XMG and Multi Word Expressions

Simon Petitjean

CRC 991, Heinrich-Heine-Universität, Germany

PARSEME 7th general meeting

Plan

1. Metagrammars, XMG, XMG2
2. The metagrammatical language
3. Multi Word Expressions with XMG

Introduction

- Different tasks of NLP, using different types of resources (grammars, lexicons, dictionnaries, . . .)
- Manual description by an expert: time consuming
- Automatic methods: need a corpus
- Precise resources, with easier development and maintenance: semi automatic methods
- MetaGrammatical Approach ([Candito, 1996]): linguistic description of the grammar

Object of the description

- Languages can be described at several levels: syntax, morphology, semantic, prosody, discourse, . . .
- Different formalisms for each level: Head-driven phrase structure grammar, Lexical Functional Grammar, Tree Adjoining Grammar, Categorial Grammar, Paradigm Function Morphology, Network Morphology, . . .
- eXtensible MetaGrammar (XMG): MetaGrammar compiler initially used to create large scale Tree Adjoining Grammars ([Joshi and Schabes, 1997]) and Interaction Grammars ([Perrier, 2000])

Description tools

- Need to use a description language, adapted to the task
- Chosing a tool (LKB, XLE, . . .): definitive, no way to adapt the description language
- Fit language to linguistic intuition

Slogan

The user should not have to adapt to the tool
The tool should adapt to the user

XMG ([Crabbé et al., 2013])

Ambition

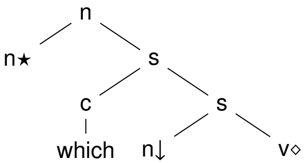
- Arbitrarily many levels of linguistic description (syntax, semantics, . . .): dimensions
- Affect a Domain Specific Language (DSL) to each one of these levels

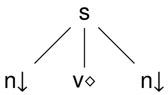
Methodology

- Declarative definition of rule fragments (classes)
- Combination of rule fragments with logical operators

XMG: example

(1) What do we want to generate? 2 trees (amongst others) for transitive verbs

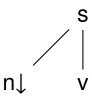




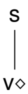
XMG: example

(2) Define fragments:

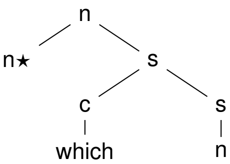
CanSubj



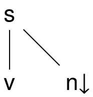
Active

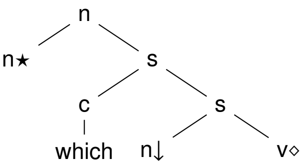


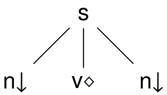
RelObj



CanObj

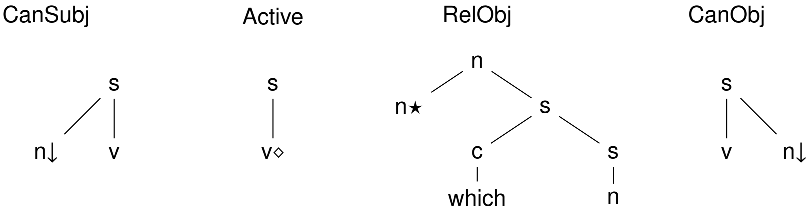






XMG: example

(2) Define fragments:



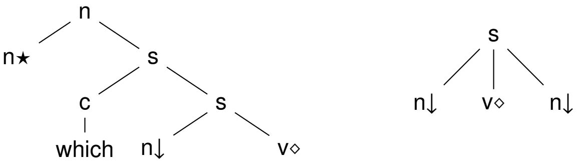
CanSubj

Active

RelObj

CanObj

(3) Combine them: CanSubj ∧ Active ∧ (RelObj ∨ CanObj)



