Machine Translation: Green , Yellow , and Red

Aarne Ranta

University of Gothenburg and Digital Grammars AB

Talk given at Hong Kong Polytechnic University

10 November 2014

CLT REMU digital Grammars

Language technology to rely on.

Versions also given at

CLT, U Gothenburg, April 2014

NLCS/NLSR, Vienna Summer of Logic, July 2014

CNL, Galway, August 2014

WoLLIC, Valparaiso, September 2014

Dept of Mathematics, U Stockholm, September 2014

Shanghai University of Finance and Economics, Nov 2014

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

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**

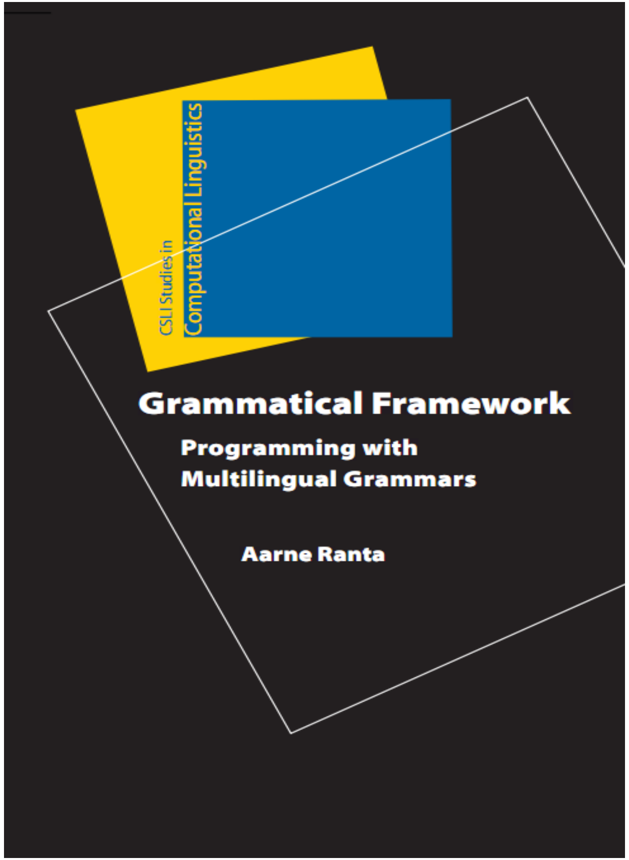
GF = Grammatical Framework

Grammar formalism based on **type theory** and **functional programming.**

Started at Xerox Research in 1998, as a tool for **highly multilingual, controlled language** translation.

Closest prior work: Montague grammar, Rosetta (Philips).

Latest developments have scaled it up in **productivity** and also **coverage**.



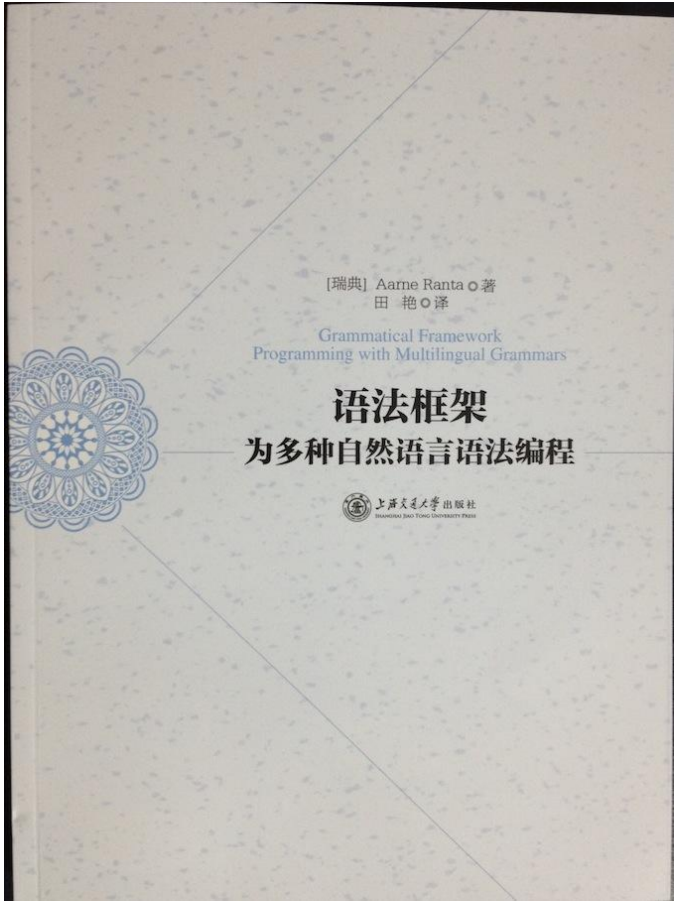
CSLI Studies in
Computational Linguistics

