Putting Control into Language Learning

Herbert Lange[1] and Peter Ljunglöf[1]

[1]Computer Science and Engineering
University of Gothenburg and
Chalmers University of Technology
Gothenburg, Sweden

CNL 2018, Maynooth, Ireland
August 27, 2018

Overview

Introduction

Problem:
Can we build a Language Learning Application that:

- is intuitively usable,
- works with less-resourced languages,
- and provides a high level of reliability?

Idea:
Use grammars

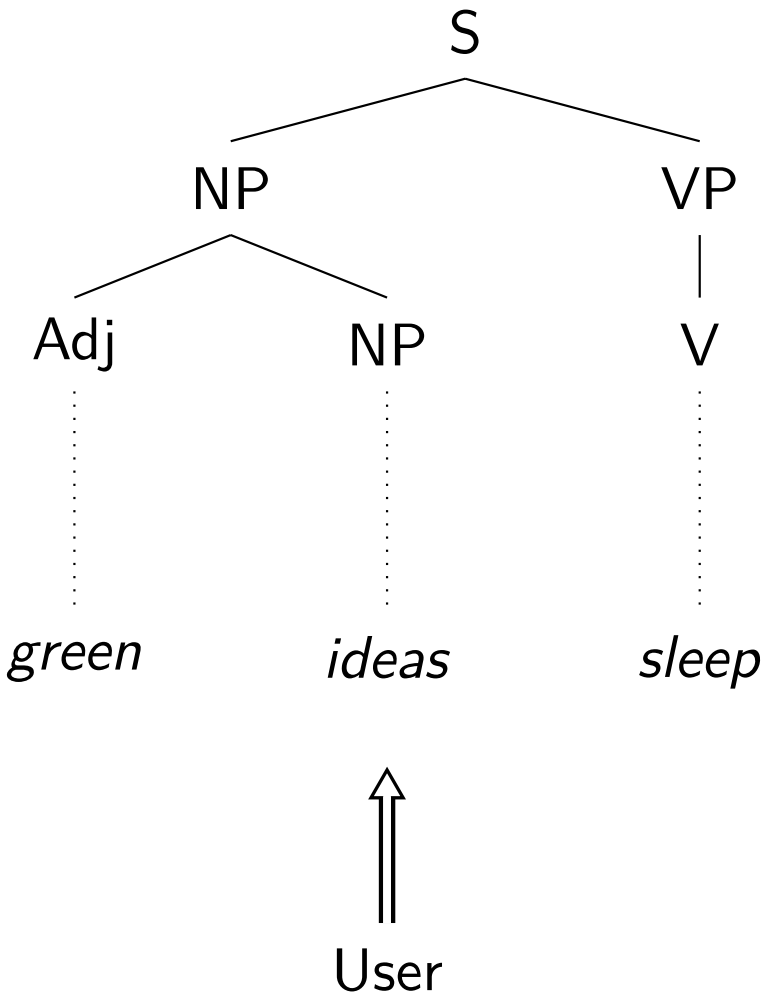
Grammar-Based Text Modification

Based on Ljunglöf (2011):
Maps edit operations on the surface to edit operations on the syntax tree

Example Grammar:

```
S    ::= NP  VP
NP   ::= Adj NP
VP   ::= V
Adj  ::= "colorless" | "green" | "quick" | "brown"
NP   ::= "ideas" | "foxes"
V    ::= "sleep" | "jump"
```

