Machine Translation:

Green, Yellow, and Red

Aarne Ranta

NLCS/NLSR, Vienna Summer of Logic
18 July 2014

CLT REMU digital Grammars
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, Prasanth Kolachina, Inari Listenmaa, Peter Ljunnglöf, K.V.S. Prasad, Scharolta Siencnik, Shafqat Virk

50+ GF Resource Grammar Library contributors
what is your wife's name
vad heter din fru
the vice president kicked the bucket
skruvstådspräsidendenten
sparkade hinken
long time no see
lång tid nej ser
what is your wife's name

vad heter din fru

the vice president kicked the bucket

skruvstådspresidenten
sparkade hinken

long time no see

lång tid nej ser
what is your wife's name
- translation by meaning
  - correct
  - idiomatic

the vice president kicked the bucket
- translation by syntax
  - grammatical
  - often strange
  - often wrong

long time no see
- translation by chunks
  - probably ungrammatical
  - probably wrong
The Vauquois triangle

- Word to word transfer
- Syntactic transfer
- Semantic interlingua
The Vauquois triangle

- **semantic interlingua**
- **syntactic transfer**
- **word to word transfer**
What is it good for?
get an idea

got the grammar right

publish the content
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
How to do it?

a brief summary
How much work is needed?
translator

chunk grammar

application grammars

resource grammar
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

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

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

Norwegian    Danish    Afrikaans

English    Swedish    German    Dutch
French    Italian    Spanish
Bulgarian    Chinese    Finnish

Hindi

English

Swedish

German

Dutch

French

Italian

Spanish

Bulgarian

Chinese

Finnish

Hindi

Nepali

Persian

Thai

Japanese

Urdu

Punjabi

Sindhi

Greek
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
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

\[ \text{Add} : \text{Exp} \to \text{Exp} \to \text{Exp} \]

\[ \text{Mul} : \text{Exp} \to \text{Exp} \to \text{Exp} \]

\[ \text{E1, E2, E3} : \text{Exp} \]

\[ \text{Add E1 (Mul E2 E3)} \]
## Concrete syntax

<table>
<thead>
<tr>
<th>abstrakt</th>
<th>Java</th>
<th>JVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add $x \ y$</td>
<td>$x \ &quot;+&quot; \ y$</td>
<td>$x \ y \ &quot;01100000&quot;$</td>
</tr>
<tr>
<td>Mul $x \ y$</td>
<td>$x \ &quot;\ast&quot; \ y$</td>
<td>$x \ y \ &quot;01101000&quot;$</td>
</tr>
<tr>
<td>$E1$</td>
<td>&quot;1&quot;</td>
<td>&quot;00000011&quot;</td>
</tr>
<tr>
<td>$E2$</td>
<td>&quot;2&quot;</td>
<td>&quot;00000100&quot;</td>
</tr>
<tr>
<td>$E3$</td>
<td>&quot;3&quot;</td>
<td>&quot;00000101&quot;</td>
</tr>
</tbody>
</table>
Compiling natural language

Abstract syntax

Pred : NP -> V2 -> NP -> S
Mod : AP -> CN -> CN
Love : 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>
Word alignment

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></th>
<th>English</th>
<th>Latin</th>
</tr>
</thead>
<tbody>
<tr>
<td>CN</td>
<td>{s : Number =&gt; Str}</td>
<td>{s : Number =&gt; Case =&gt; Str ; g : Gender}</td>
</tr>
<tr>
<td>AP</td>
<td>{s : Str}</td>
<td>{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
- *kick the bucket* - *ta ner skylten*

Constructions: multiwords with arguments
- *i sötaste laget* - *excessively sweet*

Extraction from free resources (Konstruktikon)
Extraction from 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
   \[ \text{cat Chunk} \]
   \[ \text{fun SChunks : } [\text{Chunk}] \rightarrow S \]

2. Introduce chunks to cover phrases
   \[ \text{fun NP\_nom\_Chunk : NP } \rightarrow \text{ Chunk} \]
   \[ \text{fun NP\_acc\_Chunk : NP } \rightarrow \text{ Chunk} \]
   \[ \text{fun AP\_sg\_masc\_Chunk : AP } \rightarrow \text{ Chunk} \]
   \[ \text{fun AP\_pl\_fem\_Chunk : AP } \rightarrow \text{ Chunk} \]
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.

Prep         Det+Fem+Sg   N+Fem+Sg          A+Fem+Sg

*dans*     *la*         *maison*       *bleue*

Prep-Det+Neutr+Sg+Dat   A+Weak+Dat   N+Neutr+Sg

*im*   *blauen*   *Haus*
Recall: chunks are just a by-product of the real grammar.

Their size span is

single words \( \longleftrightarrow \) 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**

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

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

```
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

GF compiler

PGF binary

probability model
user interface

PGF runtime system

PGF binary
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 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>