Grammatical Framework

Programming with
Multilingual Grammars

Aarne Ranta

CSLI, Stanford, 2011



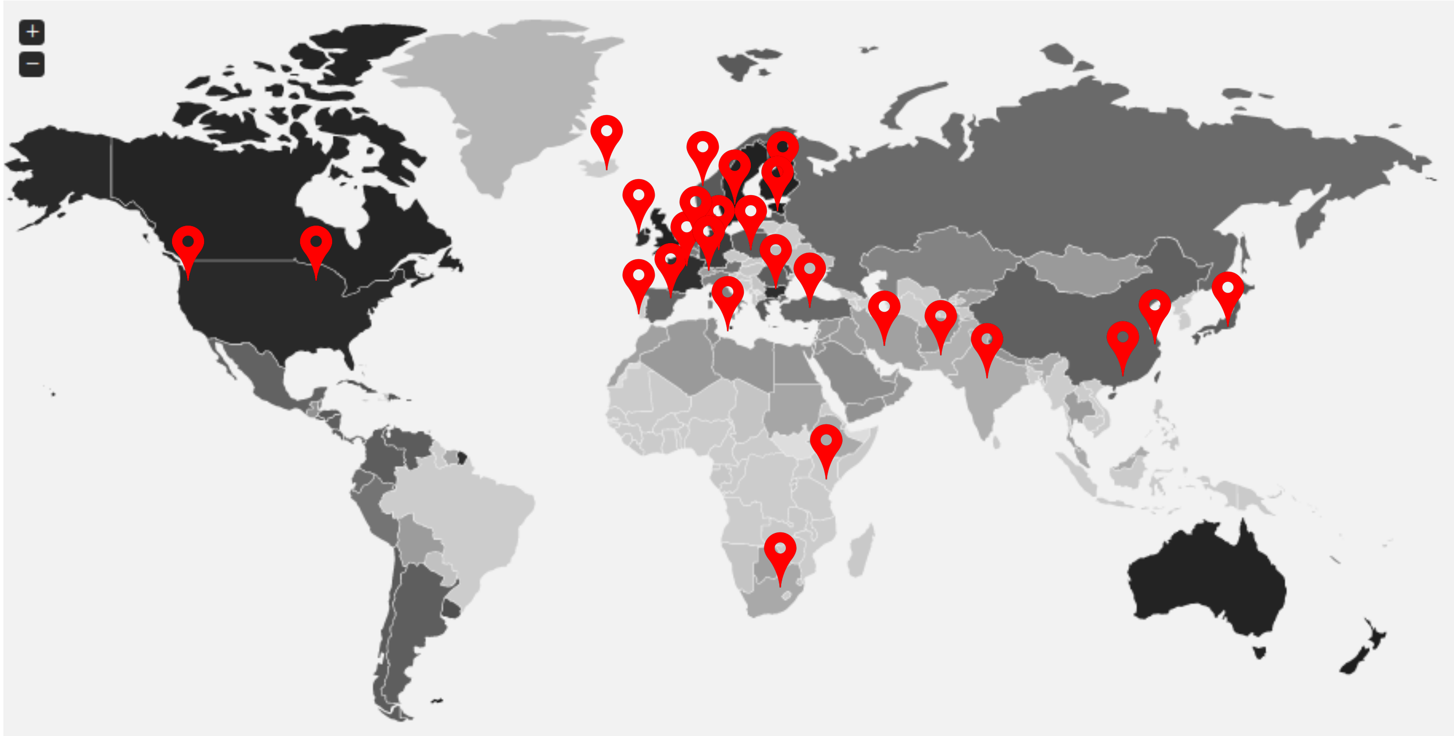
[瑞典] Aarne Ranta○著
田 艳○译

Grammatical Framework
Programming with Multilingual Grammars

语法框架
为多种自然语言语法编程

上海交通大学出版社
SHANGHAI JIAO TONG UNIVERSITY PRESS

Shanghai Jiao Tong University press, 2014