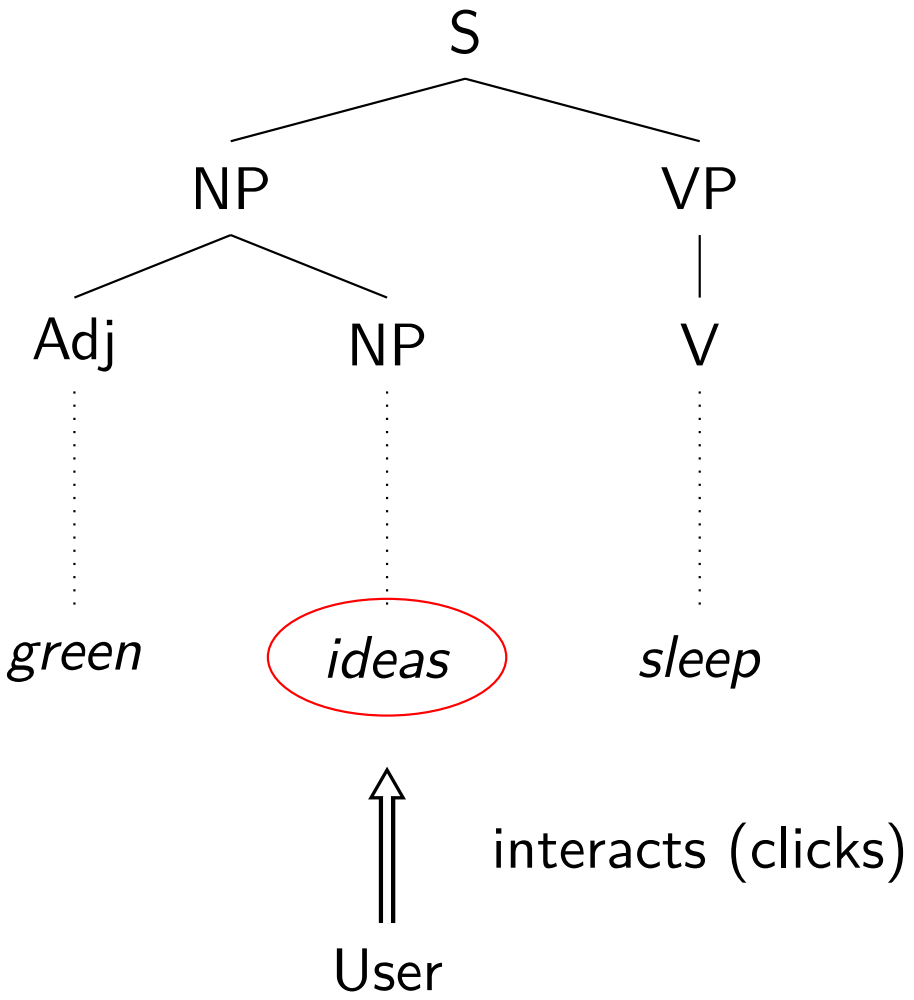
Grammar-Based Text Modification



Syntax Tree

Surface String

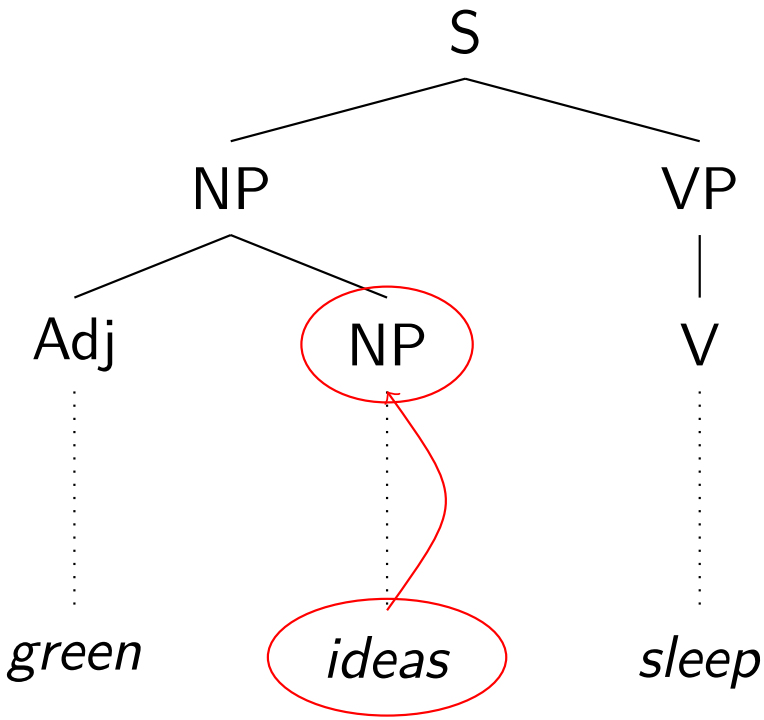
Grammar-Based Text Modification



Syntax Tree

Surface String

Grammar-Based Text Modification



