Grammar Engineering Tools

John Camilleri, Ramona Enache, Thomas Hallgren, Aarne Ranta

MOLTO Final Presentation 2013

Grammars as Software

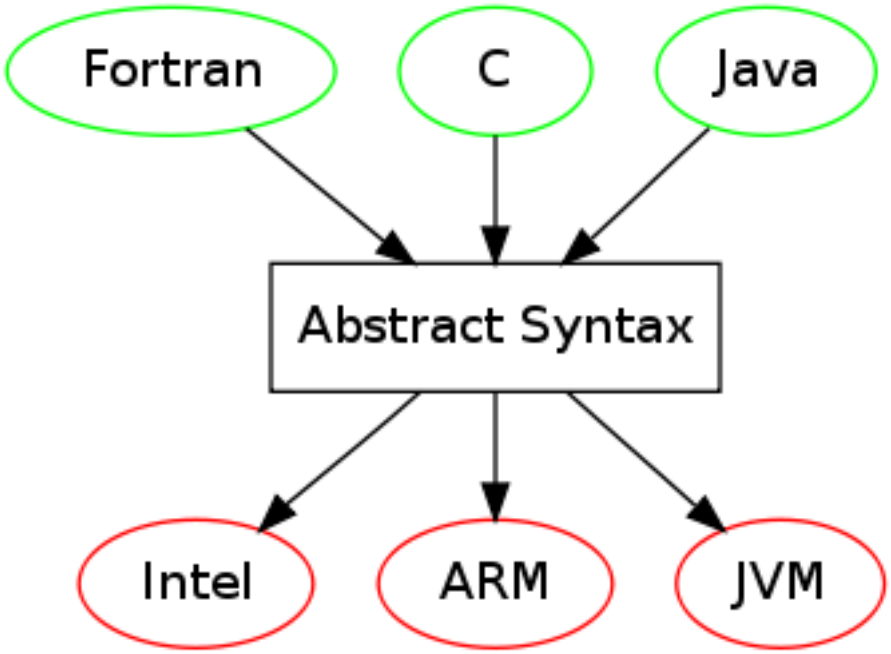
Key to high-quality translation: control over details, debugging

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

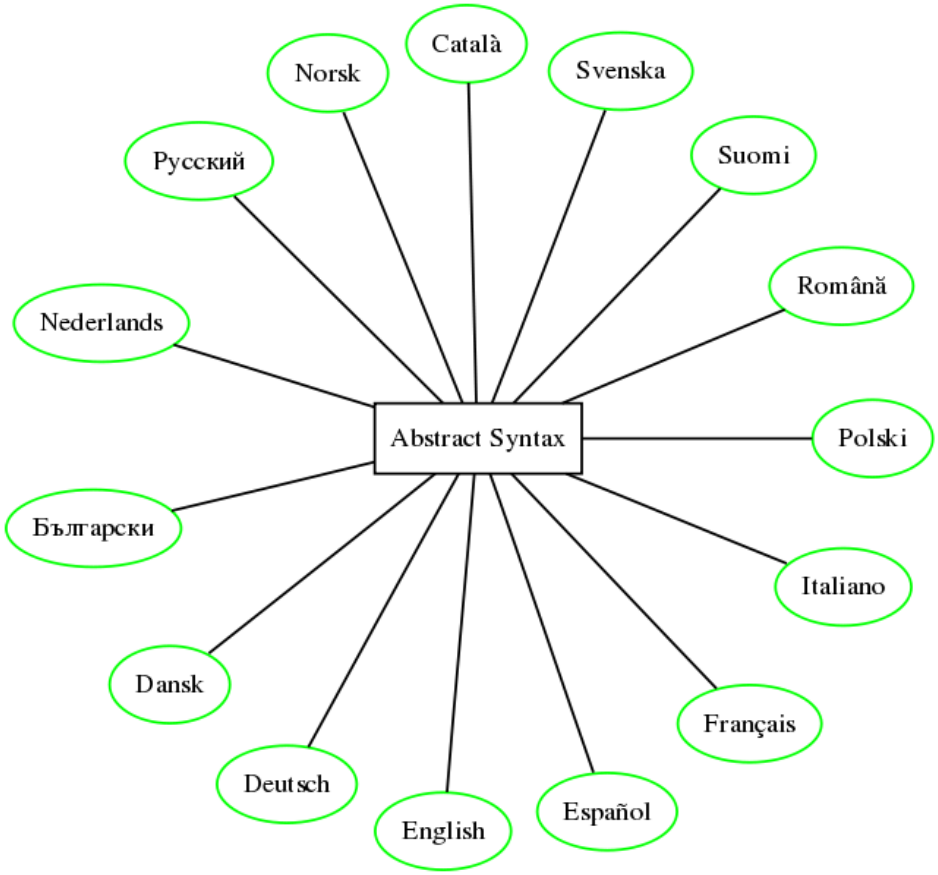
Similar to: compilers (translators of computer languages)

