Grammar Engineering Tools

John Camilleri, Ramona Enache, Thomas Hallgren, Aarne Ranta

MOLTO Final Presentation 2013
Grammars as Software

Key to high-quality translation: control over details, debugging

As opposed to: holistic systems, more data, parameter tuning

Similar to: compilers (translators of computer languages)

- expected to translate correctly
- pipeline: parsing + semantic analysis + generation
- semantics encoded in abstract syntax
Compilation via abstract syntax

- Fortran
- C
- Java
- Intel
- ARM
- JVM
Translation via abstract syntax
Translation example

Catalan: Guernica està pintat sobre llenç per Pablo Picasso en 1937.
Dutch: Guernica werd in 1937 door Pablo Picasso op canvas geschilderd.
English: Guernica was painted on canvas by Pablo Picasso in 1937.
Finnish: Guernican maalasi Pablo Picasso kankaalle vuonna 1937.
French: Guernica a été peint sur canvas par Pablo Picasso en 1937.
Multilingual grammar in GF

Declarative program defining the translation relation among any number $n$ of languages

- Abstract: fun Painted : Painting -> Painter -> Fact

- English: lin Painted x y = x ++ "painted" ++ y

- Finnish: lin Painted x y = x ++ "maalasi" ++ y

- French: lin Painted x y = x ++ "a peint" ++ y
But isn’t this too simple-minded?
The complexity of concrete syntax

French: agreement, clitics, ... (il a peint X vs. j’ai peint X vs. il les a peintes ...)

```plaintext
lin
Painted x y = x.s ! Nom ++ case y.isPron of {
  True => y.s ! Acc ++ avoir_V ! x.agr ++ peindre_V ! PastPart y.agr ;
  False => avoir_V ! x.agr ++ peindre_V ! PastPart MascSg ++ y.s ! Acc
}

avoir_V = table ["avoir","ai","as","a","avons",...]
```

Moreover: tenses, negation, question forms, ...
The complexity of multilingual systems

Two dimensions: semantic components X languages. For example:

<table>
<thead>
<tr>
<th>module</th>
<th>Bulgarian</th>
<th>Catalan</th>
<th>Dutch</th>
<th>English</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Answer</td>
<td>AnwerBul</td>
<td>AnswerCat</td>
<td>AnswerDut</td>
<td>AnswerEng</td>
<td>...</td>
</tr>
<tr>
<td>Query</td>
<td>QueryBul</td>
<td>QueryCat</td>
<td>QueryDut</td>
<td>QueryEng</td>
<td>...</td>
</tr>
<tr>
<td>Text</td>
<td>TextBul</td>
<td>TextCat</td>
<td>TextDut</td>
<td>TextEng</td>
<td>...</td>
</tr>
<tr>
<td>Lexicon</td>
<td>LexiconBul</td>
<td>LexiconCat</td>
<td>LexiconDut</td>
<td>LexiconEng</td>
<td>...</td>
</tr>
<tr>
<td>Data</td>
<td>DataBul</td>
<td>DataCat</td>
<td>DataDut</td>
<td>DataEng</td>
<td>...</td>
</tr>
</tbody>
</table>

Museum Library (WP8): \((1 + 15) \times 5 = 80\) modules

Mathematics Library (WP6): \((1 + 15) \times 16 + 27 = 676\) modules
Mastering the complexity

**Programming language**: GF - functions, types, modules

**Compiler**: type checking, optimizations

**Library**: low-level linguistic details

**Development environment**: projects consistency, code navigation

**Documentation**: tutorials, reference manuals, best practices

**Training**: tutorial events, on-line courses

**Community**: helping each other
The GF programming language

First created at Xerox Research in 1998

For **CS people**: a special-purpose functional language for grammars (like YACC, but more powerful)

For **MT people**: a formalism for synchronous grammar (like TAG, but more powerful)

For **language theory people**: a front-end to PMCFG (Parallel Multiple Context-Free Grammars)

**New things during MOLTO:**

- probabilistic GF grammars
The GF compiler

From high-level GF to low-level PGF (Portable Grammar Format)

Separate compilation of modules

Code generation to different formats (e.g. Nuance, XFST/Lexc, Giza)