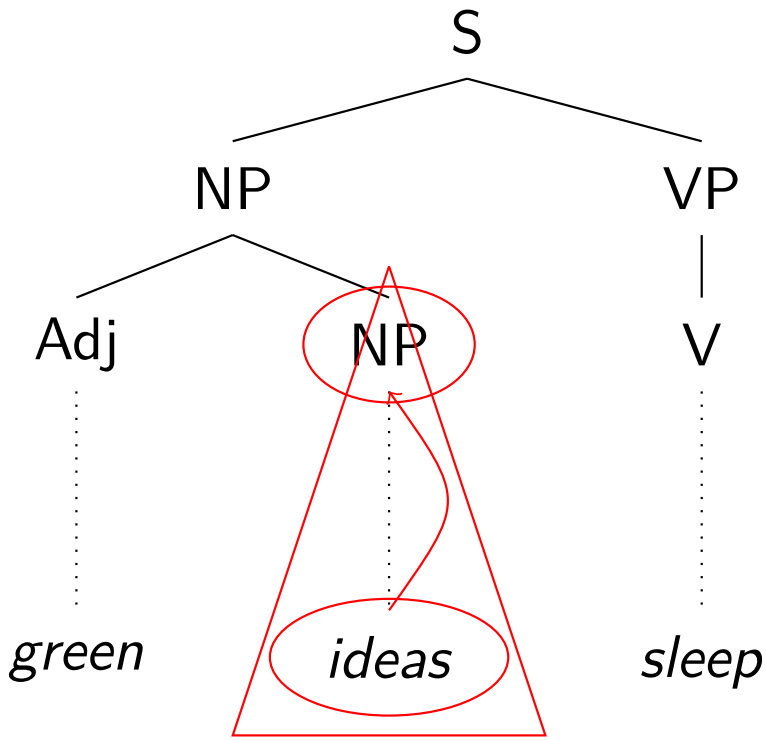
Syntax Tree

Map from Surface to Node

Surface String

User

Grammar-Based Text Modification



Syntax Tree

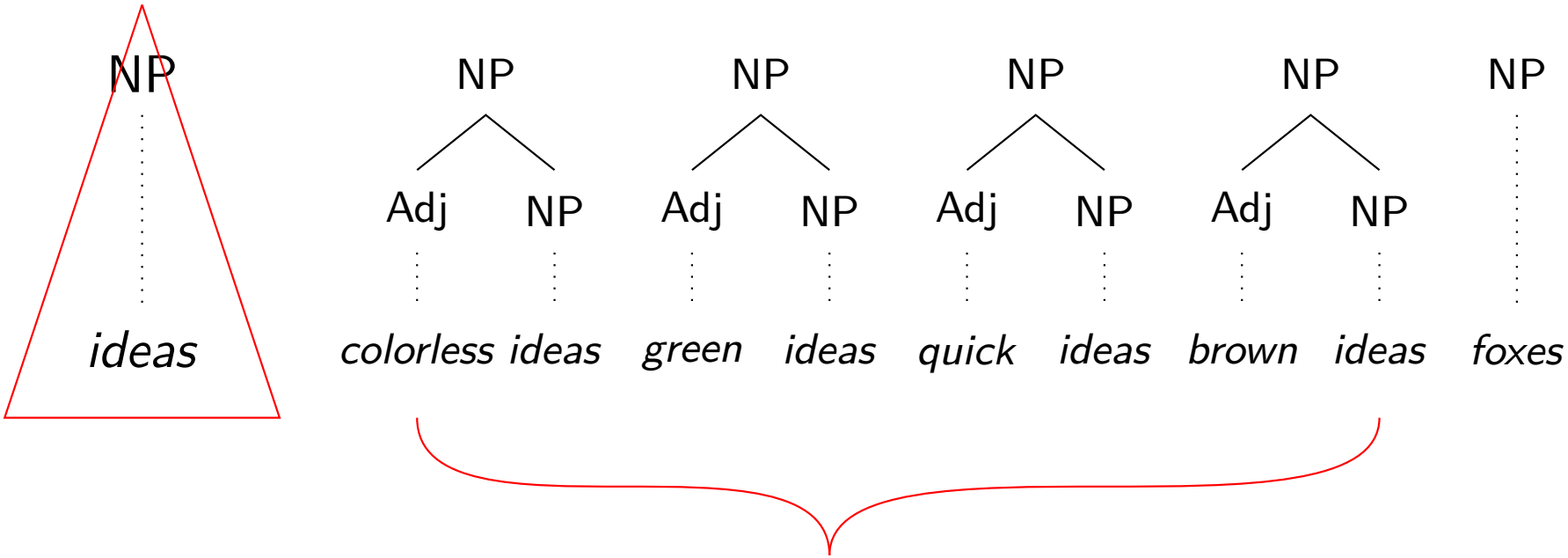
Extract Subtree covered by Node

Surface String