digitalGrammars

Language technology to rely on.

5 March 2014 -

REMU

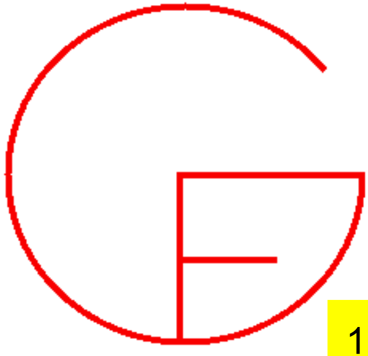
VR 2013 - 2017

MOLTO

EU 2010 - 2013

CLT

2009 -



1998 -

Translation: producer vs. consumer

Consumer:

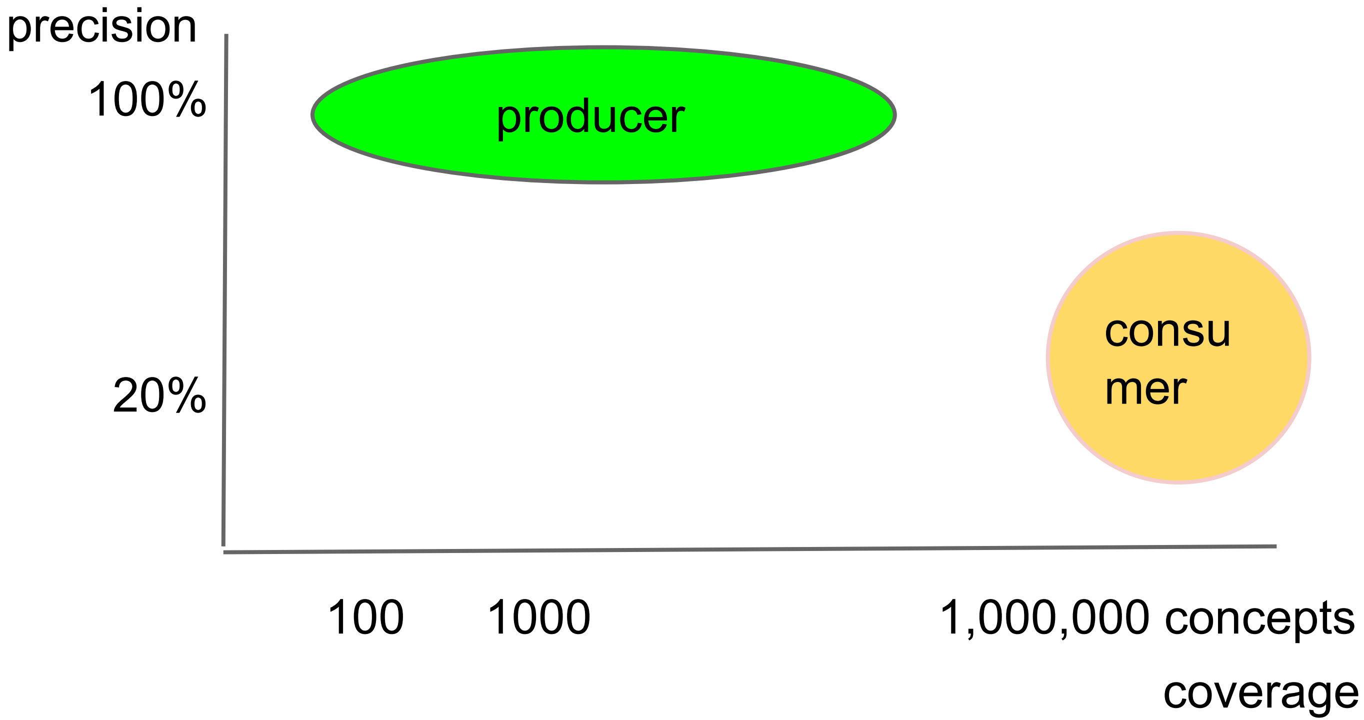
- must translate anything
- browsing quality enough