Tools

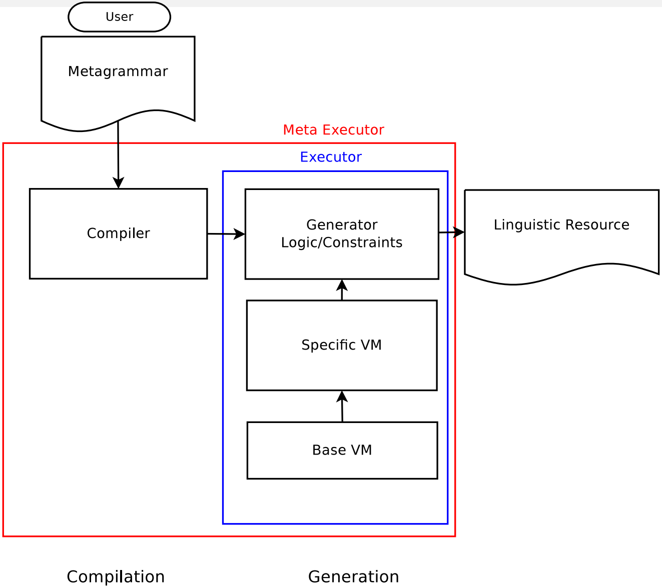
XMG1

- eXtensible (?) Metagrammar
- Only 3 dimensions

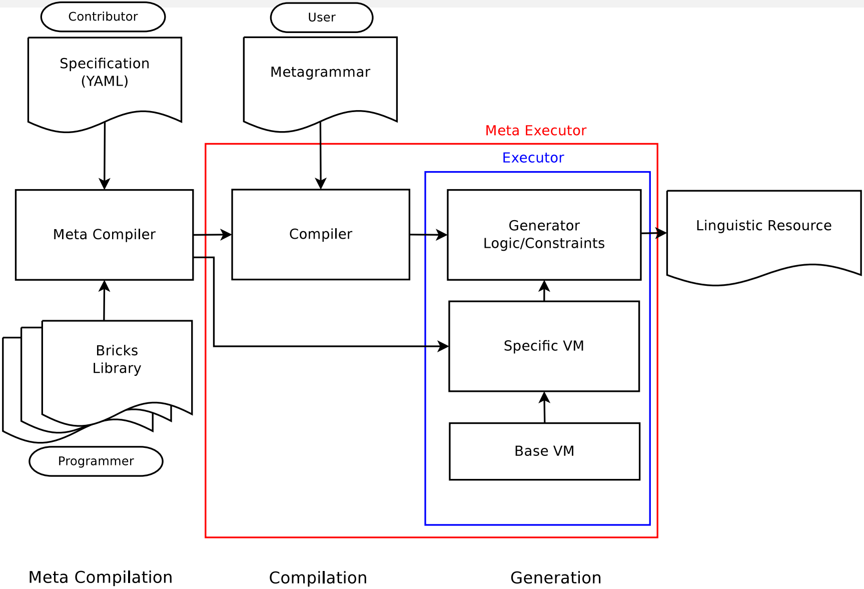
XMG2

- Arbitrarily many dimensions, with DSLs
- Modular assembly of DSL, using bricks, like a LEGO^{TM} construction
- Methodology to generate a whole processing chain