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

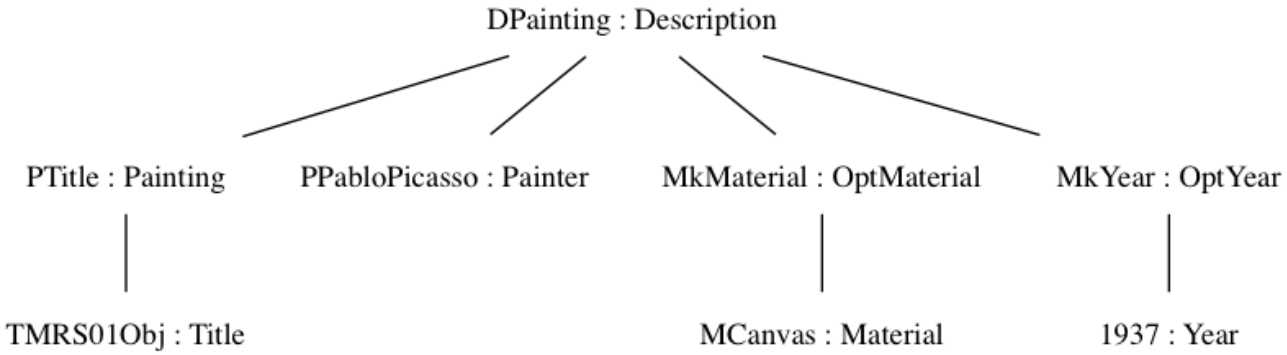
Compilation via abstract syntax



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 two simple-minded?

