Syntax and Semantics of Translation

Aarne Ranta

WoLLIC 2014, Valparaiso, 1-4 September
Machine Translation: Green, Yellow, and Red

Aarne Ranta

WoLLIC-2014, Valparaiso
4 September 2014
Versions also given at

CLT, Gothenburg, April 2014

NLCS/NLSR, Vienna Summer of Logic, July 2014

CNL, Galway, August 2014
Executive summary

We want to have machine translation that

- delivers **publication quality** in areas where reasonable effort is invested
- degrades gracefully to **browsing quality** in other areas
- shows a clear distinction between these

We do this by using grammars and type-theoretical interlinguas implemented in GF, Grammatical Framework.
Executive summary

We want to have machine translation that

- delivers **publication quality** in areas where reasonable effort is invested
- degrades gracefully to **browsing quality** in other areas
- shows a clear distinction between these

We do this by using **grammars** and **type-theoretical interlinguas** implemented in **GF, Grammatical Framework**
Joint work with

Krasimir Angelov, Björn Bringert, Grégoire Détrez, Ramona Enache, Erik de Graaf, Thomas Hallgren, Qiao Haiyan, Prasanth Kolachina, Inari Listenmaa, Peter Ljunnglöf, K.V.S. Prasad, Scharolta Siencnik, Shafqat Virk

50+ GF Resource Grammar Library contributors
what is your name
comó te llamas

the vice chancellor ordered red wine
el canciller de vicio ordenaba vino rojo

he don't care
él no cuidado
what is your name

como te llamas

the vice chancellor ordered red wine

el canciller de vicio ordenaba vino rojo

he don't care

él no cuidado
what is your name

como te llamas

the vice chancellor ordered red wine

el canciller de vicio ordenaba vino rojo

he don't care

él no cuidado
The Vauquois triangle

**Word to Word**

**Syntactic Transfer**

**Semantic Interlingua**
The Vauquois triangle

semantic interlingua

syntactic transfer

word to word transfer
What is it good for?
publish the content
get the grammar right
get an idea
Who is doing it?
GF in MOLTO

GF the last 15 months

Google, Bing, Apertium
What should we work on?
All!

- semantics for full quality and speed
- syntax for grammaticality
- chunks for robustness and speed
We want a system that
● can reach perfect quality
● has robustness as back-up
● tells the user which is which

We “combine GF, Apertium, and Google”

But we do it all in GF!
The idea is to understand real problems that one would like to solve, and to do it with the standards of the highest quality research. This combines the best features of “applied research” and “basic research.” I’ve always found it productive to look at the details of real problems. Real problems often reveal issues that you wouldn’t think of otherwise.

William A. Woods, ACL Lifetime Achievement Award
The Right Tools: Reflections on Computation and Language
Interlude: SMT
How SMT works

SMT = Statistical Machine Translation

“Lexicon”: word alignments

“Syntax”: n-grams

Word order: distortion model
Word alignments

wine ——— vino 0.7
rojo 0.4
red ——— roja 0.2
rojos 0.2
rojas 0.1
tinto 0.001
black ——— tintos 0.0002
n-grams (n = 2)

libro  rojo  0.01
  roja  0.0001
casa roja  0.01
  rojos  0.00001
vino rojo  0.001
  roja  0.00001
tinto  0.2
Decoding

Selecting the best translation from $f$ to $e$

\[ \hat{e} = \text{argmax}_{e} p(f|e) p(e) \]

Shannon’s noisy channel model (1948)
Decoding in action: word alignments

red       rojo
roja
rojos
rojas
tinto
wine     vino
Decoding in action: distortion

wine  red
red  rojo
roja
rojos
rojas
tinto
Decoding in action: n-grams

wine     vino
red      rojo
         roja
         rojos
         rojas
         tinto
Modern version: phrase alignment

red wine        vino tinto  0.99
              vino rojo  0.01
Problems with SMT

When things are far apart \((n > 3)\)

Sparse data: a language has \(10^6\) “words”

Fundamentally random and uncontrolled

Hard to fix bugs
How to do it in GF?

a brief summary
translator

chunk grammar

resource grammar

application grammar
How much work is needed?
resource grammar

- morphology
- syntax
- generic lexicon

precise linguistic knowledge
manual work can’t be escaped
Words

Suitable word sequences

- Local agreement
- Local reordering

Easily derived from resource grammar
easily varied

Minimize hand-hacking
application grammars

domain semantics, domain idioms
● need domain expertise
use resource grammar as library
● minimize hand-hacking

de the work never ends
● we can only cover some domains
PGF run-time system
- parsing
- linearization
- disambiguation
generic for all grammars
portable to different user interfaces
- web
- mobile
Disambiguation?