XMG1: Architecture



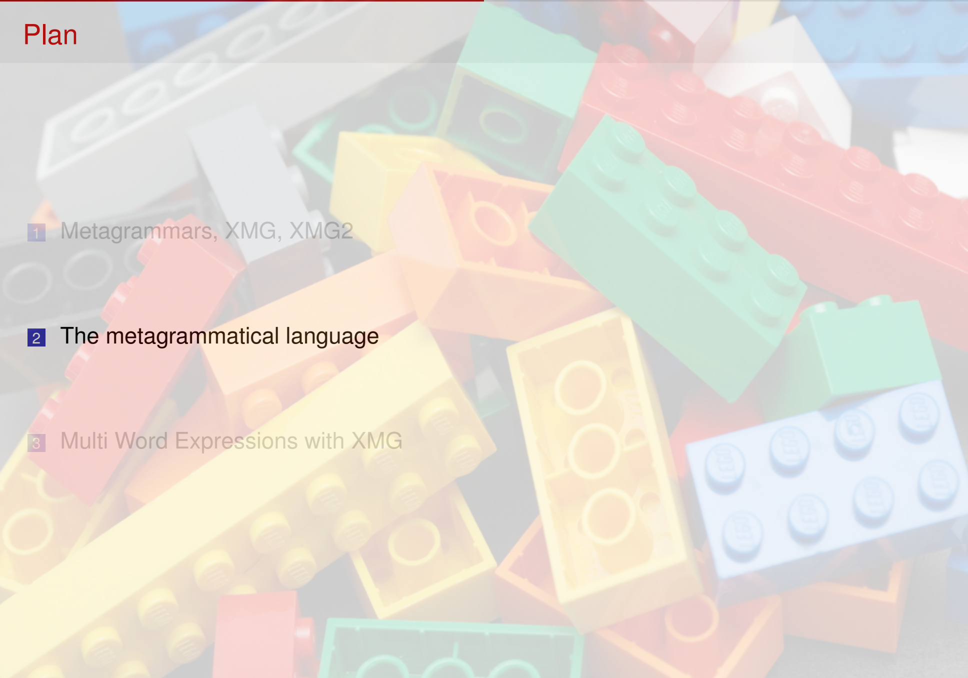
XMG2: Architecture



XMG2: what is new?

- The website (tools and documentation): `http://xmg.phil.hhu.de/`
- 3 options: installation, virtualization or web interface
- New online viewer for the grammars
- New compilers including new dimensions: morphology, frame semantics

Plan



1 Metagrammars, XMG, XMG2

2 The metagrammatical language

3 Multi Word Expressions with XMG

The control language

XMG descriptions:

- Associate a content to an identifier (abstraction)
- Describe structures inside dimensions, with dedicated languages
- Use other abstractions (classes)
- Combining contents in a disjunctive or a conjunctive way

$$\begin{aligned} \mathit{Class} &:= \mathit{Name} \rightarrow \mathit{Content} \\ \mathit{Content} &:= \langle \mathit{Dimension} \rangle \{ \mathit{Description} \} \mid \mathit{Name} \mid \\ &\quad\ \mathit{Content} \vee \mathit{Content} \mid \mathit{Content} \wedge \mathit{Content} \end{aligned}$$

Using classes

Classes allow to:

- Control the scope of variables
- Make (parametrized) abstractions

```
class nx0Vnx1
export ?S ?NP_Subj ?VP ?V ?NP_Obj
declare ?S ?NP_Subj ?VP ?V ?NP_Obj ?X0 ?X1
```

Using classes

Classes allow to:

- Control the scope of variables
- Make (parametrized) abstractions

```
class nx0Vnx1
export ?S ?NP_Subj ?VP ?V ?NP_Obj
declare ?S ?NP_Subj ?VP ?V ?NP_Obj ?X0 ?X1
```

```
class kicked_the_bucket
import nx0Vnx1[]
declare ?X0 ?X1
```

Describing trees

The <**syn**> dimension

- Declaring nodes: keyword **node**, optional node variable, optional features and properties
  node ?S [cat=s]

- Expressing constraints between nodes: dominance operators (`->`, `->+`, `->*`) and precedence operators (`>>`, `>>+`, `>>*`)

```
node ?S [cat=s];
node ?NP (mark=subst) [cat=np];
node VP [cat=vp];
?S -> ?NP;
?S -> ?VP;
?NP >> ?VP
```

Alternative syntax: bracket notation

The <syn> dimension

- Declaring nodes: optional node variable, optional features and properties
- Expressing dominance and precedence constraints thanks to bracketing
- Lexical nodes declared thanks to double quotes

```
?NP_Obj {
  ?D [cat=det] "the"
  ?N [cat=n, e=?X0] "bucket"
}
```