Producer:

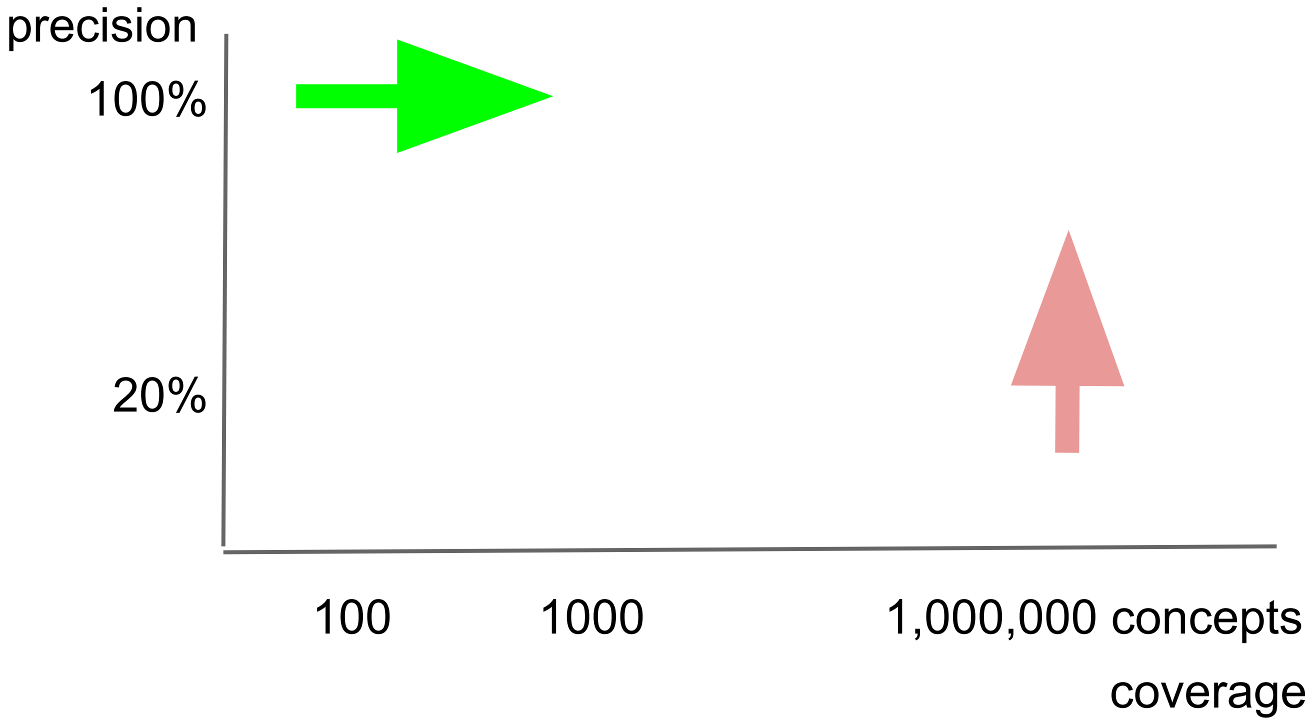
- must translate my content
- publication quality required