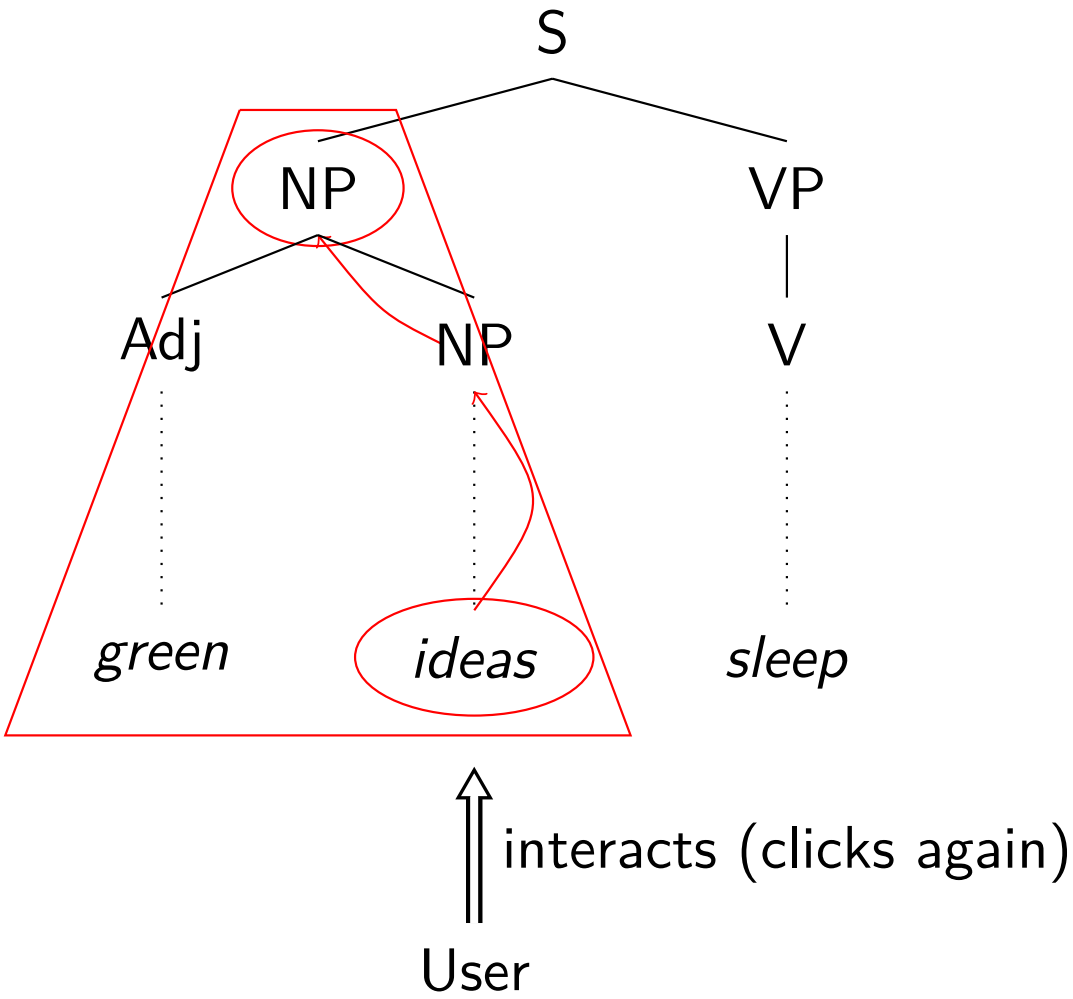
Generate similar Trees

User

Grammar-Based Text Modification



Grammar-Based Text Modification



Syntax Tree

Walking up the Tree

Surface String

Grammar Creations

3 steps from a textbook lesson to a lesson grammar:

- adopt vocabulary
- convert sentences to syntax trees
- extract grammar from syntax trees

Grammar Creation - Step 1

Step 1: Vocabulary list (e.g. (Ehrling, 2015, p. 11))

```
latinsk ord                 svensk översättning

imperium -i (n)             rike, makt;befälsrätt
Romanus, -a, -um            romersk
magnus, -a, -um (adj)       stor
esse                        att vara (oregelbunden verb)
est                         (han/hon/den/det) är
[...]
```

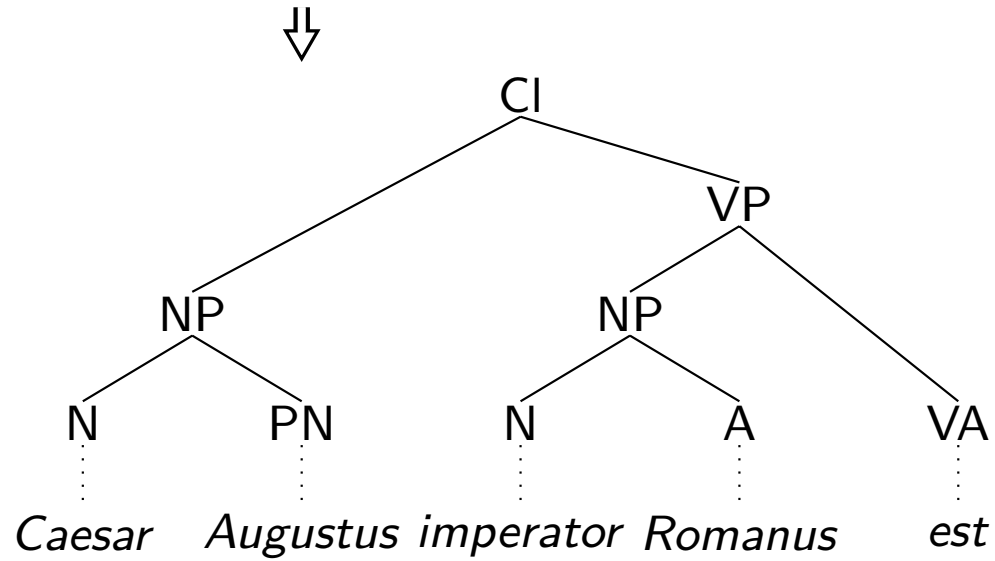
```
fun
  copula_VA : VA ;
  copula_V2 : V2 ;
  -- Vocabulary p11
  imperium_N : N ;
  Romanus_A : A ;
  magnus_A : A ;
  [...]
}
```

Grammar Creation - Step 2

Step 2: Text Fragment (e.g. (Ehrling, 2015, p. 10))