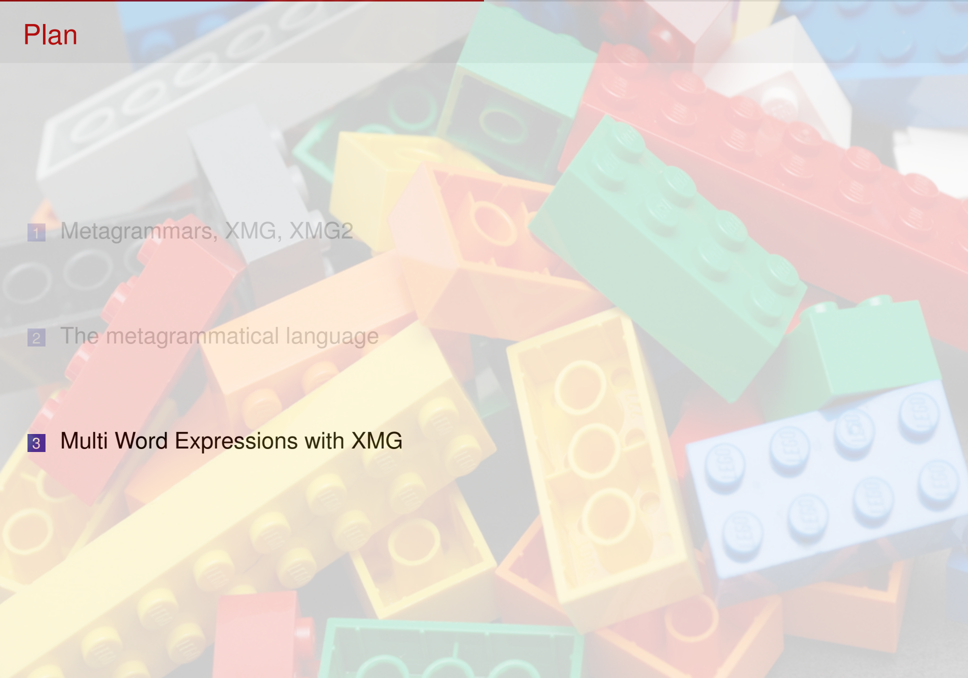
Using dimensions

Contributing descriptions

- Descriptions (constraints) are accumulated into dimensions
- Every dimension is associated to a solver (sometimes identity)
- **<syn>**: a tree solver generates all minimal models

```
<syn>{
    node ?V [e=?X0];
    node ?K (mark=lex) [cat=kicked, e=?X0];
    ?V -> ?K;
    ?NP_Obj {
      ?D [cat=det] "the"
      ?N [cat=n, e=?X0] "bucket"
    }
}
```

