introduction to grammar engineering with XMG

Two kinds of grammar implementation



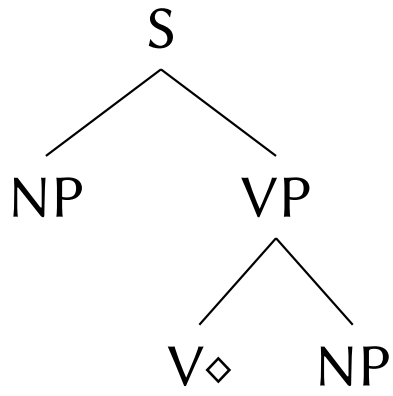
As is frequently pointed out but cannot be overemphasized, an important goal of formalization in linguistics is to enable subsequent researchers to see the defects of an analysis as clearly as its merits; only then can progress be made efficiently. (Dowty 1979: 322)

Two kinds of grammar implementation



What kind of grammar resource?

tree template



lexical insertion

anchor

repairs

Today

- XTAG: an example for a manually constructed grammar
- Examples with XMG: active-passive alternation, wh-movement, multi-word expressions

The XTAG-project

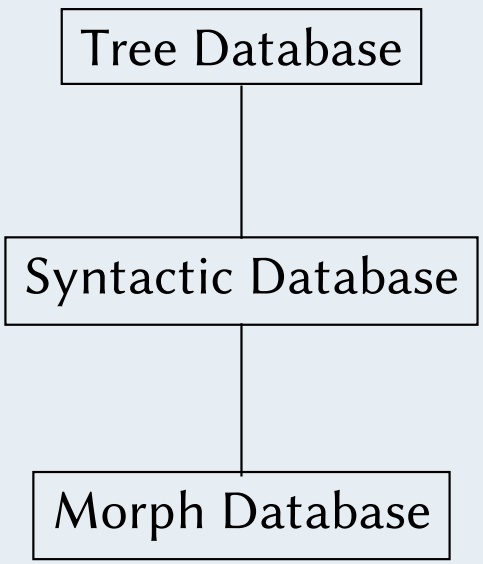
- was located at the University of Pennsylvania (ca. 1988-2001)
  - **grammar** (set of tree templates/families)
  - **tools** (browser, editor, parser, …)

The XTAG-project

- was located at the University of Pennsylvania (ca. 1988-2001)
  - **grammar** (set of tree templates/families)
  - **tools** (browser, editor, parser, ...)
- URL: `http://www.cis.upenn.edu/~xtag/`
  [Manual: XTAG Research Group [14]]

The XTAG-project

- was located at the University of Pennsylvania (ca. 1988-2001)
  - **grammar** (set of tree templates/families)
  - **tools** (browser, editor, parser, ...)
- URL: `http://www.cis.upenn.edu/~xtag/`
  [Manual: XTAG Research Group [14]]
- the architecture of the XTAG-grammar

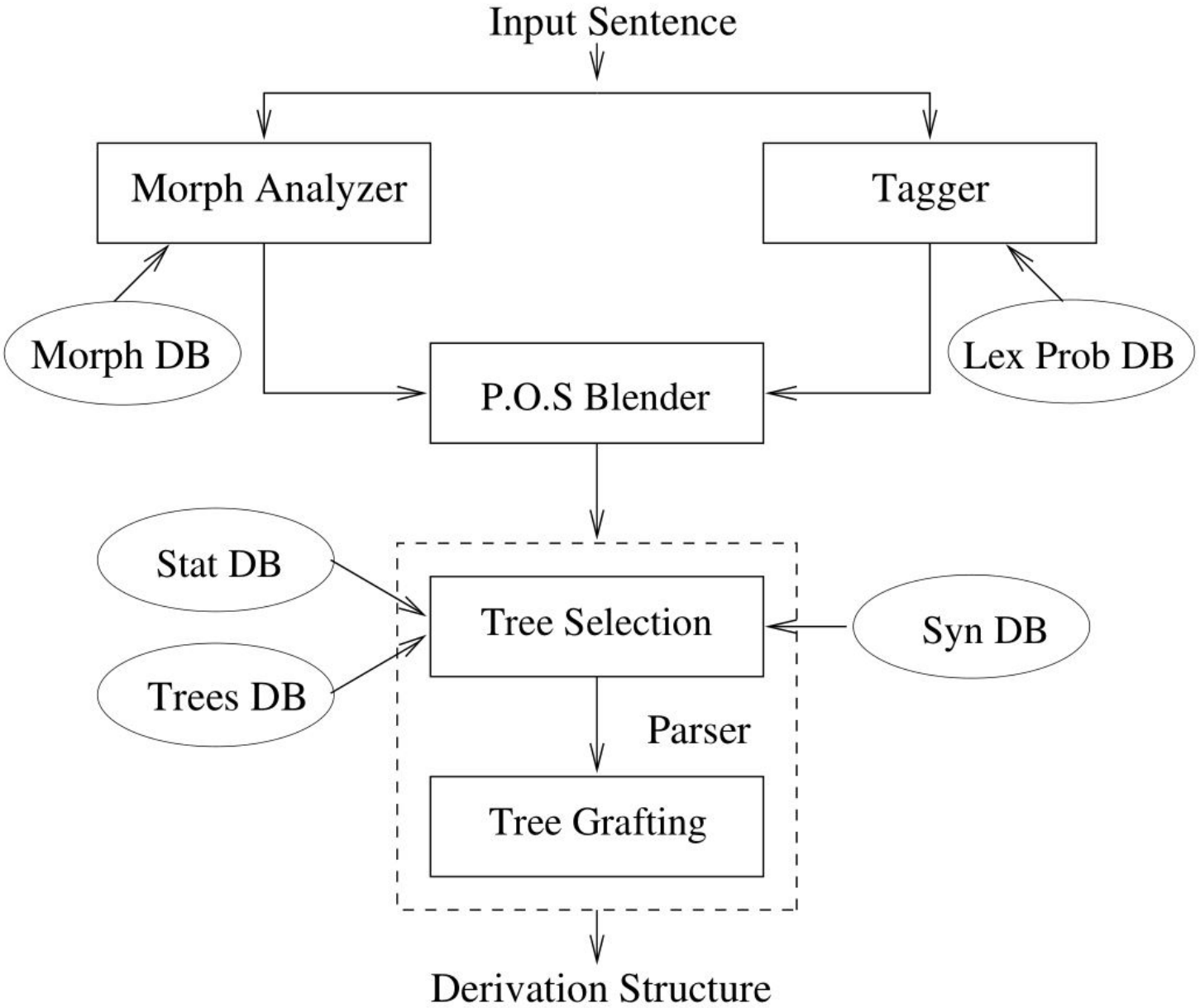


list of tree templates and tree families

root form, POS → list of tree templates or
or tree families, list of feature equations

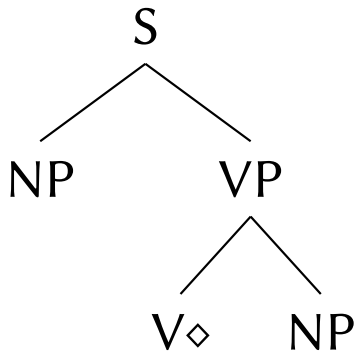
inflected form → root form,
POS, inflectional information

The architecture of the XTAG-grammar



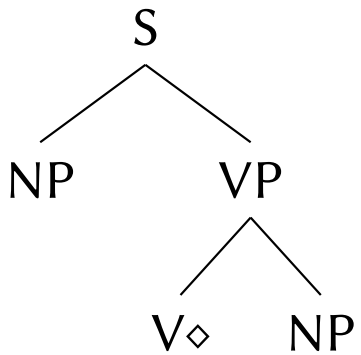
The architecture of the XTAG-grammar

Example: **Tree template** for the declarative transitive verb (αnx0Vnx1), where $\diamond$ marks the lexical insertion site:



The architecture of the XTAG-grammar

Example: **Tree template** for the declarative transitive verb (αnx0Vnx1), where $\diamond$ marks the lexical insertion site:



A tree family

- is a set of tree templates,
- represents a subcategorization frame, and
- contains all syntactic configurations the subcategorization frame can be realized in.

Example: αnx0Vnx1 $\in$ Tnx0Vnx1

Tree families

Example tree families

- intransitive: `Tnx0V`
  tree set containing: base tree, wh-moved subject, imperative, determiner gerund, ... etc.

Tree families

Example tree families

- intransitive: `Tnx0V`
  tree set containing: base tree, wh-moved subject, imperative, determiner gerund, ... etc.
- transitive: `Tnx0Vnx1`
  tree set containing: base tree, wh-moved subject, wh-moved object, imperative, determiner gerund, passive with *by*, passive without *by* ... etc.

Tree families

Example tree families

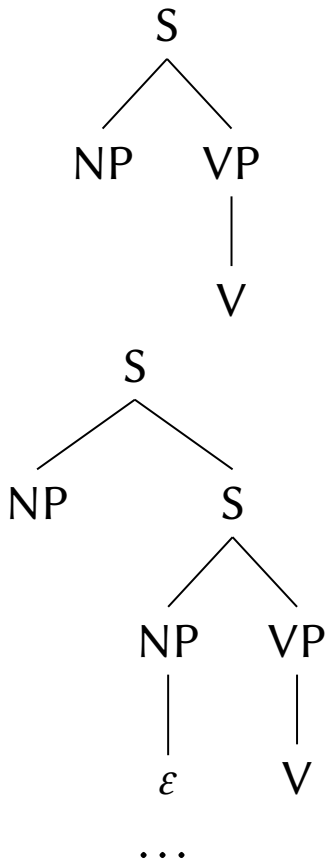
- intransitive: `Tnx0V`
  tree set containing: base tree, wh-moved subject, imperative, determiner gerund, ... etc.
- transitive: `Tnx0Vnx1`
  tree set containing: base tree, wh-moved subject, wh-moved object, imperative, determiner gerund, passive with *by*, passive without *by* ... etc.