**Grammatical**: give priority to green over yellow, yellow over red

**Statistical**: use a distribution model for grammatical constructs (incl. word senses)

**Interactive**: for the last mile in the green zone
Advantages of GF

Expressivity: easy to express complex rules
- agreement
- word order
- discontinuity

Abstractions: easy to manage complex code

Interlinguuality: easy to add new languages
Resources: basic and bigger

Norwegian    Danish    Afrikaans

English    Swedish    German    Dutch
French    Italian    Spanish
Bulgarian    Chinese    Finnish

Catalan    Estonian

Greek    Thai    Japanese    Urdu    Punjabi    Sindhi

Nepali    Persian
my new house is very big

मेरा अजनबी शाला बहुत महत्वपूर्ण है

你爱我吗

est-ce que tu m'aimes

ich wohne in einem gelben Haus

io risiedo in una casa gialla

jag är inte en älg

minä en ole hirvi
How to do it?

some more details
Translation model: multi-source multi-target compiler

Diagram:
- Fortran
- C
- Java
- Abstract Syntax
- Intel
- ARM
- JVM
Translation model: multi-source multi-target compiler-decompiler

Abstract Syntax

- Hindi
- Chinese
- Finnish
- Bulgarian
- Spanish
- Italian
- German
- French
- Swedish
- English
Word alignment: compiler

1 + 2 * 3

00000011 00000100 00000101 01101000 01100000
Abstract syntax

Add : Exp -> Exp -> Exp
Mul : Exp -> Exp -> Exp
E1, E2, E3 : Exp

Add E1 (Mul E2 E3)
## Concrete syntax

<table>
<thead>
<tr>
<th>Abstract</th>
<th>Java</th>
<th>JVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add $x , y$</td>
<td>$x , \text{“}+\text{“} , y$</td>
<td>$x , y , \text{“}01100000\text{“}$</td>
</tr>
<tr>
<td>Mul $x , y$</td>
<td>$x , \text{“}\times\text{“} , y$</td>
<td>$x , y , \text{“}01101000\text{“}$</td>
</tr>
<tr>
<td>$E_1$</td>
<td>“1”</td>
<td>“00000011”</td>
</tr>
<tr>
<td>$E_2$</td>
<td>“2”</td>
<td>“00000100”</td>
</tr>
<tr>
<td>$E_3$</td>
<td>“3”</td>
<td>“00000101”</td>
</tr>
</tbody>
</table>
Compiling natural language

Abstract syntax

\[ \text{Pred} : \text{NP} \rightarrow \text{V2} \rightarrow \text{NP} \rightarrow \text{S} \]
\[ \text{Mod} : \text{AP} \rightarrow \text{CN} \rightarrow \text{CN} \]
\[ \text{Love} : \text{V2} \]

Concrete syntax:

<table>
<thead>
<tr>
<th></th>
<th>English</th>
<th>Latin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pred s v o</td>
<td>s v o</td>
<td>s o v</td>
</tr>
<tr>
<td>Mod a n</td>
<td>a n</td>
<td>n a</td>
</tr>
<tr>
<td>Love</td>
<td>“love”</td>
<td>“amare”</td>
</tr>
</tbody>
</table>
the clever woman loves the handsome man

femina sapiens virum formosum amat

Pred (Def (Mod Clever Woman)) Love
(Def (Mod Handsome Man))
Linearization types

<table>
<thead>
<tr>
<th>English</th>
<th>Latin</th>
</tr>
</thead>
<tbody>
<tr>
<td>CN</td>
<td>{s : Number =&gt; Str} {s : Number =&gt; Case =&gt; Str ; g : Gender}</td>
</tr>
<tr>
<td>AP</td>
<td>{s : Str} {s : Gender =&gt; Number =&gt; Case =&gt; Str}</td>
</tr>
</tbody>
</table>

Mod ap cn

\{s = \n => ap.s ++ cn.s ! n\} \{s = \n,c => cn.s ! n ! c ++ ap.s ! cn.g ! n ! c ; g = cn.g \}
Abstract syntax trees

my name is John

HasName I (Name “John”)
Abstract syntax trees

my name is John

HasName I (Name “John”)

Pred (Det (Poss i_NP) name_N)) (NameNP “John”)
Abstract syntax trees

my name is John

HasName I (Name “John”)

Pred (Det (Poss i_NP) name_N)) (NameNP “John”)

[DetChunk (Poss i_NP), NChunk name_N, copulaChunk, NPChunk (NameNP “John”)]
Building the yellow part
Building a basic resource grammar

Programming skills
Theoretical knowledge of language
3-6 months work
3000-5000 lines of GF code
- not easy to automate
+ only done once per language
Building a large lexicon

Monolingual (morphology + valencies)
- extraction from open sources (SALDO etc)
- extraction from text (extract)
- smart paradigms

Multilingual (mapping from abstract syntax)
- extraction from open sources (Wordnet, Wiktionary)
- extraction from parallel corpora (Giza++)

Manual quality control at some point needed
Improving the resources

**Multiwords**: non-compositional translation
- *red wine* - *vino tinto*

**Constructions**: multiwords with arguments
- *x’s name is y* - *x se llama y*

Extraction from free resources (Konstruktkikon)
Extraction from SMT phrase tables
- example-based grammar writing
It’s important to look at the details. Try to understand what would be necessary to solve the whole problem. At this point, don’t settle for approximations.