Plan



1 Metagrammars, XMG, XMG2

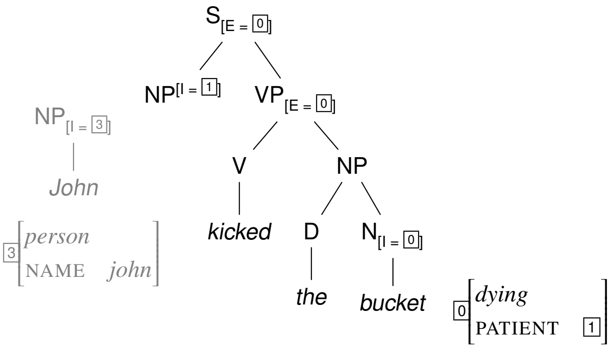
2 The metagrammatical language

3 Multi Word Expressions with XMG

XMG in the previous PARSEME meetings

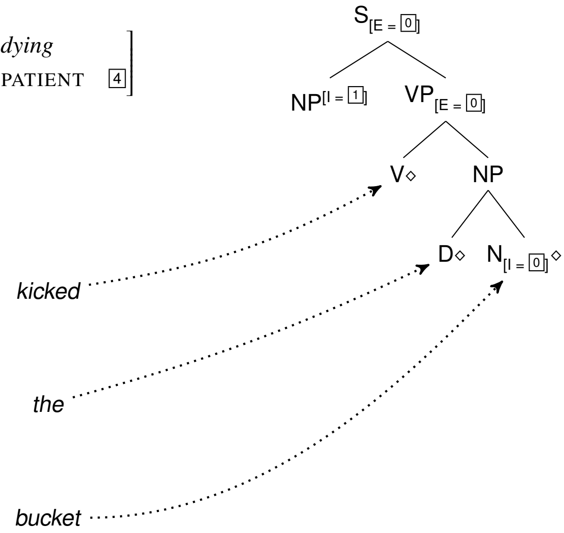
- Internal vs external encoding in Tree Adjoining Grammars (with Timm Lichte)
- Comparison with other encoding tools (with Timm Lichte, Yannick Parmentier, Agata Savary and Jakub Waszczuk)

Internal vs external encoding in Tree Adjoining Grammars

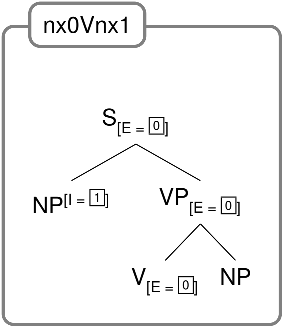


External anchoring

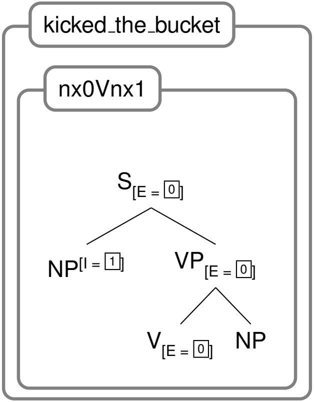
$$\boxed{3}\begin{bmatrix} \textit{dying} & \\ \text{PATIENT} & \boxed{4} \end{bmatrix}$$