Some figures [Prolo [12]]

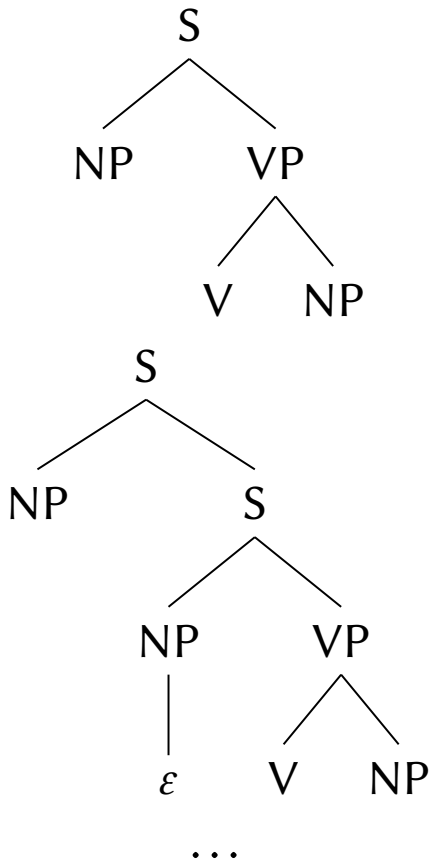
subcat. group	no. of families	no. of trees
intransitive	1	12
transitive	1	39
ditransitive	1	46
light verb constr.	2	53
⋮	⋮	⋮
TOTAL:	57	1008

The situation: templates ...

12 templates for intransitive verbs



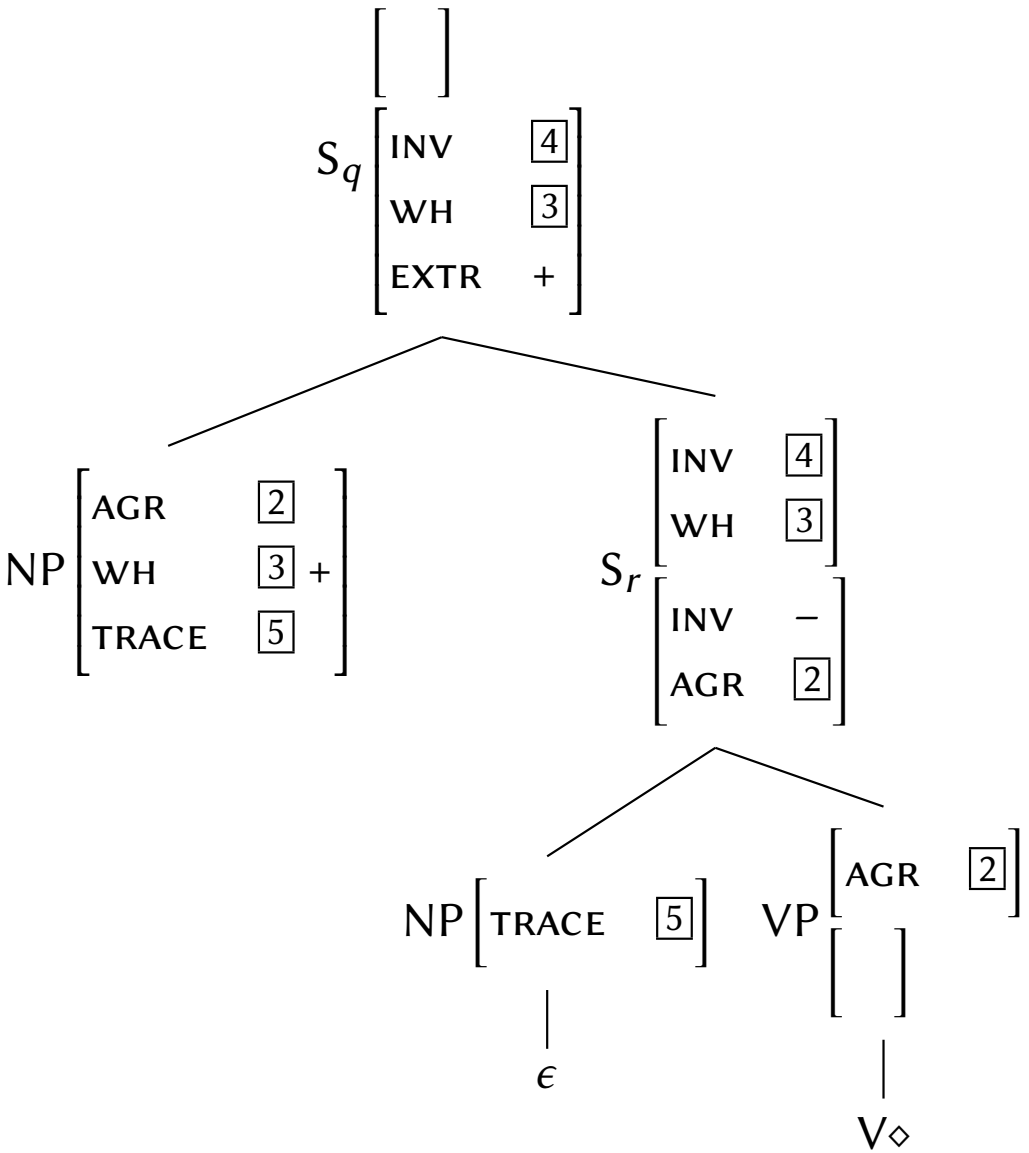
39 tree templates for transitive verbs



Basically, XTAG defines a set of 1008 unrelated tree templates.

The situation: templates with features

αW0nx0V:



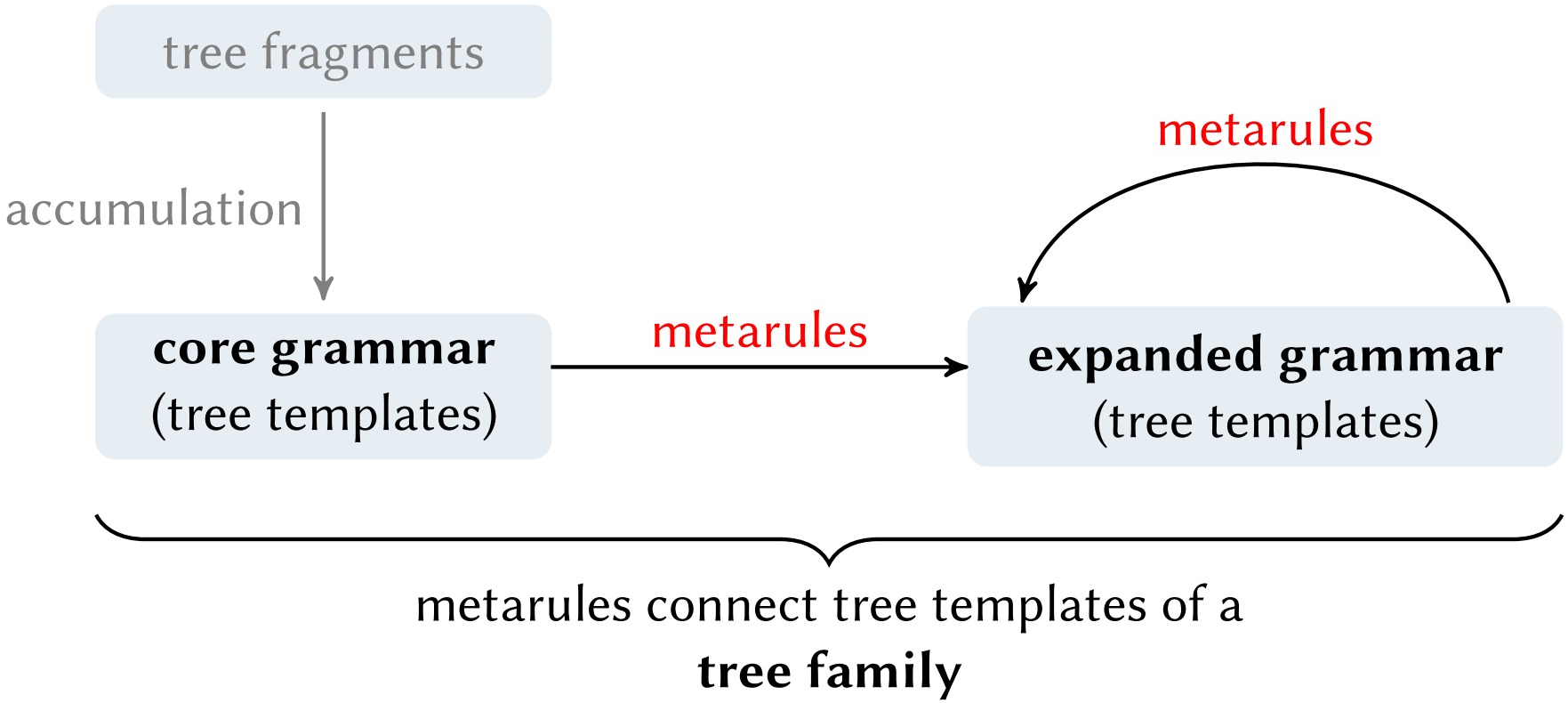
The situation: templates with **many** features

See full entries here:

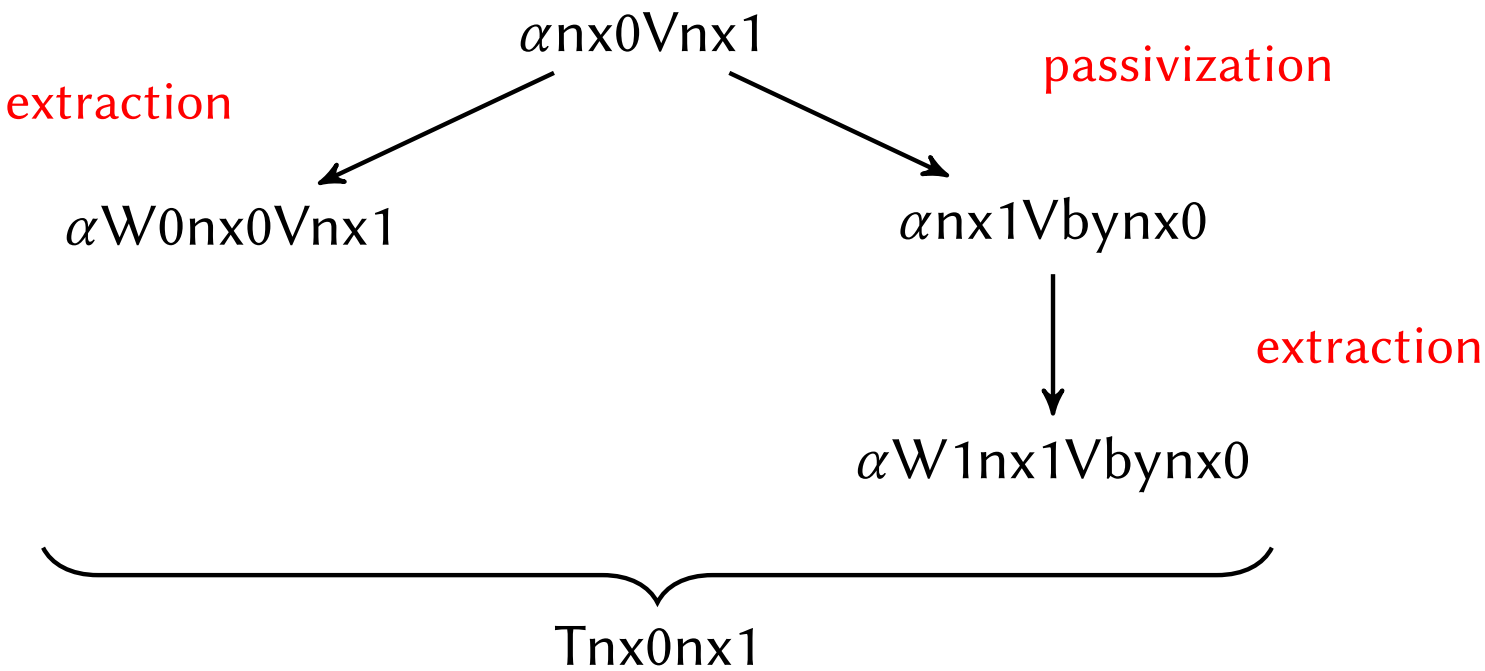
`http://xmg.phil.hhu.de/index.php/upload/xmg_xtag`