New things during MOLTO:

- the PGF format
- optimized compilation
- run-time bindings from C, C++, Java, Python
- compilation as cloud service
The GF Resource Grammar Library

Complete morphology engine + comprehensive syntax + lexicon

Afrikaans  Bulgarian  Catalan  Chinese  Danish  Dutch  English
Finnish  French  German  Greek  Hindi  Italian  Japanese
Latvian  Maltese  Nepali  Norwegian  Persian  Polish  Punjabi
Romanian  Russian  Sindhi  Spanish  Swedish  Thai  Urdu

New during MOLTO:

• 13 new languages (built outside MOLTO): 9 Asian, 3 EU
• big lexicon resources (10-100k lemmas) for 11 languages
The library API

<table>
<thead>
<tr>
<th>Function</th>
<th>Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>generic1</td>
<td>VP -&gt; CI</td>
<td>one sleeps</td>
</tr>
<tr>
<td>mk1</td>
<td>NP -&gt; V -&gt; CI</td>
<td>she sleeps</td>
</tr>
<tr>
<td>mk1</td>
<td>NP -&gt; V2 -&gt; NP -&gt; CI</td>
<td>she loves him</td>
</tr>
<tr>
<td>mk1</td>
<td>NP -&gt; V3 -&gt; NP -&gt; NP -&gt; CI</td>
<td>she sends it to him</td>
</tr>
<tr>
<td>mk1</td>
<td>NP -&gt; VV -&gt; VP -&gt; CI</td>
<td>she wants to sleep</td>
</tr>
<tr>
<td>mk1</td>
<td>NP -&gt; VS -&gt; S -&gt; CI</td>
<td>she says</td>
</tr>
<tr>
<td>mk1</td>
<td>NP -&gt; VQ -&gt; QS -&gt; CI</td>
<td>she will</td>
</tr>
<tr>
<td>mk1</td>
<td>NP -&gt; VA -&gt; A -&gt; CI</td>
<td>she became</td>
</tr>
<tr>
<td>mk1</td>
<td>NP -&gt; VA -&gt; AP -&gt; CI</td>
<td>she became</td>
</tr>
<tr>
<td>mk1</td>
<td>NP -&gt; V2A -&gt; NP -&gt; A -&gt; CI</td>
<td>she paid</td>
</tr>
<tr>
<td>mk1</td>
<td>NP -&gt; V2A -&gt; NP -&gt; AP -&gt; CI</td>
<td>she paid</td>
</tr>
<tr>
<td>mk1</td>
<td>NP -&gt; V2S -&gt; NP -&gt; S -&gt; CI</td>
<td>she asks</td>
</tr>
<tr>
<td>mk1</td>
<td>NP -&gt; V2Q -&gt; NP -&gt; QS -&gt; CI</td>
<td>she asks</td>
</tr>
<tr>
<td>mk1</td>
<td>NP -&gt; V2V -&gt; NP -&gt; VP -&gt; CI</td>
<td>she begins</td>
</tr>
<tr>
<td>mk1</td>
<td>NP -&gt; VPSlash -&gt; NP -&gt; CI</td>
<td>she begins</td>
</tr>
<tr>
<td>mk1</td>
<td>NP -&gt; A -&gt; CI</td>
<td>she is</td>
</tr>
<tr>
<td>mk1</td>
<td>NP -&gt; A2 -&gt; NP -&gt; CI</td>
<td>she is</td>
</tr>
<tr>
<td>mk1</td>
<td>NP -&gt; AP -&gt; CI</td>
<td>she is</td>
</tr>
<tr>
<td>mk1</td>
<td>NP -&gt; N -&gt; CI</td>
<td>she is</td>
</tr>
<tr>
<td>mk1</td>
<td>NP -&gt; CN -&gt; CI</td>
<td>she is</td>
</tr>
<tr>
<td>mk1</td>
<td>NP -&gt; Adv -&gt; CI</td>
<td>she is</td>
</tr>
<tr>
<td>mk1</td>
<td>NP -&gt; VP -&gt; CI</td>
<td>she also</td>
</tr>
<tr>
<td>mk1</td>
<td>N -&gt; CI</td>
<td>there is/nose</td>
</tr>
</tbody>
</table>