Woods, *ibid.*
Building the red part
1. Write a grammar that builds sentences from sequences of chunks

   cat Chunk
   fun SChunks : [Chunk] -> S

2. Introduce chunks to cover phrases

   fun NP_nom_Chunk : NP -> Chunk
   fun NP_acc_Chunk : NP -> Chunk
   fun AP_sg_masc_Chunk : AP -> Chunk
   fun AP_pl_fem_Chunk : AP -> Chunk
Do this for all categories and feature combinations you want to cover.

Include both long and short phrases
- long phrases have better quality
- short phrases add to robustness

Give long phrases priority by probability settings.
Long chunks are better:

[this yellow house] - [det här gula huset]

[this] [yellow house] - [den här] [gult hus]

[this] [yellow] [house] - [den här] [gul] [hus]

Limiting case: whole sentences as chunks.
Accurate feature distinctions are good, especially between closely related language pairs.

Apertium does this for every language pair.
Resource grammar chunks of course come with reordering and internal agreement.

\[
\begin{align*}
\text{Prep} & \quad \text{Det+Fem+Sg} & \quad \text{N+Fem+Sg} & \quad \text{A+Fem+Sg} \\
dans & \quad la & \quad maison & \quad bleue \\
im & \quad & \quad & \\
\text{blauen} & \quad & \quad Haus & \\
\text{Prep-Det+Neutr+Sg+Dat} & \quad & \quad A+Weak+Dat & \quad N+Neutr+Sg
\end{align*}
\]
Recall: chunks are just a by-product of the real grammar.

Their size span is

single words  <---->  entire sentences

A wide-coverage chunking grammar can be built in a couple of hours by using the RGL.
If you have a practical job to do, and it’s important to get it done quickly as well as possible, and you can only do that by partially solving the problem, then by all means do that. That’s practical engineering, and I do that with my Engineer’s hat on. But that’s not going to advance the science.

Woods, ibid.
Building the green part
Define **semantically based abstract syntax**

```latex
fun HasName : Person -> Name -> Fact
```

Define **concrete syntax by mapping to resource grammar structures**

```latex
lin HasName p n = mkCl (possNP p name_N) y

my name is John

lin HasName p n = mkCl p heta_V2 y

jag heter John

lin HasName p n = mkCl p (reflV chiamare_V) y

(io) mi chiamo John
```
Resource grammars give crucial help

- application grammarians need not know linguistics
- a substantial grammar can be built in a few days
- adding new languages is a matter of a few hours

MOLTO’s goal was to make this possible.
Automatic extraction of application grammars?

- abstract syntax from ontologies
- concrete syntax from examples
  - including phrase tables

As always, full green quality needs expert verification

- formal methods help (REMU project)
These grammars are a source of
● “non-compositional” translations
● compile-time transfer
● idiomatic language
● translating meaning, not syntax

Constructions are the generalized form of this idea, originally domain-specific.
Building the translation system
GF source
GF source

probability model
GF source → GF compiler → PGF binary
user interface

PGF runtime system

PGF binary
user interface

PGF runtime system

another PGF binary

another app
White: free, open-source. Green: a business idea

generic user interface

PGF runtime system

generic grammar

app

custom user interface
User interfaces

command-line
shell
web server
web applications
mobile applications
Demos
To test it yourself

Android app

http://www.grammaticalframework.org/demos/app.html

Web app

http://www.grammaticalframework.org/demos/translation.html
Agenda for future work
Improve the lexicon

Split senses

Improve disambiguation

Introduce constructions

Design and perform evaluation
# Current dictionary coverage

<table>
<thead>
<tr>
<th>Language</th>
<th>Total words</th>
<th>Checked words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulgarian</td>
<td>36666</td>
<td>21372</td>
</tr>
<tr>
<td>Chinese</td>
<td>17000</td>
<td>16475</td>
</tr>
<tr>
<td>Dutch</td>
<td>17000</td>
<td>2154</td>
</tr>
<tr>
<td>English</td>
<td>66000</td>
<td>66000</td>
</tr>
<tr>
<td>Finnish</td>
<td>57000</td>
<td>4700</td>
</tr>
<tr>
<td>French</td>
<td>20000</td>
<td>1155</td>
</tr>
<tr>
<td>German</td>
<td>22000</td>
<td>1693</td>
</tr>
<tr>
<td>Hindi</td>
<td>34000</td>
<td>175</td>
</tr>
<tr>
<td>Italian</td>
<td>16000</td>
<td>641</td>
</tr>
<tr>
<td>Spanish</td>
<td>21000</td>
<td>2285</td>
</tr>
<tr>
<td>Swedish</td>
<td>25000</td>
<td>2259</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>total words</th>
<th>checked words</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>


Splitting senses

time
Splitting senses

time_N

time_V
Splitting senses

tiempo

time_N

vez
Splitting senses

time_1_N  tiempo

time_2_N  vez
Splitting senses

time_1_N   tiempo   Zeit

time_2_N   vez    Mal
Splitting senses

weather_N

Wetter

time_1_N
tiempo

Zeit

time_2_N
vez

Mal
See also: 4th GF Summer School

19-31 July 2015 in Marsalforn, Malta