Metarules for LTAG

Idea from GPSG[10], later applied to XTAG[2,3,12]

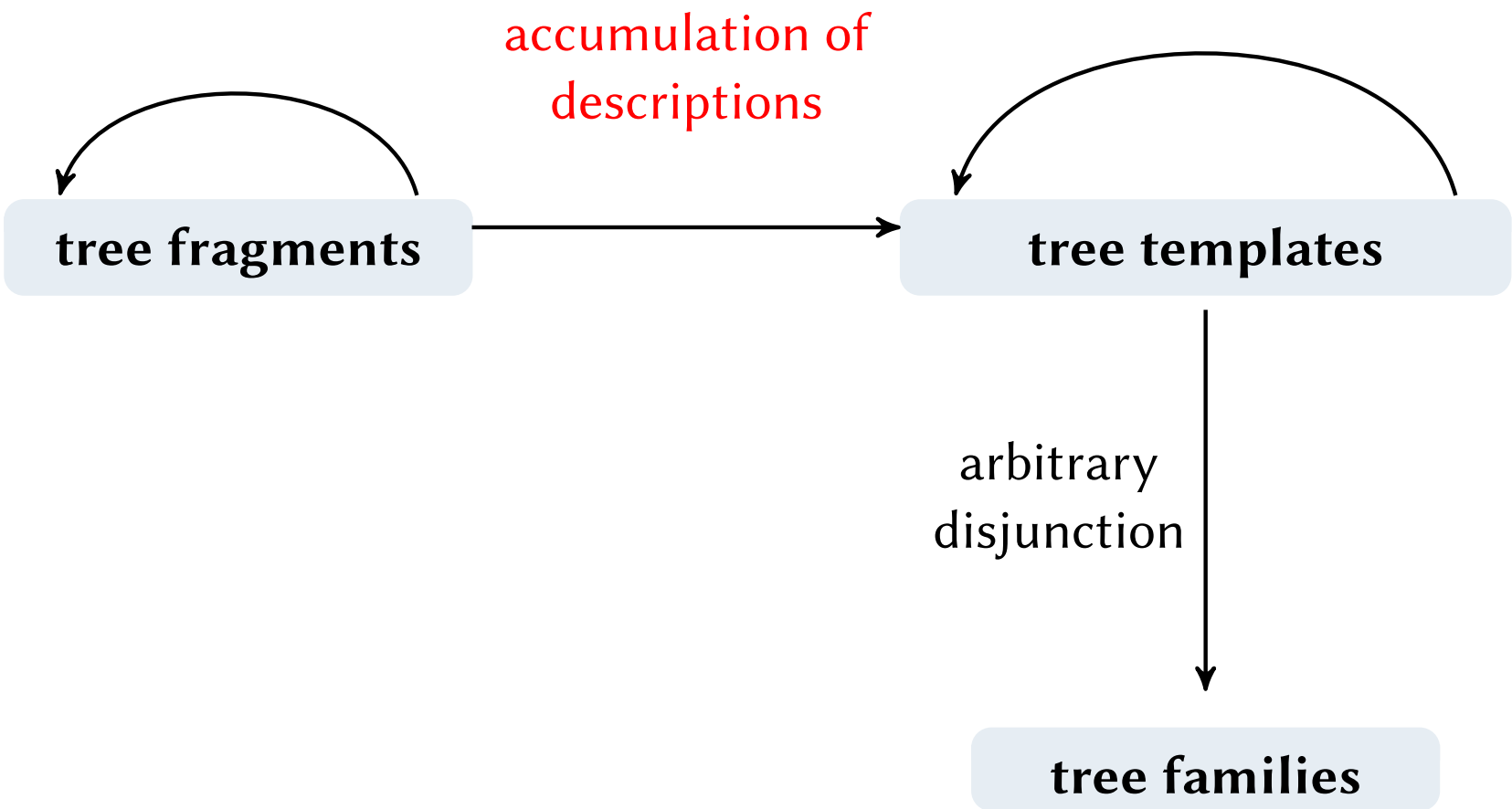


Metarules for LTAG: Example



Metagrammars for LTAG

Candito (1996)[7,8,13]