- API: smart (mk1 she NP want_VV (mk1 sleep_V))
- Afr: sy wil te slaap
- Bul: ma uexa da cru
- Cat: ella vol dormir
- Dan: hun vil sove
- Dut: ze wil slapen
- Eng: she wants to sleep
- Fin: hän tahuoukka
- Fre: elle veut dormir
- Ger: sie will schlafen
- Hin: यह मौन बाहर है
- Ita: lei vuole dormire
- Jpn: 彼女は寝たがっている
- Lav: viiga grib gula
- Nep: जनी सुम्न पाठुपियाँ
- Nor: hun vil sove
- Pers: از می خواهید خوابید
- Pnb: اور سونام جاندی اے
- Pol: ona chce span
- Rom: ea vrea să doarmă
- Rus: ona xovem cnam
- Snd: तुम्हें समय आउँदे
- Spa: ella quiere dormir
- Swe: hon vill sova
- Tha: ผู้ช่วยсонภูษณ์
- Urd: روں سر چاہئے ہی
The painted predicate with RGL

One-liners in every language - grammar writer can ignore details

```haskell
lin Painted x y = mkS pastTense (mkCl x paint_V2 y)
lin Painted x y = mkS pastTense (mkCl x maalata_V2 y)
lin Painted x y = mkS perfectTense (mkCl x peindre_V2 y)
```
GF development environments

GF shell: support for interactive compilation and testing

IDE (Integrated Development Environment) - an Eclipse plug-in

Cloud-based grammar editor: on-line grammar development

New during MOLTO:

- the Eclipse IDE
- the cloud-based grammar editor
GF documentation

http://www.grammaticalframework.org/

100+ articles on GF

New during MOLTO:

- 30+ articles
- Best practices
GRAMMATICAL FRAMEWORK is a programming language designed for writing grammars, which has the capability of addressing several languages in parallel. This thorough introduction demonstrates how to write grammars in Grammatical Framework and use them in applications such as tourist phrasebooks, spoken dialogue systems, and natural language interfaces. The examples and exercises presented here address several languages, and the readers are shown how to look at their own languages from the computational perspective.

Since the book requires no previous knowledge of linguistics, it can be an effective and useful resource for computer scientists and programmers, while introducing linguists to a novel approach to multilingual grammars inspired by the theory of programming languages.

Aarne Ranta is professor of computer science at the University of Gothenburg, Sweden. He is the acting coordinator of the European Union research project MOLTO (Multilingual On-Line Translation), which develops techniques for high-quality translation among fifteen languages.
GF training events

Tutorials in large conferences: LREC-2010, CADE-2011, ICFP-2012

GF Summer Schools: 2009 Gothenburg, **2011 Barcelona**, 2013 Frauenchiemsee (Bavaria)

- 2-week event with 30 participants from 15 countries
GF community

117 members in gf-dev mailing list

Around 50 resource grammar developers

Coverage of world's languages:  http://www.postcrashgames.com/gf_world/

Developers in most of these countries
What is possible

Size of an average application: 15 languages, 200 functions

Size of the biggest application: 5 languages, 56k functions

Effort for building an average grammar: days for the first language, hours for the next ones

Skills required:

• to get a project started: domain expertise, some days of GF training
• to add a language: practical language skills, some hours of GF training
Bootstrapping a grammar

To get started: design abstract syntax to fit an ontology

The first language: concrete syntax using RGL API and parsing examples

Later languages: change the words, and perhaps a few syntax functions

Extend vocabulary: extract words from other sources (wordnet, Wikipedia, Wiktionary)
Example: abstract syntax for CRM ontology

abstract QueryPainting = {
    cat
    Painting ; Query ;
    fun
    QPainter : Painting -> Query ;  -- who painted x
    QYear   : Painting -> Query ;  -- when was x painted
    QMuseum : Painting -> Query ;  -- where is x displayed
    QColour : Painting -> Query ;  -- what colours does x have
    QSize   : Painting -> Query ;  -- what is the size of x
    QMaterial : Painting -> Query ;  -- what material is x painted on
Example: concrete syntax for English