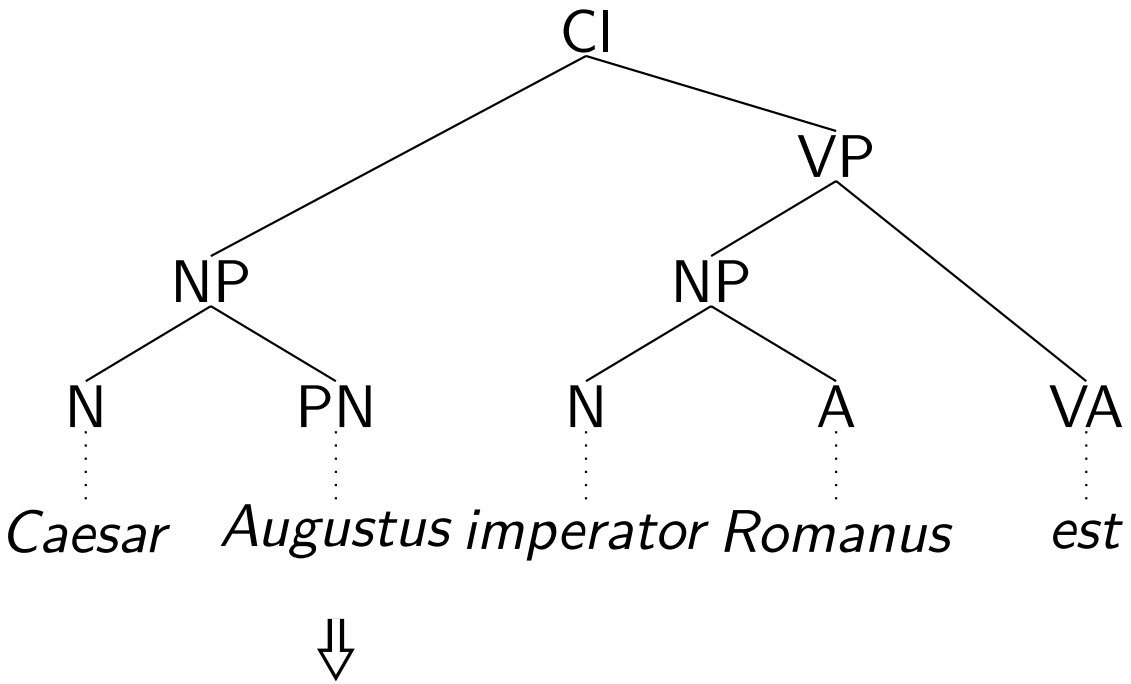
```
  Prima scripta Latina

[...] Imperium imperatorem habet. Imperator imperium tenet. Caesar
Augustus imperator Romanus est. Imperium Romanum tenet. Multas
civitates externas vincit. Saepe civitates victae provinciae deveniunt.
[...]
```

Caesar Augustus imperator Romanus est.



Grammar Creation - Step 3

Step 3: Syntax Trees to Grammars



⇓

```
NP  ::= N PN
NP  ::= A N
VP  ::= VA NP
Cl  ::= NP VP
S   ::= Cl
...
```

First Lesson Grammar

```
abstract PrimaRules = Cat, Conjunction ** {
cat CS ;
fun
  useA : A -> AP ;
  simpleCl : NP -> VP -> Cl ;
  usePN : PN -> NP ;
  usePron : Pron -> NP ;
  useCNdefsg : CN -> NP ;
  useCNindefsg : CN -> NP ;
  useCNindefpl : CN -> NP ;
  complexNP : Det -> CN -> NP ;
  conjNP : NP -> NP -> ListNP ;
  extConjNP : ListNP -> NP -> ListNP ;
  useConjNP : Conj -> ListNP -> NP ;
  useN : N -> CN ;
  attribCN : AP -> CN -> CN ;
  apposCNdefsg : CN -> PN -> NP ;
  useCl : Cl -> S ;
  advS : Adv -> S -> S ;
  intransV : V -> VP ;
  transV : V2 -> NP -> VP ;
  complVA : VA -> AP -> VP ;
  useS : S -> CS ;
}
```

```
abstract PrimaLex = Cat ** {
fun
  copula_VA : VA ;
  copula_V2 : V2 ;
  -- Vocabulary p11
  imperium_N : N ;
  Romanus_A : A ;
  magnus_A : A ;
  imperator_N : N ;
  habere_V2 : V2 ;
  tenere_V2 : V2 ;
  multus_Det : Det ;
  civitas_N : N ;
  externus_A : A ;
  vincere_V2 : V2 ;
  victus_A : A ;
  saepe_Adv : Adv ;
  provincia_N : N ;
  devenire_V2 : V2 ;
  Gallia_PN : PN ;
  Africa_PN : PN ;
  Germanus_N : N ;
  hostis_N : N ;
  dicere_V : V ;
  -- More vocabulary p19
  puella_N : N ;
  laetus_A : A ;
  amicus_N : N ;
  anxius_A : A ;
  vinum_N : N ;
  bonus_A : A ;
  pater_N : N ;
  felix_A : A ;
  coniux_N : N ;
  sapiens_A : A ;
  numen_N : N ;
  ingens_A : A ;
  -- Not in vocabulary list
  Augustus_PN : PN ;
  Caesar_N : N ;
  he_PP : Pron ;
  and_Conj : Conj ;
}
```

CHALMERS | GÖTEBORGS UNIVERSITET

Properties of the Grammars

- limited vocabulary
- small set of syntax rules
- implicitly defined syntactic complexity

Deterministically interpretable (P^4): Fully formalized grammars. Sentence are mapped to finite set of abstract syntax trees