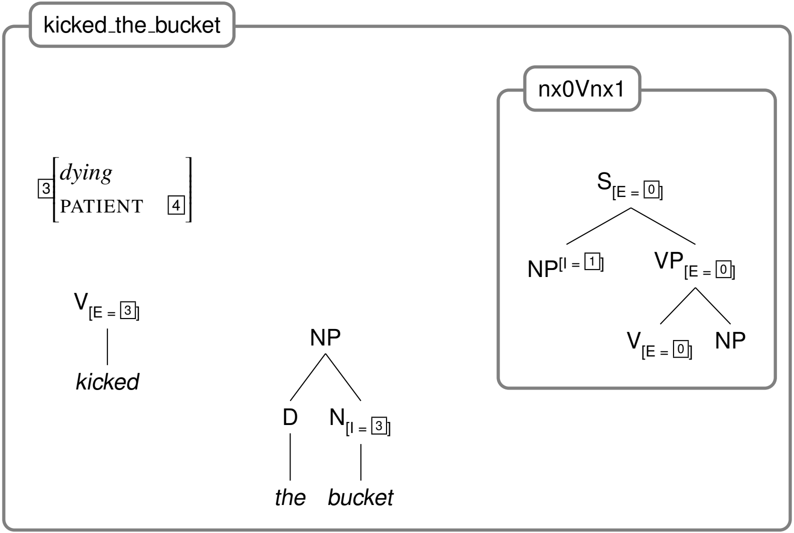
Internal anchoring



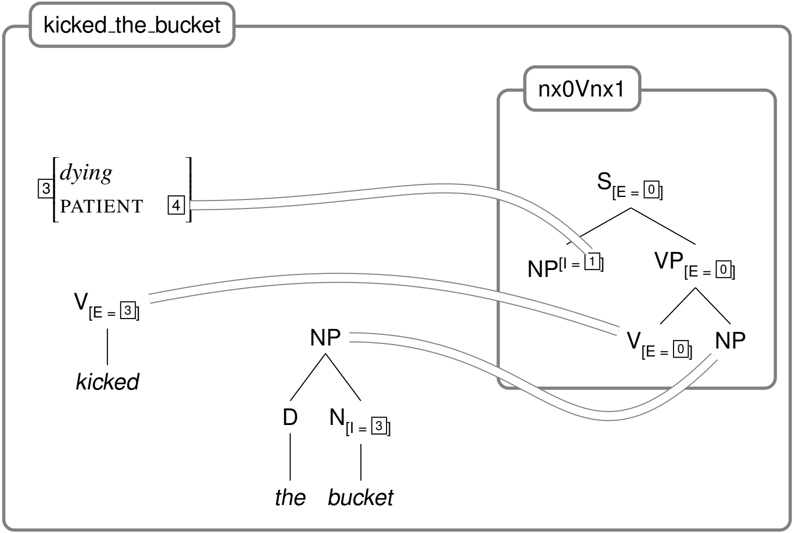
Internal anchoring



Internal anchoring



Internal anchoring



Example: implementation

```
class nx0Vnx1
export ?S ?NP_Subj ?VP ?V ?NP_Obj
declare ?S ?NP_Subj ?VP ?V ?NP_Obj
    ?X0 ?X1
{
  <syn>{
     ?S [cat=s, e=?X0] {
       ?NP_Subj [cat=np,i=?X1]
       ?VP [cat=vp, e=?X0] {
         ?V [cat=v, e=?X0]
         ?NP_Obj [cat=np] }}
  }
}
```

```
class kicked_the_bucket
import nx0Vnx1[]
declare ?X3 ?X4
{
  <syn>{
     ?NP_Subj [i=?X4];
     ?V [e=?X3] "kicked";
     ?NP_Obj [] {
       [cat=det] "the"
       [cat=n, i=?X3] "bucket" }
  }
  ;
  <frame>{
    ?X3[dying,
      patient:?X4]
  }
}
```

DuELME

Dutch Electronic Lexicon of Multiword Expressions [Grégoire, 2010]:

- electronic lexicon comprising roughly 5000 Dutch multi-word expressions
- 141 pattern descriptions [Grégoire, 2007]

Description for *zijn kansen waarnemen* (‘to seize the opportunity’):

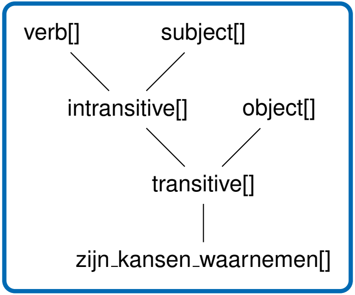
```
% Pattern description
PATTERN_NAME ec1
POS d n v
PATTERN  [.VP [.obj1:NP [.det:D (1) ] [.hd:N (2) ]] [.hd:V (3) ]]

% MWE description
EXPRESSION zijn kansen waarnemen
CL zijn kans[pl] waar_nemen[part]
PATTERN_NAME ec1
```

Implementation with XMG

```
class intransitive
import subject[] verb[]
{ <syn> {
    ?Subj >>+ ?V
} }

class transitive
import intransitive[] object[]
{ <syn> {
    ?Subj >>+ ?Obj;
    ?Obj >>+ ?V
} }

class zijn_kansen_waarnemen
import transitive[]
declare ?NUM ?PERS ?GEND
{ <syn> {
    ?Subj[num=?NUM,pers=?PERS,gend=?GEND];
    ?Obj [] {
      [cat=d,num=pl,possnum=?NUM,pers=?PERS,gend=?GEND] "zijn"
      [cat=n,modifiable=-,num=pl] "kans"};
    ?V[] "waar_nehmen"
} }
```