congrete QueryPaintingEng of QueryPainting =
  open LexiconPaintingEng, SyntaxEng, ParadigmsEng in {
  lincat
    Painting = NP ; Query = QS ;
  lin
    QPainter t = mkQS pastTense (mkQCl who_IP paint_V2 t);
    QYear t = mkQS pastTense (mkQCl when_IAdv (mkCl t (passiveVP paint_V2)));
    QMuseum t = mkQS (mkQCl where_IAdv (mkCl t displayed_VP))
    QColour t = mkQS (mkQCl whatPl_IP (mkNP thePl_Det (mkCN (mkN2 colour_N) t))
    QMaterial t = mkQS (mkQCl whatSg_IP (mkNP the_Det (mkCN (mkN2 material_N) t))
    QSize t = mkQS (mkQCl whatSg_IP (mkNP the_Det (mkCN (mkN2 size_N) t)))

Example: concrete syntax for German

concrete QueryPaintingGer of QueryPainting =
  open LexiconPaintingGer, SyntaxGer, ParadigmsGer in {
  lincat
    Painting = NP ; Query = QS ;
  lin
    QPainter t = mkQS pastTense (mkQCl who_IP malen_V2 t) ;
    QYear t = mkQS pastTense (mkQCl when_IAdv (mkCl t (passiveVP malen_V2))) ;
    QMuseum t = mkQS (mkQCl where_IAdv (mkCl t ausgestellt_VP))
    QColour t = mkQS (mkQCl whatPl_IP (mkNP thePl_Det (mkCN (mkN2 farbe_N) t)))
    QMaterial t = mkQS (mkQCl whatSg_IP (mkNP the_Det (mkCN (mkN2 material_N) t)))
    QSize t = mkQS (mkQCl whatSg_IP (mkNP the_Det (mkCN (mkN2 groesse_N) t))) ;
}
The smartest solution: functor

incomplete concrete QueryPaintingI of QueryPainting =
open LexiconPainting, Syntax in {
  lincat
  Painting = NP ; Query = QS ;
  lin
  QPainter t = mkQS pastTense (mkQCl who_IP paint_V2 t) ;
  QYear t = mkQS pastTense (mkQCl when_IAdv (mkCl t (passiveVP paint_V2))) ;
  QMuseum t = mkQS (mkQCl where_IAdv (mkCl t displayed_VP)) ;
  QColour t = mkQS (mkQCl whatPl_IP (mkNP thePl_Det (mkCN (mkN2 colour_N) t))) ;
  QMaterial t = mkQS (mkQCl whatSg_IP (mkNP the_Det (mkCN (mkN2 material_N) t))) ;
  QSize t = mkQS (mkQCl whatSg_IP (mkNP the_Det (mkCN (mkN2 size_N) t))) ;
}

sharing all code but the lexicon (works for 90% of rules)
Example-based grammar writing

Extract translation rule by parsing an example

<table>
<thead>
<tr>
<th>Abstract syntax</th>
<th>Like She He</th>
</tr>
</thead>
<tbody>
<tr>
<td>English example</td>
<td>she likes him</td>
</tr>
<tr>
<td>German translation</td>
<td>er gefällt ihr</td>
</tr>
<tr>
<td>resource tree</td>
<td>mkCl he_Pron gefallen_V2 she_Pron</td>
</tr>
<tr>
<td>concrete syntax rule</td>
<td>Like x y = mkCl y gefallen_V2 x</td>
</tr>
</tbody>
</table>

ORACLE = native speaker or statistical sentence alignment

Methodology with some tool support
The MOLTO heritage

More languages in RGL: reason to build more applications

Applications: reason to support more languages in RGL

Tool of choice for controlled language implementation

Community growth, enterprise awareness

Next step: scaling up to open-domain translation (first experiments in MOLTO)
Demo: `eclipse-film.m4v`

Grammar cloning, library browsing, regression testing
Publications related to MOLTO grammar tools


G. Détrez and A. Ranta. Smart Paradigms and the Predictability and Complexity of Inflectional Morphology. EACL (European Association for Computational Linguistics), Avignon, April 2012.