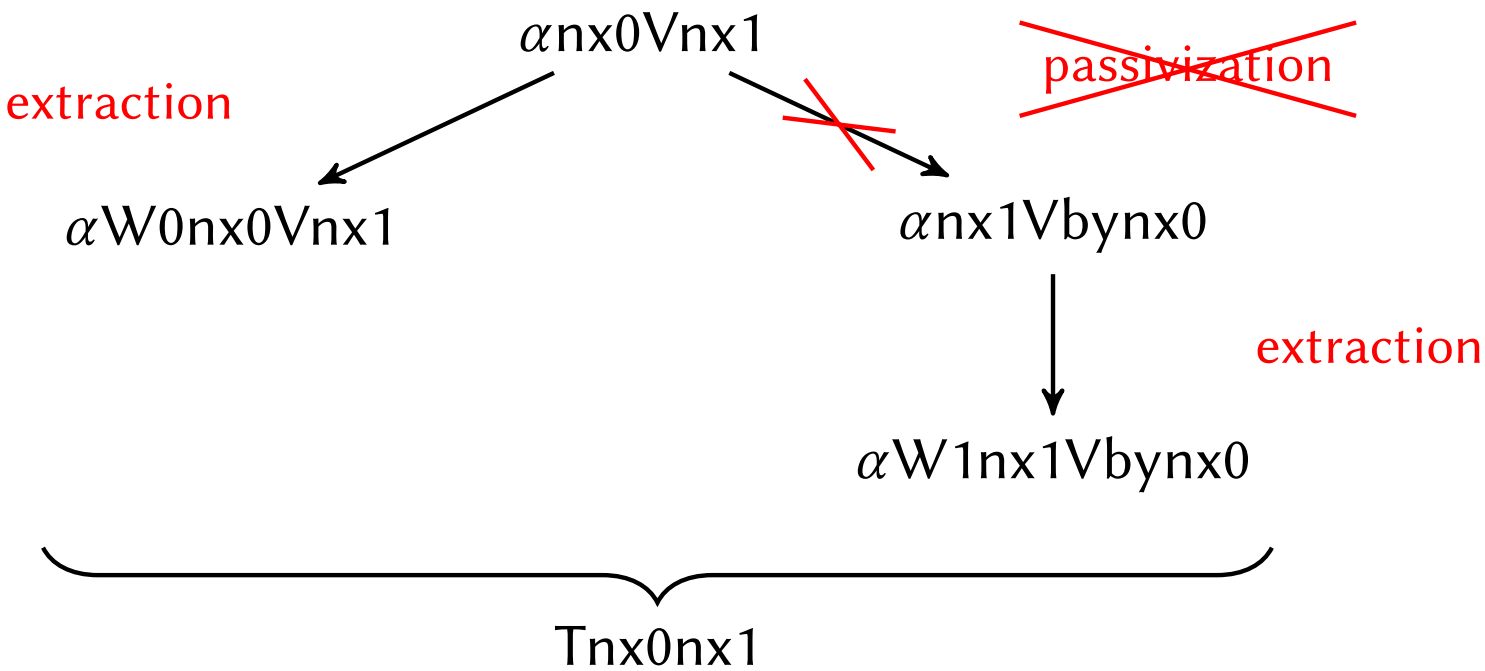


Metagrammars for LTAG: Properties

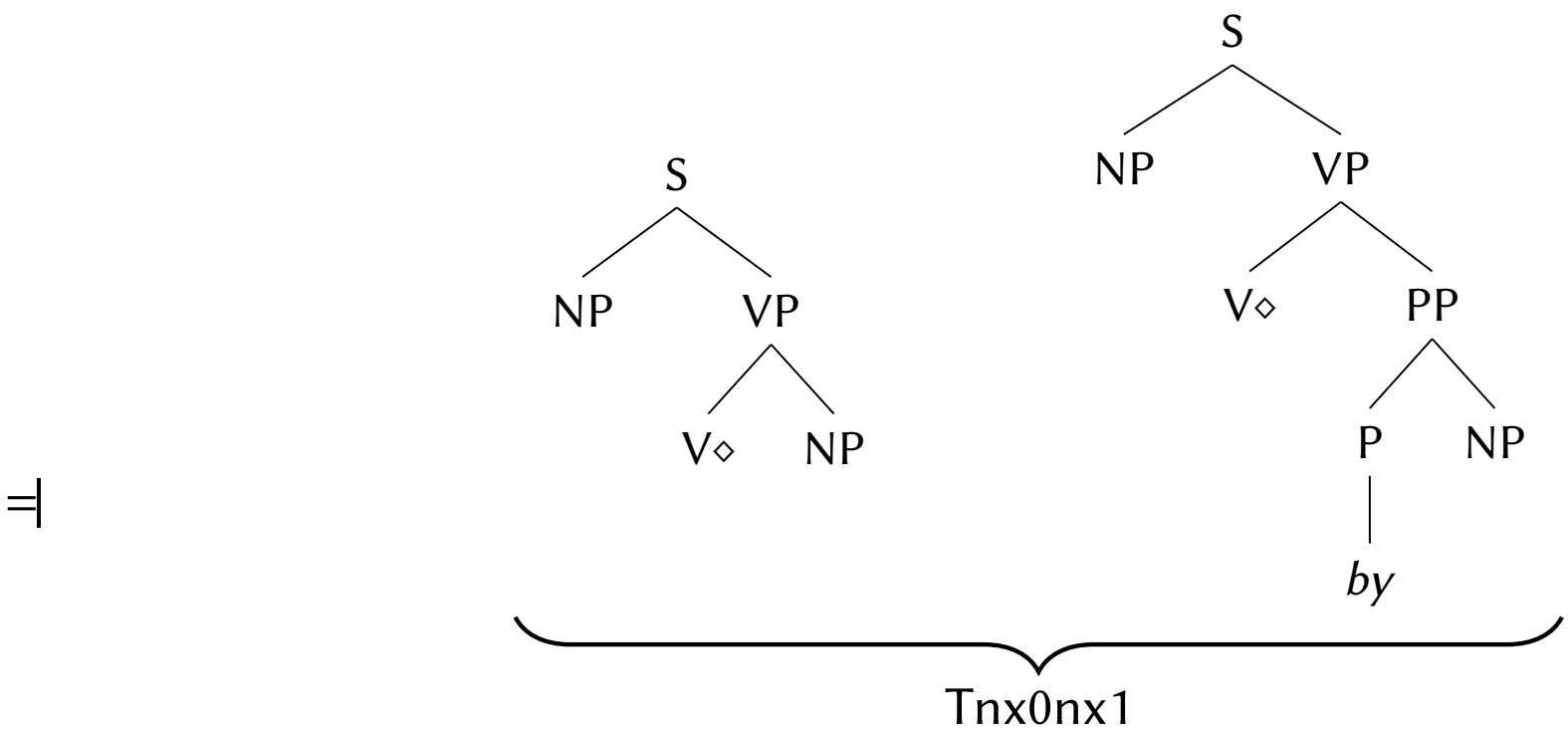
- no deletion, no copying, no recursion
- declarative, order insensitive
- The number of minimal models is finite.
- BUT: the number of minimal models can grow exponentially ($\mathcal{O}(n!)$) in terms of the number of described nodes.

Does it suffice? How to express passivization?