The complexity of concrete syntax

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

```
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 $\times$ languages

module	**Bulgarian**	**Catalan**	**Dutch**	**English**	**...**
Answer	AnwerBul	AnswerCat	AnswerDut	AnswerEng	...
Query	QueryBul	QueryCat	QueryDut	QueryEng	...
Text	TextBul	TextCat	TextDut	TextEng	...
Lexicon	LexiconBul	LexiconCat	LexiconDut	LexiconEng	...
Data	DataBul	DataCat	DataDut	DataEng	...

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 - abstractions, type system, module system

Compiler: type checking, optimizations

Library: low-lever linguistic details

Development environment: keep projects consistent, help navigate libraries

Documentation: tutorials, reference manuals, best practices

Training: tutorial events, on-line courses

Community: get help from others

The GF programming language

First created at Xerox Research in 1998

For CS people: a special-purpose functional language with modules and dependent types (like YACC, but much more)

For MT people: a formalism for feature-based synchronous grammar

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	**Nepali**	Norwegian	**Persian**	Polish	**Punjabi**	Romanian
Russian	**Sindhi**	Spanish	Swedish	**Thai**	**Urdu**	

New during MOLTO:

- 12 new languages (built outside MOLTO): 9 Asian, 2 EU
- big lexicon resources (10-100k lemmas) for 11 languages

The library API

Cl - declarative clause, with all tenses

Function	Type	Example
`generic Cl`	VP -> Cl	*one sleeps*
`mkCl`	NP -> V -> Cl	*she sleeps*
`mkCl`	NP -> V2 -> NP -> Cl	*she loves him*
`mkCl`	NP -> V3 -> NP -> NP -> Cl	*she sends it to him*
`mkCl`	NP -> VV -> VP -> Cl	*she wants to sleep*
`mkCl`	NP -> VS -> S -> Cl	*she say*
`mkCl`	NP -> VQ -> QS -> Cl	*she wo*
`mkCl`	NP -> VA -> A -> Cl	*she bec*
`mkCl`	NP -> VA -> AP -> Cl	*she bec*
`mkCl`	NP -> V2A -> NP -> A -> Cl	*she pai*
`mkCl`	NP -> V2A -> NP -> AP -> Cl	*she pai*
`mkCl`	NP -> V2S -> NP -> S -> Cl	*she ans*
`mkCl`	NP -> V2Q -> NP -> QS -> Cl	*she ask*
`mkCl`	NP -> V2V -> NP -> VP -> Cl	*she beg*
`mkCl`	NP -> VPSlash -> NP -> Cl	*she beg*
`mkCl`	NP -> A -> Cl	*she is*
`mkCl`	NP -> A -> NP -> Cl	*she is*
`mkCl`	NP -> A2 -> NP -> Cl	*she is*
`mkCl`	NP -> AP -> Cl	*she is*
`mkCl`	NP -> NP -> Cl	*she is*
`mkCl`	NP -> N -> Cl	*she is*
`mkCl`	NP -> CN -> Cl	*she is*
`mkCl`	NP -> Adv -> Cl	*she is*
`mkCl`	NP -> VP -> Cl	*she alw*
`mkCl`	N -> Cl	*there is a house*

- API: `mkUtt (mkCl she_NP want_VV (mkVP sleep_V))`
- Afr: *sy wil te slaap*
- Bul: *тя иска да спи*
- Cat: *ella vol dormir*
- Dan: *hun vil sove*
- Dut: *ze wil slapen*
- Eng: *she wants to sleep*
- Fin: *hän tahtoo nukkua*
- Fre: *elle veut dormir*
- Ger: *sie will schlafen*
- Hin: *वह सोना चाहती है*
- Ita: *lei vuole dormire*
- Jpn: *彼女は寝たがっている*
- Lav: *viņa grib gulēt*
- Nep: *उनी सुत्न चाहन्छिन्*
- Nor: *hun vil sove*
- Pes: *او می خواهد بخوابد*
- Pnb: *او سونا چاندی اے*
- Pol: *ona chce spać*
- Ron: *ea vrea să doarmă*
- Rus: *она хочет спать*
- Snd: *هوءَ سمھ□ چاھی□ی*
- Spa: *ella quiere dormir*
- Swe: *hon vill sova*
- Tha: *หล่อนอยากนอนหลับ*
- Urd: *وہ سونا چاہتی ہے*

The painted predicate with RGL

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

```
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
- The GF book: Aarne Ranta, *Grammatical Framework: Programming with Multilingual Grammars*, CSLI Publications, Stanford, 2011.
- Chinese translation of the book by Yan Tian, Shanghai, 2013.

CSLI Studies in Computational Linguistics

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.

CSLI PUBLICATIONS
Center for the Study of Language and Information
Stanford, California

ISBN-13 978-1-57586-626-0
ISBN-10 1-57586-626-9

Aarne Ranta

Grammatical Framework
Programming with Multilingual Grammars

CSLI

CSLI Studies in Computational Linguistics

Grammatical Framework

Programming with Multilingual Grammars

Aarne Ranta

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

$\sim$ 50 resource grammar developers

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

Developers in most of these countries


Partial coverage
Full coverage

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

```
concrete 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)
    QMaterial t = mkQS (mkQCl whatSg_IP (mkNP the_Det (mkCN (mkN2 material_
    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)
    QMaterial t = mkQS (mkQCl whatSg_IP (mkNP the_Det (mkCN (mkN2 material_
    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)
    QMaterial t = mkQS (mkQCl whatSg_IP (mkNP the_Det (mkCN (mkN2 material_
    QSize t = mkQS (mkQCl whatSg_IP (mkNP the_Det (mkCN (mkN2 size_N) t)))
```

with the language-dependent parameters in

```
interface LexiconPainting = {
  oper paint_V2 : V2 ; displayed_VP : VP ; colour_N, material_N, size_N : N
```