Implementation with XMG

```
class intransitive
import subject[] verb[]
{ <syn> {
    ?Subj >>+ ?V
} }

class transitive
import intransitive[] object[]
{ <syn> {
    ?Subj >>+ ?Obj;
    ?Obj >>+ ?V
} }

class zijn_kansen_waarnemen
import transitive[]
declare ?NUM ?PERS ?GEND
{ <syn> {
    ?Subj[num=?NUM,pers=?PERS,gend=?GEND];
    ?Obj [] {
      [cat=d,num=pl,possnum=?NUM,pers=?PERS,gend=?GEND] "zijn"
      [cat=n,modifiable=-,num=pl] "kans"};
    ?V[] "waar_nehmen"
} }
```



verb[]
subject[]
intransitive[]
object[]
transitive[]
zijn_kansen_waarnemen[]

Walenty

Walenty:

- Polish large-scale valence dictionary offering a rather expressive formalism [Przepiórkowski et al., pear]
- includes an elaborate phraseological component [Przepiórkowski et al., 2014]

Example of entry:

(1) dobrze [KOMUŚ] z oczu patrzy
well someone.DAT from eyes.GEN looks
'Someone looks like a good person.'

Walenty description:

```
patrzeć: np(dat)+advp(misc)+lex(prepnp(z,gen),pl,'oko',natr)
```

Implementation with XMG

```
class impers_intransitive
export ?VP ?V
declare ?VP ?V
{ <syn>{
    ?VP [cat=vp] { ?V [cat=v,pers=3,num=pl] }
} }

class impers_intransitive_IndObj_PP
import impers_intransitive[] indir_object[] prep_compl[]
{ <syn> {
    ?VP -> ?PP;
    ?VP -> ?IndObj
} }

class dobrze_z_oczu_patrzy
import impers_intransitive_IndObj_PP[]    adverb[]
{ <syn> {
    ?AP [] { ?A [] "dobrze"};
    ?PP [] {
      [cat=p,case=gen] "z"
      [cat=np] { [cat=n,num=pl,modifiable=-]  "oko" }};
    ?V "patrzeć";
    ?AP >>+ ?PP;
    ?AP >>+ ?V
} }
```

Conclusion

XMG2

- Arbitrarily many levels of linguistic description (syntax, semantics, . . .): dimensions
- A Domain Specific Language (DSL) for each one of these levels
- Assemble new compilers for new tasks

Methodology

- Create abstractions on rules and combine these abstractions with logical operators
- Class hierarchy, inheritance system
- MWE/TAG: internal anchoring

For any question

- `http://xmg.phil.hhu.de/`
- simon.petitjean@hhu.de

Thank You

Candito, M. (1996).
A Principle-Based Hierarchical Representation of LTAGs.
In *Proceedings of the 16th International Conference on Computational Linguistics (COLING'96)*, volume 1, pages 194–199, Copenhagen, Denmark.

Crabbé, B., Duchier, D., Gardent, C., Le Roux, J., and Parmentier, Y. (2013).
XMG : eXtensible MetaGrammar.
Computational Linguistics, 39(3):591–629.

Grégoire, N. (2007).
MWE Lexicon for Dutch: Overview of pattern descriptions.

Grégoire, N. (2010).
DuELME: A Dutch electronic lexicon of multiword expressions.
Language Resources and Evaluation, 44(1–2):23–39.

Joshi, A. K. and Schabes, Y. (1997).
Tree adjoining grammars.
In Rozenberg, G. and Salomaa, A., editors, *Handbook of Formal Languages*. Springer Verlag, Berlin.

- Perrier, G. (2000).
Interaction Grammars.
In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, volume 2, pages 600–606, Saarbrücken, Germany.

- Przepiórkowski, A., Hajič, J., Hajnicz, E., and Urešová, Z. (To appear).
Phraseology in two Slavic valency dictionaries: Limitations and perspectives.
International Journal of Lexicography.

- Przepiórkowski, A., Hajnicz, E., Patejuk, A., and Woliński, M. (2014).
Extended phraseological information in a valence dictionary for NLP applications.
In *Proceedings of the Workshop on Lexical and Grammatical Resources for Language Processing (LG-LP 2014)*, pages 83–91, Dublin, Ireland.