Implementation of a Latin Grammar in Grammatical Framework

Herbert Lange[1]

[1]Computer Science and Engineering
University of Gothenburg and Chalmers
University of Technology
Gothenburg, Sweden
herbert.lange@cse.gu.se

DATeCH 2017, Göttingen, June 1st-2nd

Overview

1. Background
    - Latin
    - Grammatical Framework
    - Resource Grammar Library
2. Grammar
    - Lexicon
    - Morphology
    - Syntax
3. Results
4. Applications

Latin

- Indo-European language
- development almost from 240 b.c. to the beginning of the 20th century (and in some fields still continues)
- Usual focus on classic period (from the first public speeches of M. Tullius Cicero (ca. 80 b.c. to ca. 117 a.d.)
- Strong inflectional and synthetic language
- Rather free word order (but strong tendencies within certain text domains and time periods, also not all combinations seem to be acceptable)

Grammatical Framework

- Modern grammar formalism based on type theory and inspired by functional programming languages (especially Haskell)
- Variant of context-free grammars extended by so called tables and records
- Expressivity equivalent to Parallel Multiple Context-Free Grammars (PMCFG) (mildly context-sensitive)
- Separation in Abstract and Concrete Syntax
- Open Source

CHALMERS | GÖTEBORGS UNIVERSITET

Example

```
param Number = Sg | Pl ;
param Case = Nom | Acc | Dat | Abl | Voc ;
param Gen = Masc | Fem | Neutr ;

lincat N = { s : Number => Case => Str ; g : Gender } ;

lin
  man_N = {
    s = table {
      Sg => table {
        Nom => "vir" ; Acc => "virum " ; Gen => "viri" ;
        Dat => "viro" ; Abl => "viro" ; Voc => "vir" } ;
      Pl => table {
        Nom => "viri" ; Acc => "viros" ; Gen => "virorum" ;
        Dat => "viris" ; Abl => "viris" ; Voc => "viri" ; }
  } ;
  g = Masc
}
```

Resource Grammar Library

- Grammarians equivalent to a Programming Languages standard library
- Common basis for multilingual applications (and machine translation)
- Also Open Source, and this grammar is part of it

General Concept

- Originally: Translate information from a standard (school) grammar book into a computerized form
- Implement the RGL abstract syntax
- Now: Application-specific resources especially for language learning
- Constituent grammar not dependency grammar (not as cool but conversion between GF trees and UD is possible)

Lexicon

- Basic Lexicon with ca 350 entries
- Contains mostly base forms, uses morphological rules to generate the whole paradigms
- Main problem: modern concepts (e.g. refrigerator) and homonyms (e.g. bank)
- Use of web-based and collaborative resources (e.g. Latin Wikipedia, Wiktionary)
- Work in progress: Adopt other lexical resources

Morphology

- Strongly inflectional but quite regular morphology
- Use as little information as possible to generate the whole paradigm (smart paradigms)
- Several inflection classes for different lexical categories
- Use tables for word forms depending on grammatical features

Word class	Inherent	Parametric	No. of Inflection classes
Noun	Gender	Number, Case	5
Adjective		Degree, Gender, Number, Case	3
Verb (active)		Anteriority, Tense, Number, Person	4 regular, 4 deponent
Determiner	Number	Gender, Case	

Feature	Values
Gender	Feminine, Masculine, Neuter
Number	Singular, Plural
Case	Nominative, Genitive, Dative, Accusative, Ablative, Vocative
Degree	Positive, Comparative, Superlative
Anteriority	Anterior, Simultaneous
Tense	Present Indicative, Present Subjunctive, Imperfect Indicative, Imperfect Subjunctive, Future Indicative, Future Subjunctive
Person	1, 2, 3

Syntax

- Syntax rules use basic and smaller parts to assemble larger parts up to the sentence level
- Challenge: Free word order
- Decision what parts can be completely assembled at what point (and what has to be kept apart)
- Use records to keep parts of a phrase separate
- Postpone decision on word order as long as possible

Results

- Comprehensive Latin morphology
- Implemented about 1/3 of the RGL syntactic functions (but some of them are rather obscure)
- Already usable in applications
- Future: Adding rules as they are needed and large-scale evaluation

Applications

- Our main focus: Language learning
  Grammar-based and gamified computer-aided language learning application for beginner's level
- Other possible applications in Digital Humanities
  Giving access to cultural heritage e.g. with a translation app for epigraphs

CHALMERS | GÖTEBORGS UNIVERSITET

Thanks for your attention

Thanks for your attention

Questions?

Source:
`wwww.grammaticalframework.org` and
`https://github.com/daherb/GF-latin`

CHALMERS | GÖTEBORGS UNIVERSITET

```
noun : Str -> Noun = \lexform ->
case lexform of {
- - noun1, noun2us/um/er, noun4 and noun5 are the functions for the different
- - declension classes. The 2nd class is split in three subclasses
_ + "a"  => noun1 lexform ;
_ + "us" => noun2us lexform ;
_ + "um" => noun2um lexform ;
- - "Predef.tk n word" removes a suffix of length n from word
_ + ( "er" | "ir" ) => noun2er lexform ( (Predef.tk 2 lexform) + "ri" ) ;
_ + "u"  => noun4u lexform ;
_ + "es" => noun5 lexform ;
- - Predef.error stops with a given error message
_  => Predef.error ("3rd declinsion cannot be applied " ++
      "to just one noun form " ++ lexform)
} ;
```

```
mkClause : NounPhrase -> VerbPhrase -> Clause = \np,vp -> {
  s = \\tense,anter,pol,vqf,order => case order of {
    SVO => np.s ! Nom ++ negation pol ++ vp.compl ! Ag np.g np.n Nom ++ vp.s ! VAct ( anteriorityT
    VSO => negation pol ++ vp.compl ! Ag np.g np.n Nom ++ vp.s ! VAct ( anteriorityToVAnter anter
    VOS => negation pol ++ vp.compl ! Ag np.g np.n Nom ++ vp.s ! VAct ( anteriorityToVAnter anter
    OSV => vp.obj ++ np.s ! Nom ++ negation pol ++ vp.compl ! Ag np.g np.n Nom ++ vp.s ! VAct ( an
    OVS => vp.obj ++ negation pol ++ vp.compl ! Ag np.g np.n Nom ++ vp.s ! VAct ( anteriorityToVAn
    SOV => np.s ! Nom ++ vp.obj ++ negation pol ++ vp.compl ! Ag np.g np.n Nom ++ vp.s ! VAct ( an
  }
} ;
```