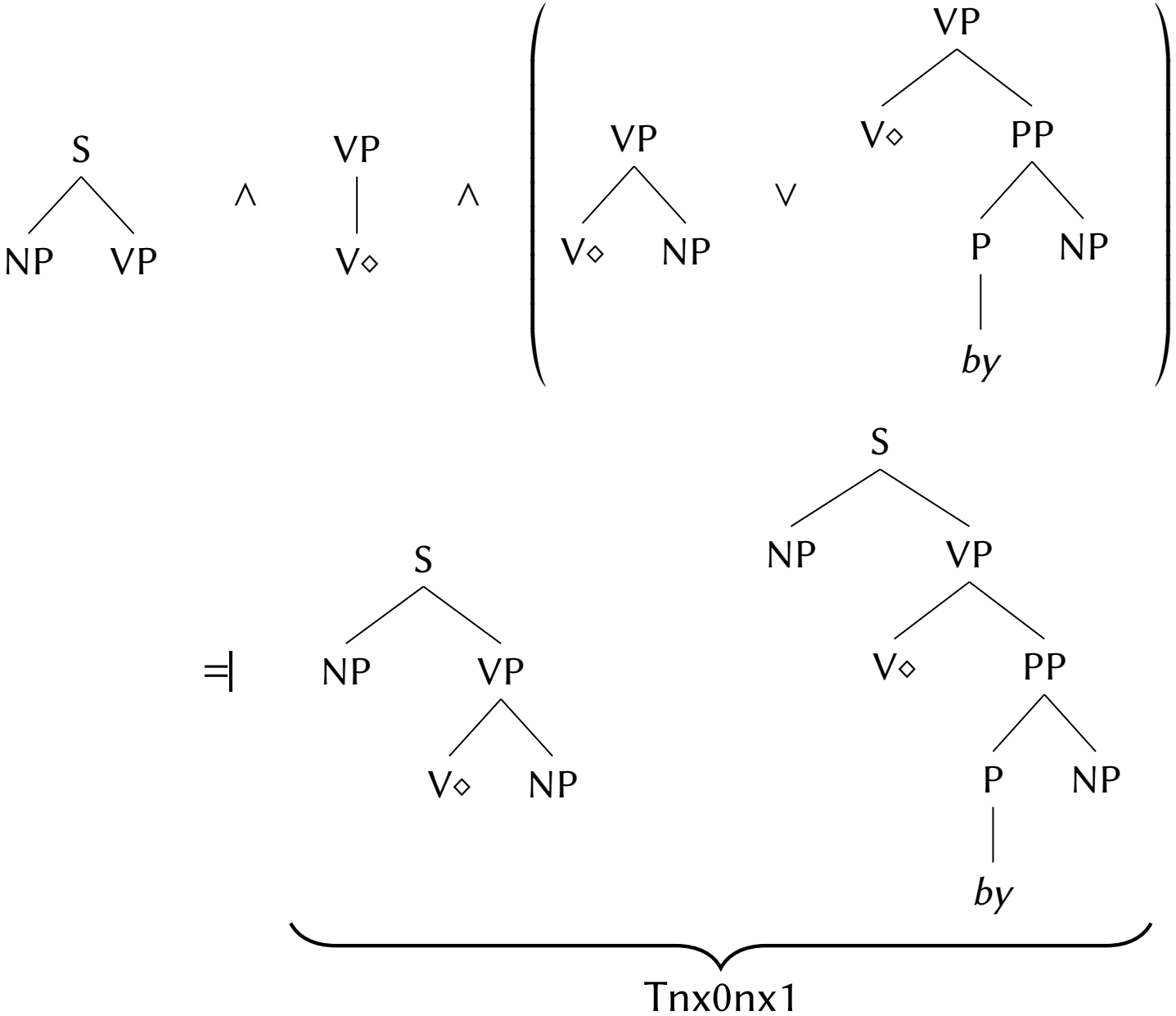
Metagrammar for LTAG: Example



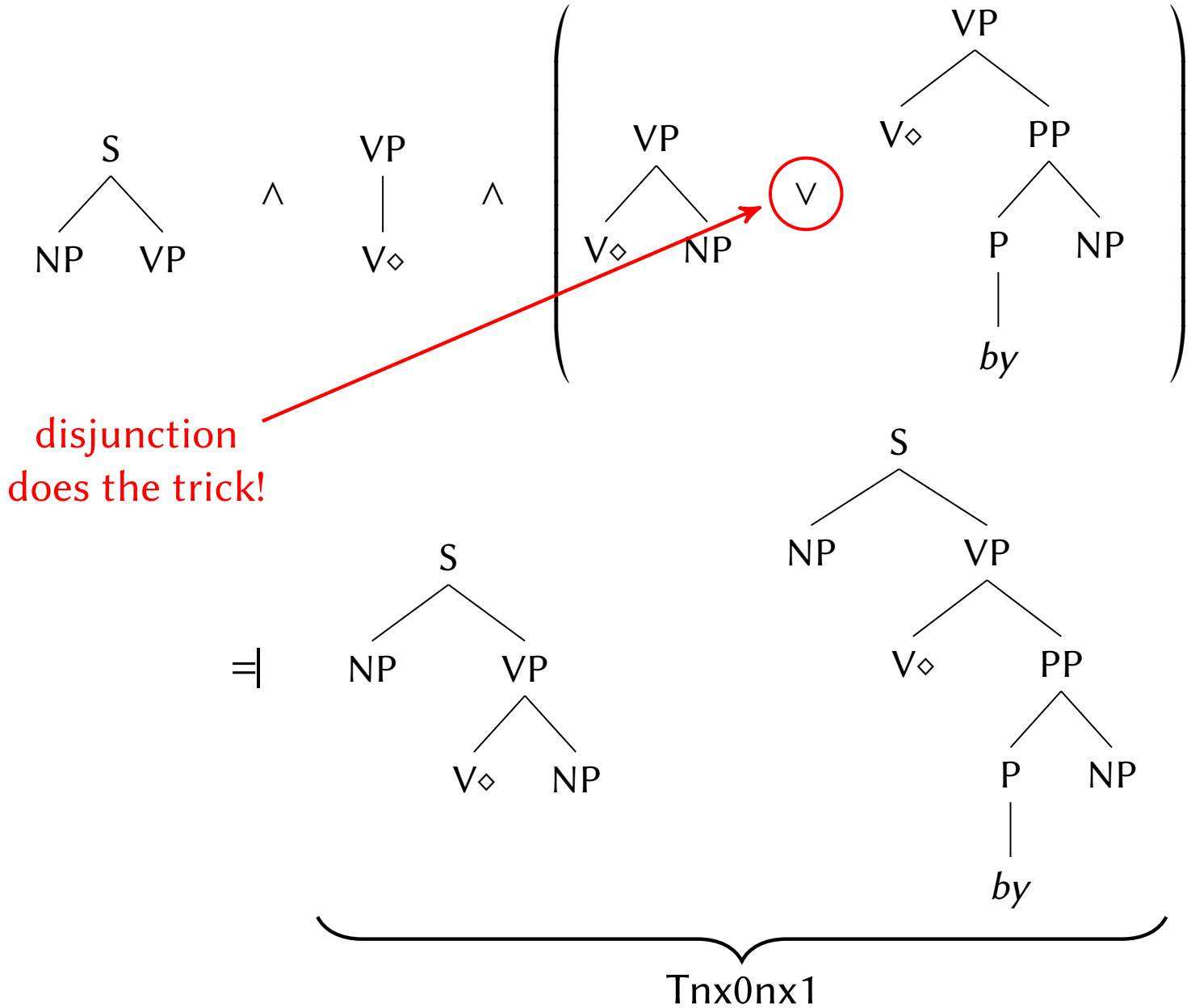
Metagrammars for LTAG: Passivization



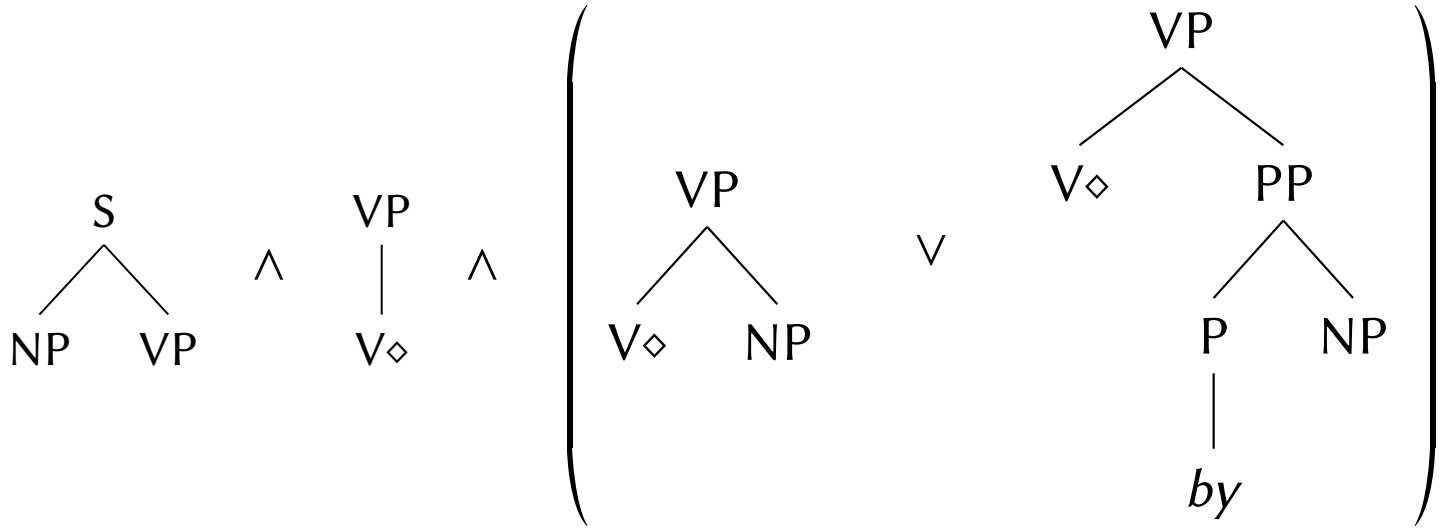
Metagrammars for LTAG: Passivization



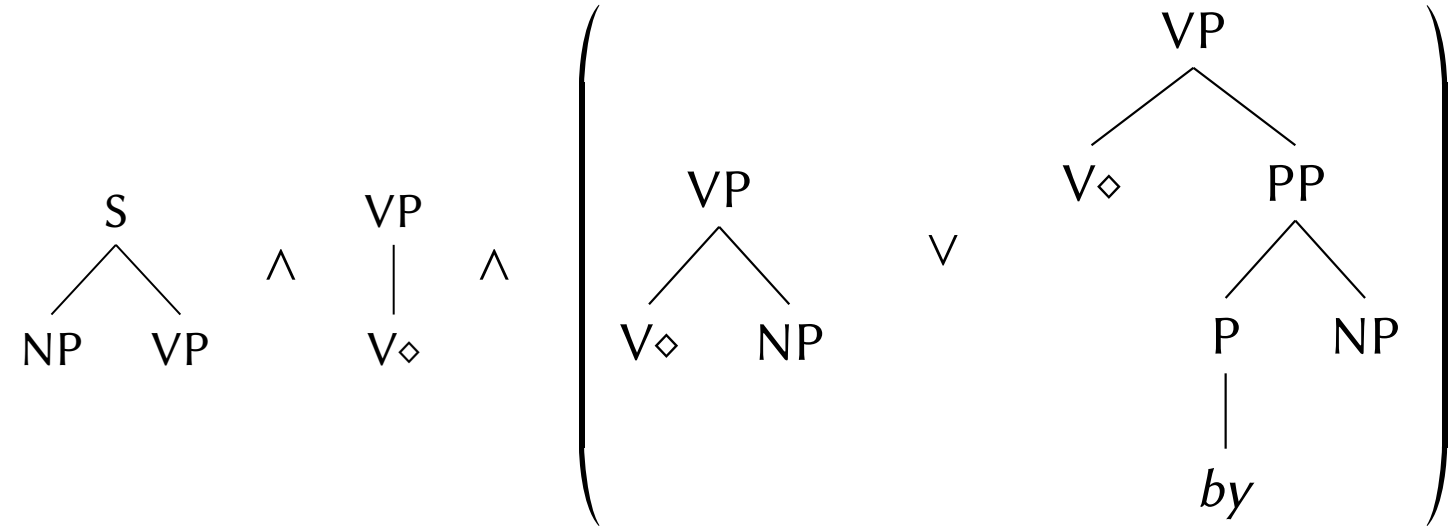
Metagrammars for LTAG: Passivization



Metagrammar for LTAG: Classes

Tnx0Vnx1: 

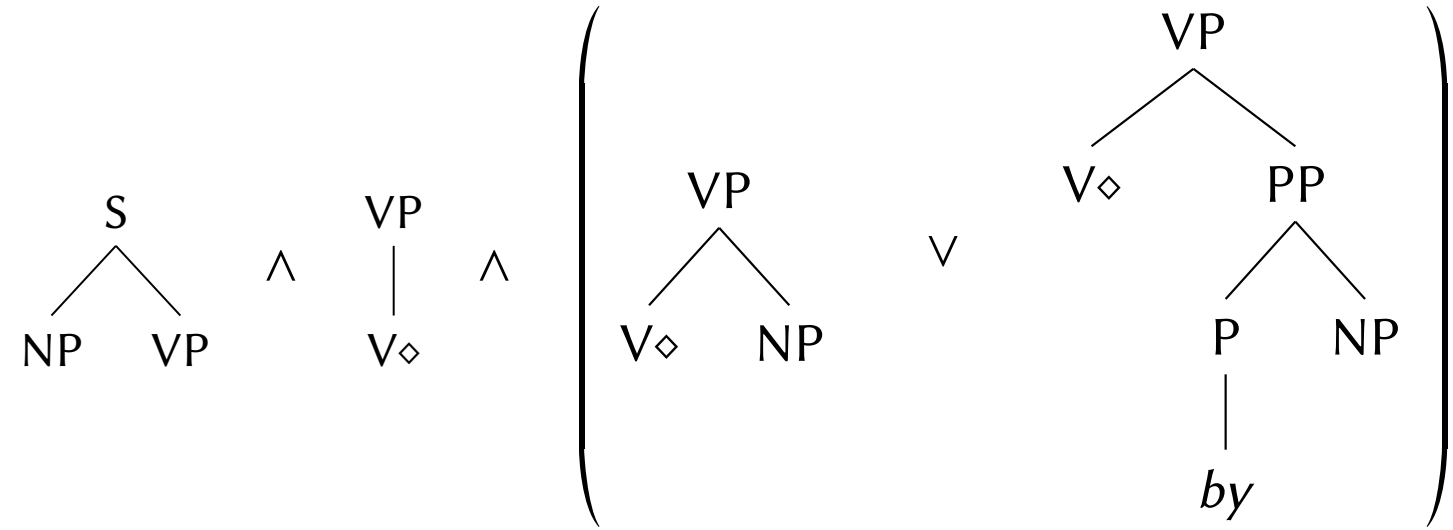
Metagrammar for LTAG: Classes

Tnx0Vnx1: 

Tnx0Vnx1: Subject ∧ VerbProjection ∧ (Object ∨ by-Phrase)

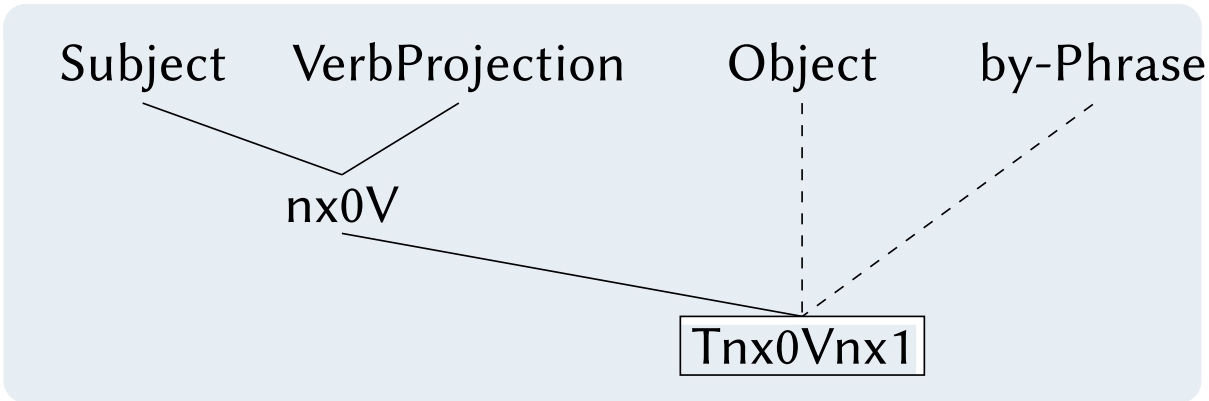
Tnx0Vnx1: nx0V ∧ (Object ∨ by-Phrase)

Metagrammar for LTAG: Classes

Tnx0Vnx1: 