Example-based grammar writing

Extract translation rule by parsing an example

Abstract syntax	`Like She He`	first grammarian
English example	*she likes him*	first grammarian
German translation	*er gefällt ihr*	ORACLE
resource tree	`mkCl he_Pron gefallen_V2 she_Pron`	GF parser
concrete syntax rule	`Like x y = mkCl y gefallen_V2 x`	variables renamed

ORACLE = native speaker or statistical sentence alignment

Methodology with some tool support

The MOLTO heritage

More languages in RGL: reason to build more applications

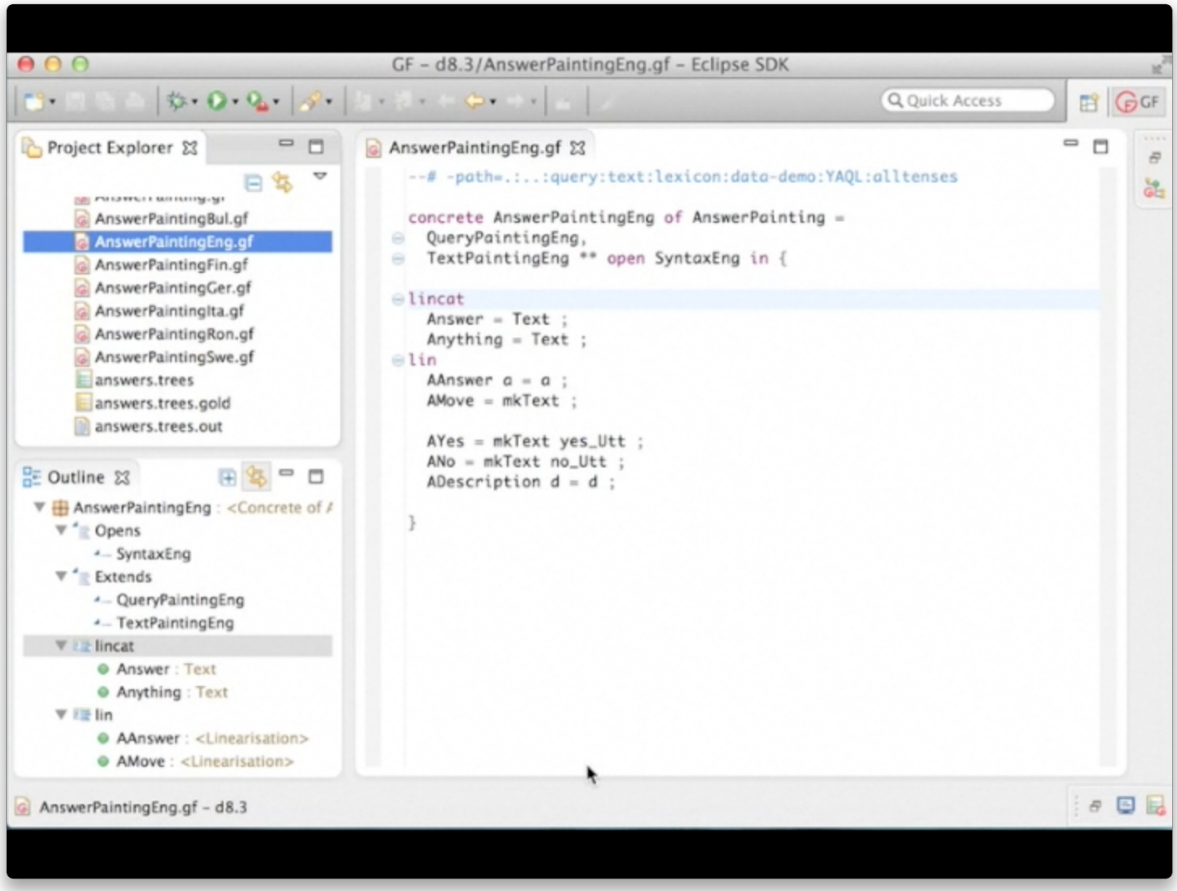
Applications: reason to support more languages in RGL

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