Languages with natural sentence (N^4): Sentences syntactically correct according to the RGL

Languages with short description (S^4): Compact grammars with limited access to external resources like the RGL and additional lexica

No classification (E^-): No formal representation besides the abstract syntax trees (expressivity not relevant for application)

Demo

Conclusion

- Result:
  - General Framework
  - Ready-to-Use System
  - Evaluation: Pilot Study
- Discussion:
  - Grammar Design and Semantics
  - User Interface Improvement
- Future Work:
  - Large-Scale Evaluation
  - Additional Lesson Types
  - and many more

Contact:

Herbert Lange: *herbert.lange@cse.gu.se*
Peter Ljunglöf: *Peter.Ljunglof@cse.gu.se*

Source: `https://github.com/MUSTE-Project/MULLE`

References:

Sara Ehrling. Lingua Latina novo modo – En nybörjarbok i latin för universitetsbruk. University of Gothenburg, 2015.

Peter Ljunglöf. Editing Syntax Trees on the Surface. In Nodalida'11: 18th Nordic Conference of Computational Linguistics, Rīga, Latvia, 2011.

```
incomplete concrete PrimaLexI of PrimaLex = Cat **
  open Structural, Lexicon in {

  lin
  tenere_V2 = Lexicon.hold_V2 ;
  magnus_A = Lexicon.big_A ;
  habere_V2 = Structural.have_V2 ;
  multus_Det = Structural.many_Det ;
  he_PP = Structural.he_Pron ;
  puella_N = Lexicon.girl_N ;
  amicus_N = Lexicon.friend_N ;
  vinum_N = Lexicon.wine_N ;
  bonus_A = Lexicon.good_A ;
  pater_N = Lexicon.father_N2 ;
  and_Conj = Structural.and_Conj ;
}
```

```
--# -path=latin-rgl/api:latin-rgl:.
concrete PrimaLexLat of PrimaLex = CatLat ** PrimaLexI
  with (Cat=CatLat), (Structural=StructuralLat),
  (Lexicon=LexiconLat) ** open ParadigmsLat, (I=IrregLat),
  Prelude, ParamX in {

  lin
    copula_VA = mkVA I.be_V ;
    copula_V2 = mkV2 I.be_V Nom_Prep ;

    imperium_N = mkN "imperium" ;
    Romanus_A = mkA "Romanus" False;
    imperator_N = mkN "imperator" "imperatoris" masculine ;
    civitas_N = mkN "civitas" "civitatis" feminine ;
    externus_A = mkA "externus" ;
    vincere_V2 = Lexicon.win_V2 ;
    victus_A = mkA "victus" ;
    saepe_Adv = mkAdv "saepe" ;
    provincia_N = mkN "provincia" ;
    devenire_V2 = mkV2 (mkV "devenire") Nom_Prep;
[...]
}
```

```
incomplete concrete PrimaRulesI of PrimaRules =
  Cat, Conjunction** open Syntax, Extra in {

lincat
  ListNP = Conjunction.ListNP;

lin
  useA a = lin AP (mkAP (lin A a)) ;
  simpleCl np vp = lin Cl (mkCl (lin NP np) (lin VP vp)) ;
  usePN pn = lin NP (mkNP (lin PN pn)) ;
  usePron pron = lin NP (mkNP (lin Pron pron)) ;
  useCNdefsg cn = lin NP (mkNP theSg_Det (lin CN cn)) ;
  useCNindefsg cn = lin NP (mkNP aSg_Det (lin CN cn)) ;
  useCNindefpl cn = lin NP (mkNP aPl_Det (lin CN cn));
  complexNP det cn = lin NP (mkNP (lin Det det) (lin CN cn)) ;
[...]
}
```

```
--# -path=latin-rgl/api:latin-rgl:.
concrete PrimaRulesLat of PrimaRules = CatLat **
  PrimaRulesI-[useCNdefsg,useCNindefsg,useCNindefpl]
  with (Cat=CatLat), (Syntax=SyntaxLat), (Extra=ExtraLat),
  (Conjunction=ConjunctionLat) ** open ResLat in {

lincat
  CS = Str ;

  lin
    useS s = combineSentence s ! SPreO ! PreV ! SOV ;
}
```