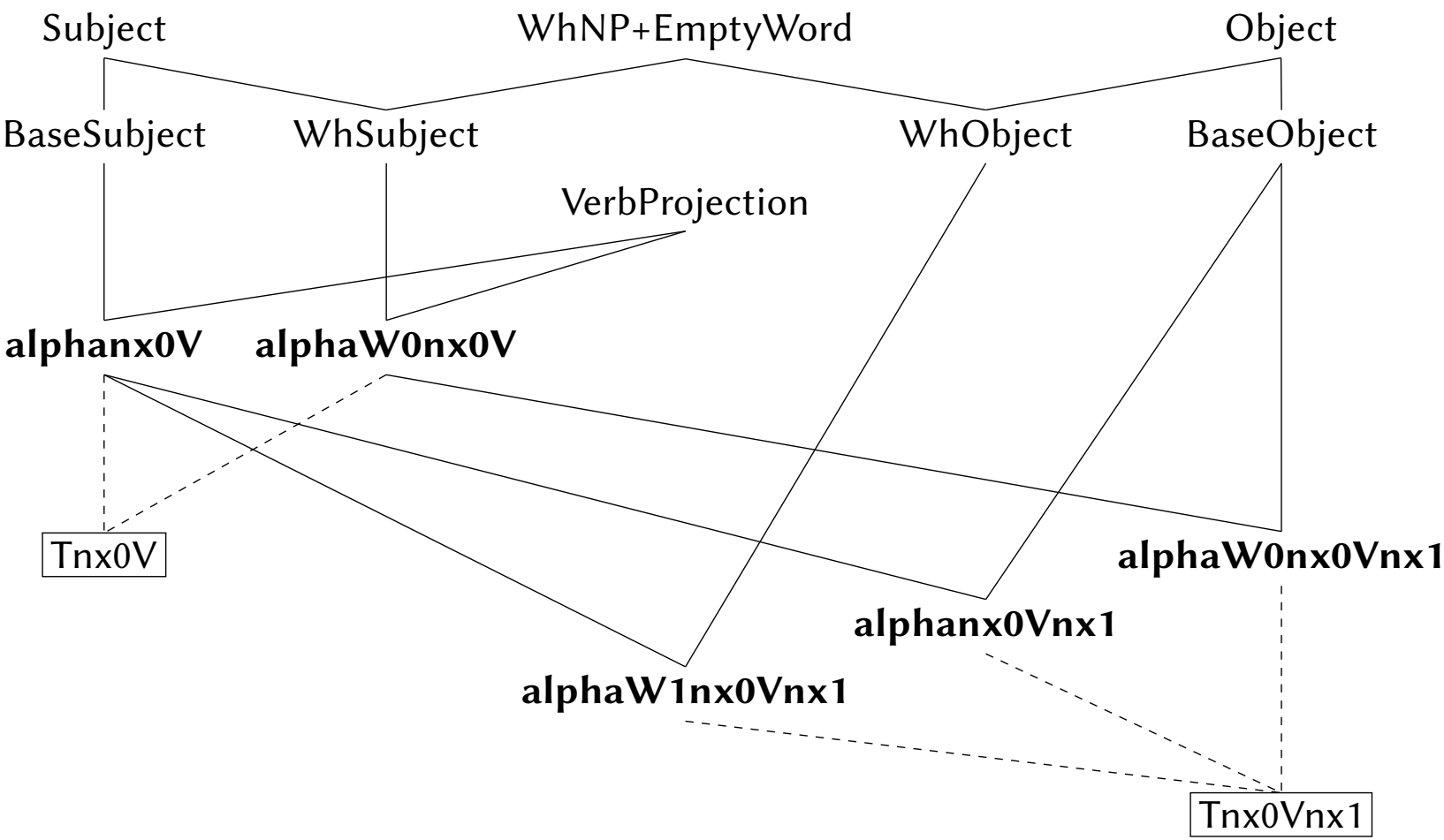
Tnx0Vnx1: Subject ∧ VerbProjection ∧ (Object ∨ by-Phrase)

Tnx0Vnx1: nx0V ∧ (Object ∨ by-Phrase)



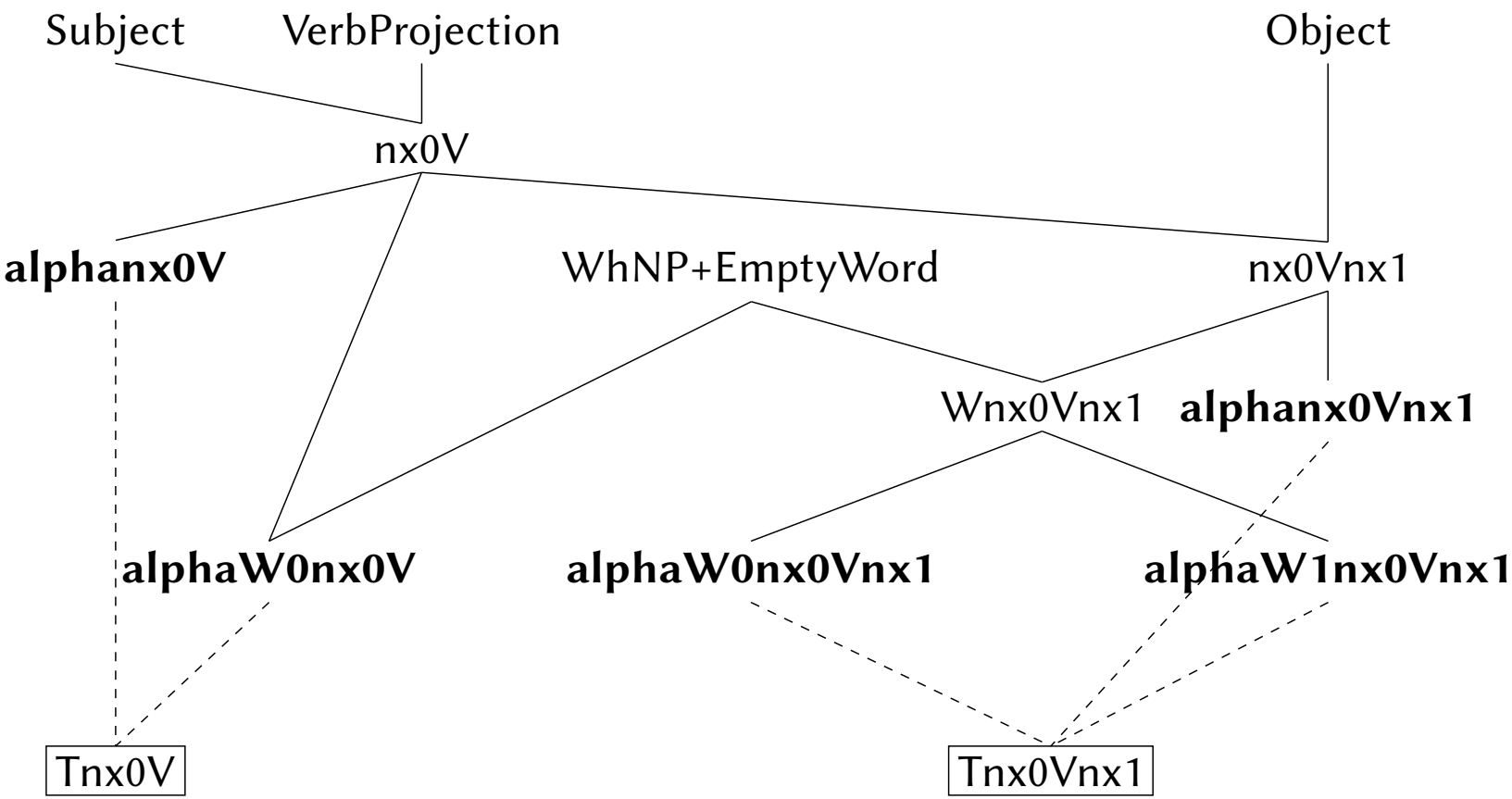
Metagrammar for LTAG: Class hierarchies

There are very many possible class hierarchies …



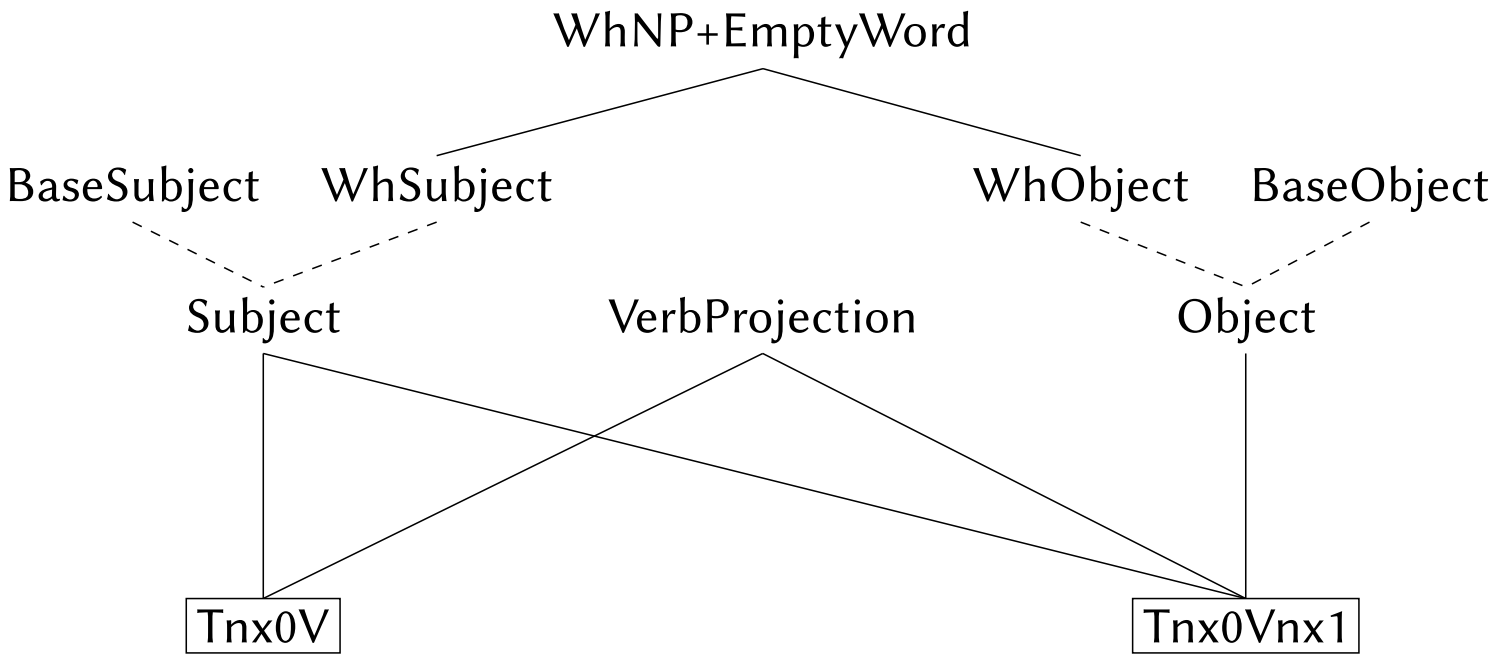
Metagrammar for LTAG: Class hierarchies

There are very many possible class hierarchies …



Metagrammar for LTAG: Class hierarchies

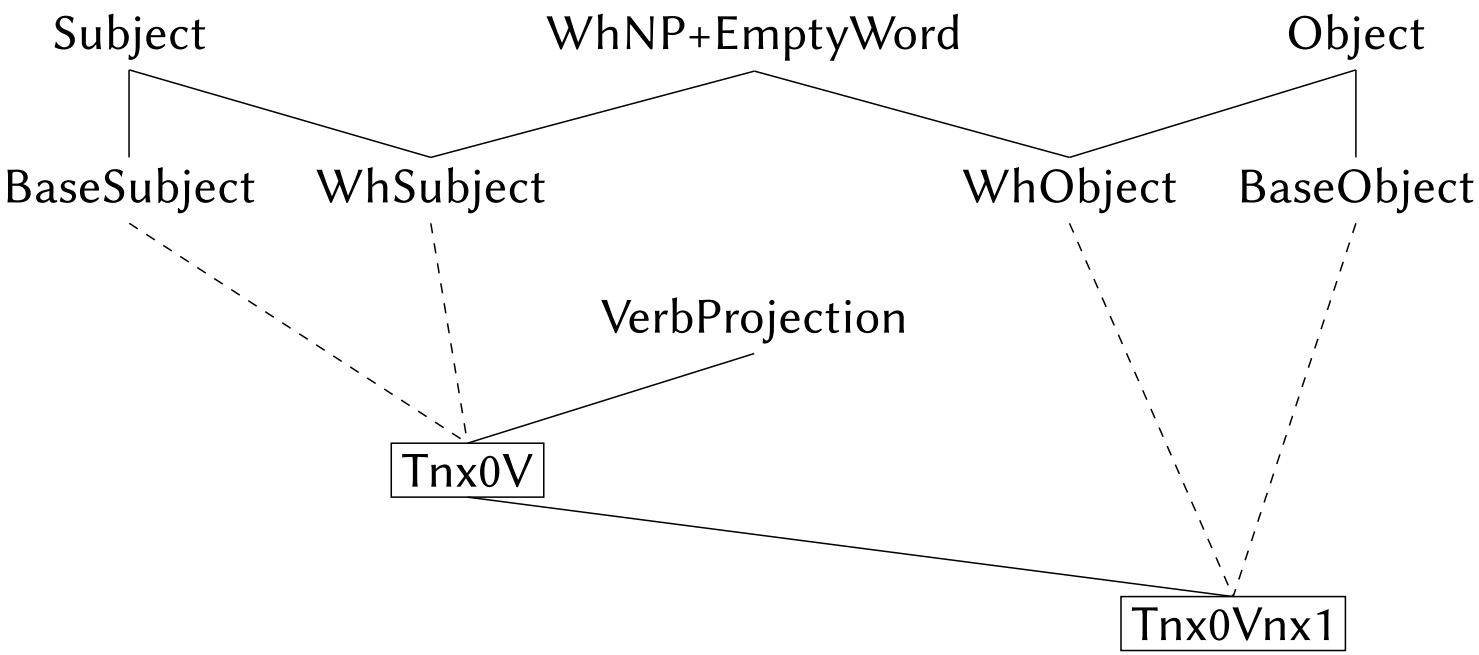
There are very many possible class hierarchies …



[1]

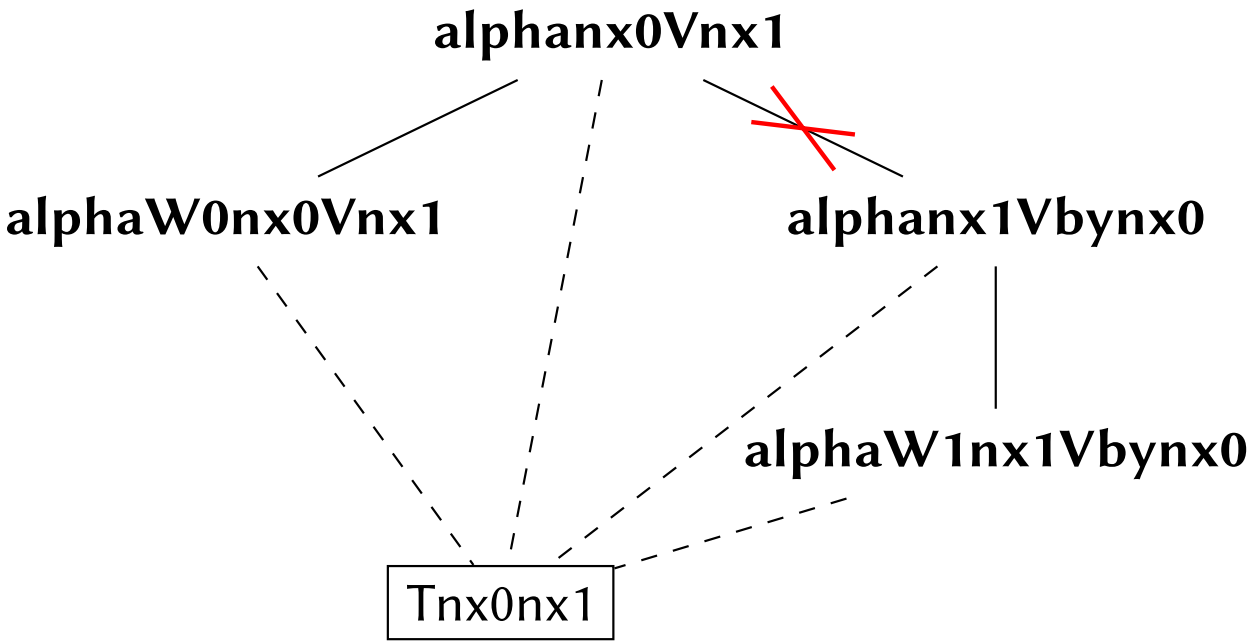
Metagrammar for LTAG: Class hierarchies

There are very many possible class hierarchies …

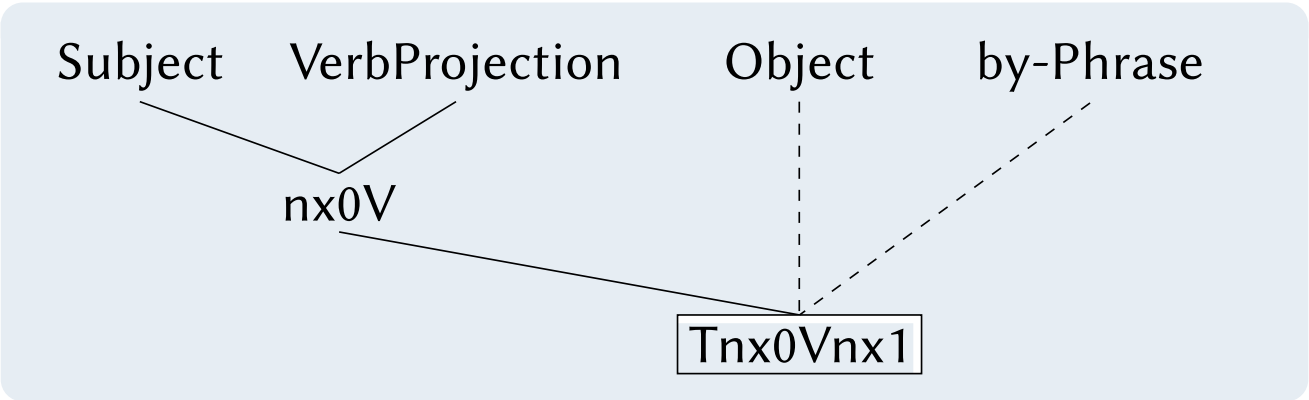


Metagrammar for LTAG: Class hierarchies

…but not everything is possible:

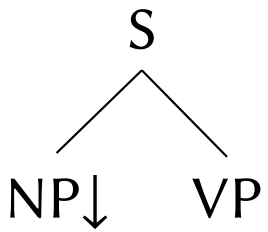


Let's play with XMG

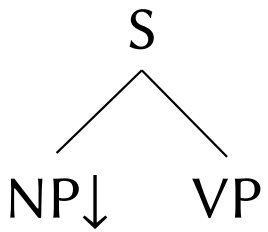


```
1  class alphanx0v
2  import VerbProjection[]
3  declare ?Subj
4  {
5    ?Subj = Subject[];
6    ?Subj.?VP = ?VP
7  }
```

Let's play with XMG

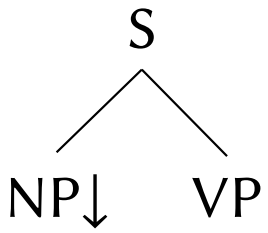


Let's play with XMG



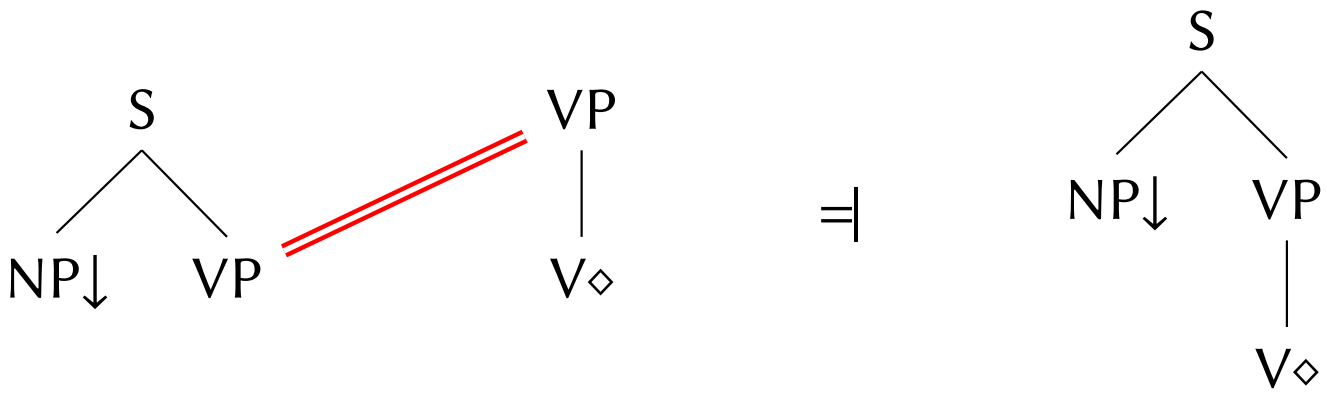

```
1  class Subject
2  export ?S
3  declare ?S ?NP ?VP
4  { <syn>{
5      node ?S [cat=s]{
6          node ?NP (mark=subst) [cat=np]
7          node ?VP [cat=vp]
8          }
9      }
10 }
```

eXtensible Metagrammar (XMG): Example

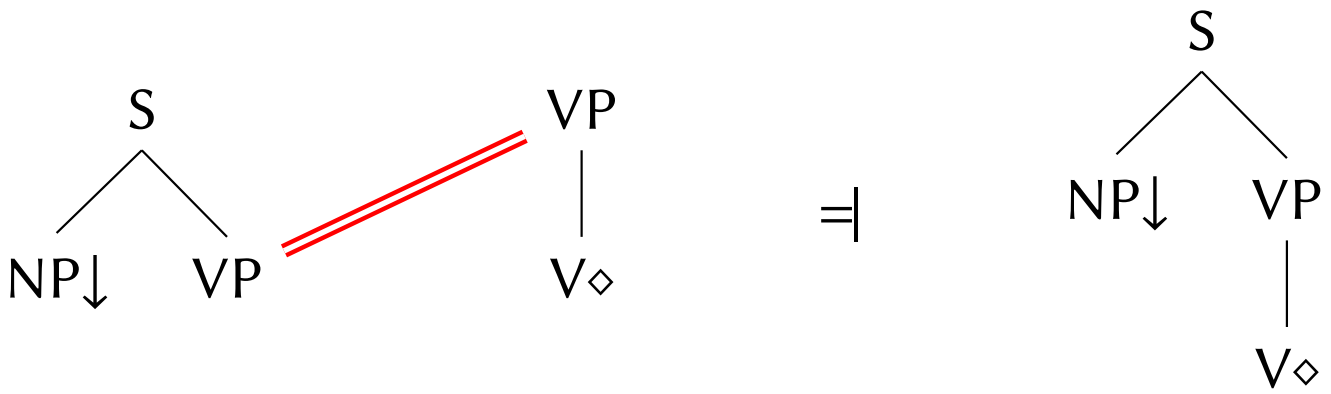



```
1  class Subject
2  export ?S
3  declare ?S ?NP ?VP
4  { <syn>{
5      node ?S [cat=s];
6      node ?NP (mark=subst) [cat=np];
7      node ?VP [cat=vp];
8      ?S -> ?NP;
9      ?S -> ?VP;
10     ?NP >> ?VP
11   }
12 }
```

eXtensible Metagrammar (XMG): Example

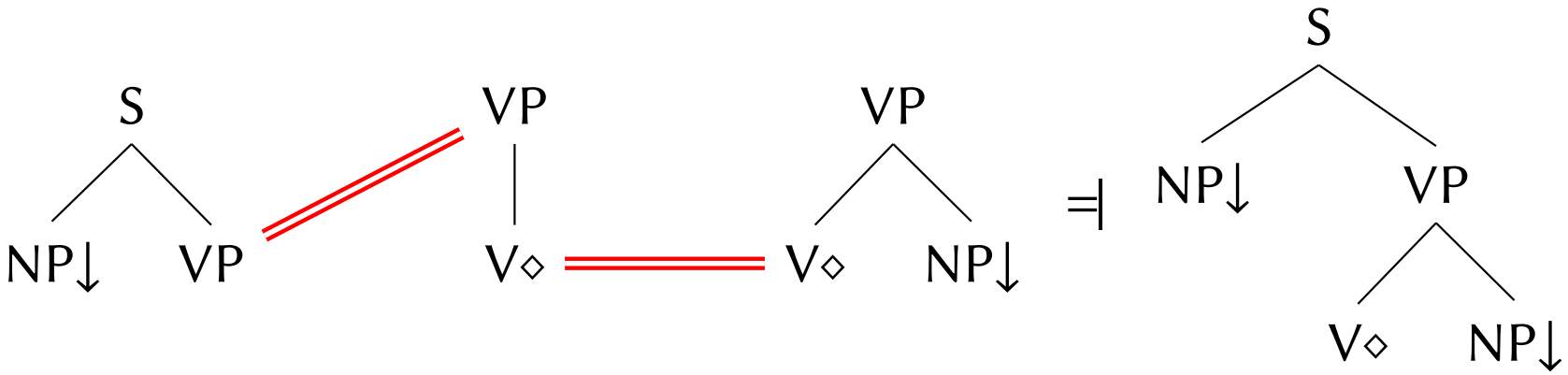


eXtensible Metagrammar (XMG): Example

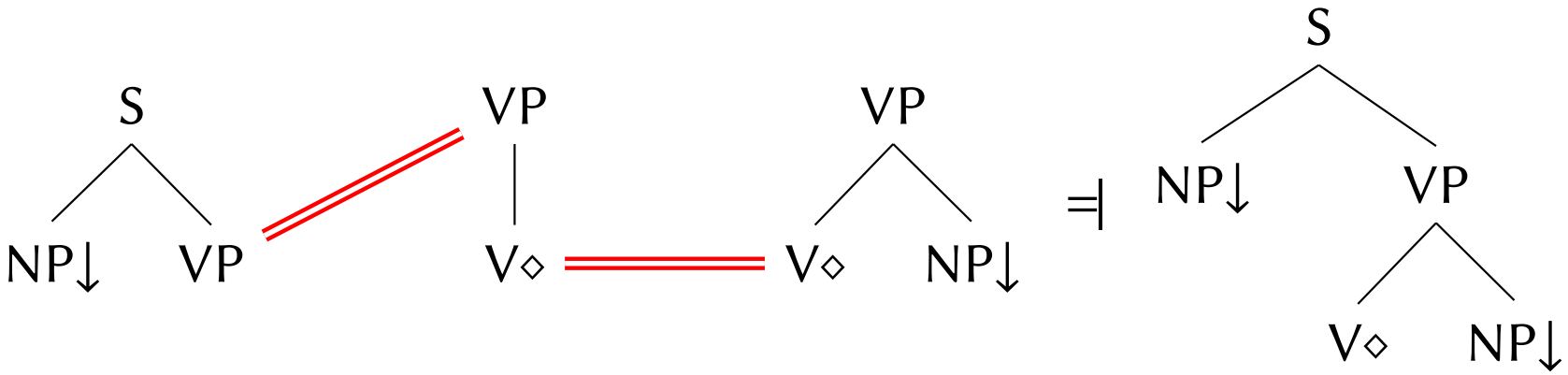



```
1  class alphanx0V
2  import VerbProjection[]
3  declare ?Subj
4  {
5    ?Subj = Subject[];
6    ?Subj.?VP = ?VP
7  }
```

eXtensible Metagrammar (XMG): Example



eXtensible Metagrammar (XMG): Example




```
1  class alphanx0Vnx1
2  import VerbProjection[]
3  declare ?Subj
4  {
5    ?Subj = Subject[];
6    ?Subj.?VP = ?VP
7    ?Obj = Object[];
8    ?Obj.?V = ?V
9  }
```

eXtensible Metagrammar (XMG): Further examples

- tree families
- agreement, case marking
- passive
- extraction

Adjacent topics

- LTAG + Metagrammar = Constructionist Grammar
- grammar induction from treebanks[5,6,11,13]
- automatic metagrammar factorization of templates

Tomorrow

Mon: introduction to grammar engineering and XMG

Tue: implementing syntax with XMG

Wed: implementing semantics with XMG

Thu: parsing implemented grammars with TuLiPA

Fri: conclusion

[1] Alahverdzhieva, Katya. 2008. *XTAG using XMG. A core Tree-Adjoining Grammar for English*. University of Nancy 2 / University of Saarland Master's Thesis. http://homepages.inf.ed.ac.uk/s0896251/pubs/msc-sb2008.pdf.

[2] Becker, Tilman. 1994. *Hytag: a new type of tree adjoining grammars for hybrid syntactic representations of free word order languages*. Universität des Saarlandes dissertation. http://www.dfki.de/~becker/becker.diss.ps.gz.

[3] Becker, Tilman. 2000. Patterns in metarules for TAG. In Anne Abeillé & Owen Rambow (eds.), *Tree Adjoining Grammars: Formalisms, linguistic analyses and processing* (CSLI Lecture Notes 107), 331–342. Stanford, CA: CSLI Publications.

[4] Candito, Marie-Hélène. 1996. A principle-based hierarchical representation of LTAGs. In *Proceedings of the 16th international Conference on Computational Linguistics (COLING 96)*. Copenhagen. http://aclweb.org/anthology-new/C/C96/C96-1034.pdf.

[5] Chen, John, Srinivas Bangalore & K. Vijay-Shanker. 2006. Automated extraction of Tree-Adjoining Grammars from treebanks. *Natural Language Engineering* 12. 251–299.

[6] Chiang, David. 2000. Statistical parsing with an automatically-extracted Tree Adjoining Grammar. In *Proceedings of the 38th annual meeting of the Association for Computational Linguistics*, 456–463. Hong Kong.

[7] Crabbé, Benoît. 2005. *Représentation informatique de grammaires d'arbres fortement lexicalisées: Le cas de la grammaire d'arbres adjoints*. Université Nancy 2 dissertation.

[8] Crabbé, Benoit, Denys Duchier, Claire Gardent, Joseph Le Roux & Yannick Parmentier. 2013. XMG: eXtensible MetaGrammar. *Computational Linguistics* 39(3). 1–66. http://hal.archives-ouvertes.fr/hal-00768224/en/.

[9] Dowty, David R. 1979. *Word meaning and Montague Grammar*. Reprinted 1991 by Kluwer Academic Publishers. Dordrecht: D. Reidel Publishing Company.

[10] Gazdar, Gerald. 1981. Unbounded dependencies and coordinated structure. *Linguistic Inquiry* 12. 155–182.

[11] Kaeshammer, Miriam & Vera Demberg. 2012. German and English treebanks and lexica for Tree-Adjoining Grammars. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk & Stelios Piperidis (eds.), *Proceedings of the eighth international Conference on Language Resources and Evaluation (LREC'12)*. Istanbul, Turkey: European Language Resources Association (ELRA).

[12] Prolo, Carlos A. 2002. Generating the XTAG English grammar using metarules. In *Proceedings of the 19th international Conference on Computational Linguistics (COLING 2002)*, 814–820. Taipei. Taiwan.

[13] Xia, Fei. 2001. *Automatic grammar generation from two different perspectives*. University of Pennsylvania dissertation. `http://faculty.washington.edu/fxia/papers_from_penn/thesis.pdf`.

[14] XTAG Research Group. 2001. *A Lexicalized Tree Adjoining Grammar for English*. Tech. rep. Philadelphia, PA: Institute for Research in Cognitive Science, University of Pennsylvania.