MT mainstream is consumer tools

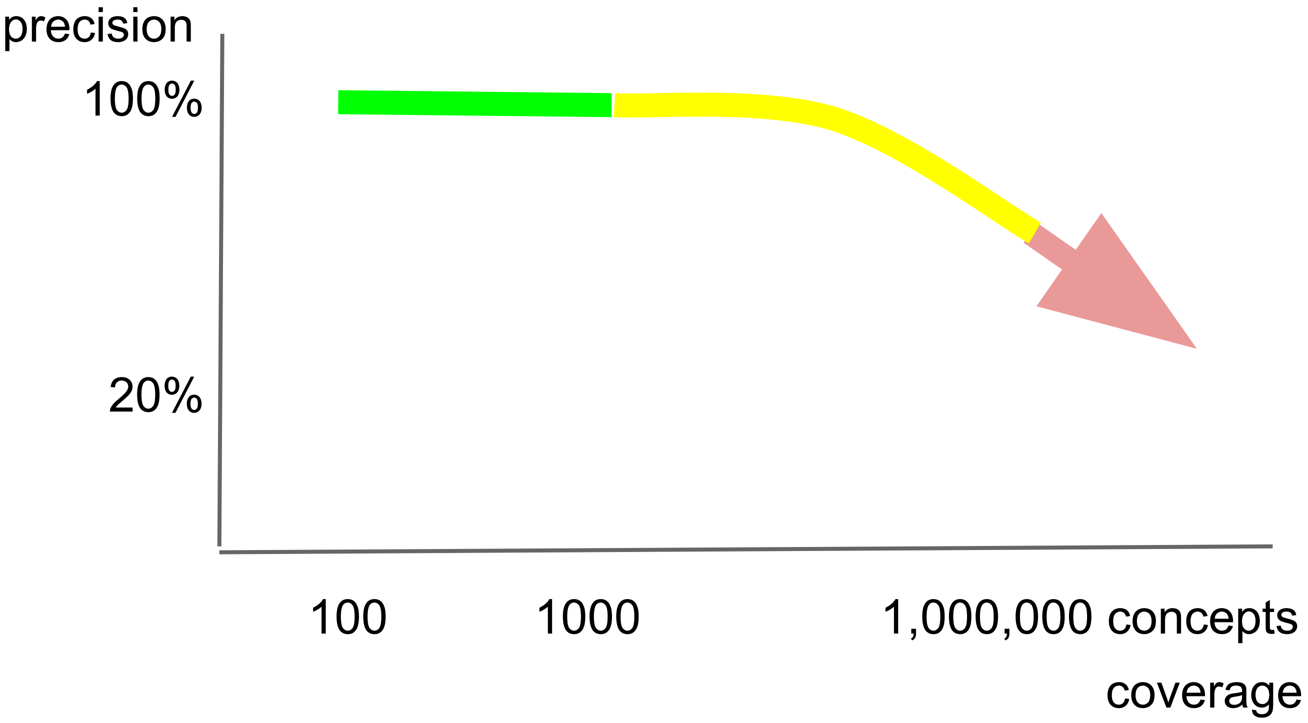
Orthogonal concepts



Two ways of developing a system



The best scenario?



An example

How far is the airport from the hotel?

从 旅 馆 到 机 场 有 多 远?

The vice dean kicked the bucket.

副 院 长 踢 了 桶.

Little boy eat big snake.

小 男 孩 吃 大 蛇.

An example

How far is the airport from the hotel?

从 旅 馆 到 机 场 有 多 远?

The vice dean kicked the bucket.

副 院 长 踢 了 桶.

Little boy eat big snake.

小 男 孩 吃 大 蛇.

An example

How far is the airport from the hotel?

从 旅 馆 到 机 场 有 多 远?

meaning

The vice dean kicked the bucket.

副 院 长 踢 了 桶.

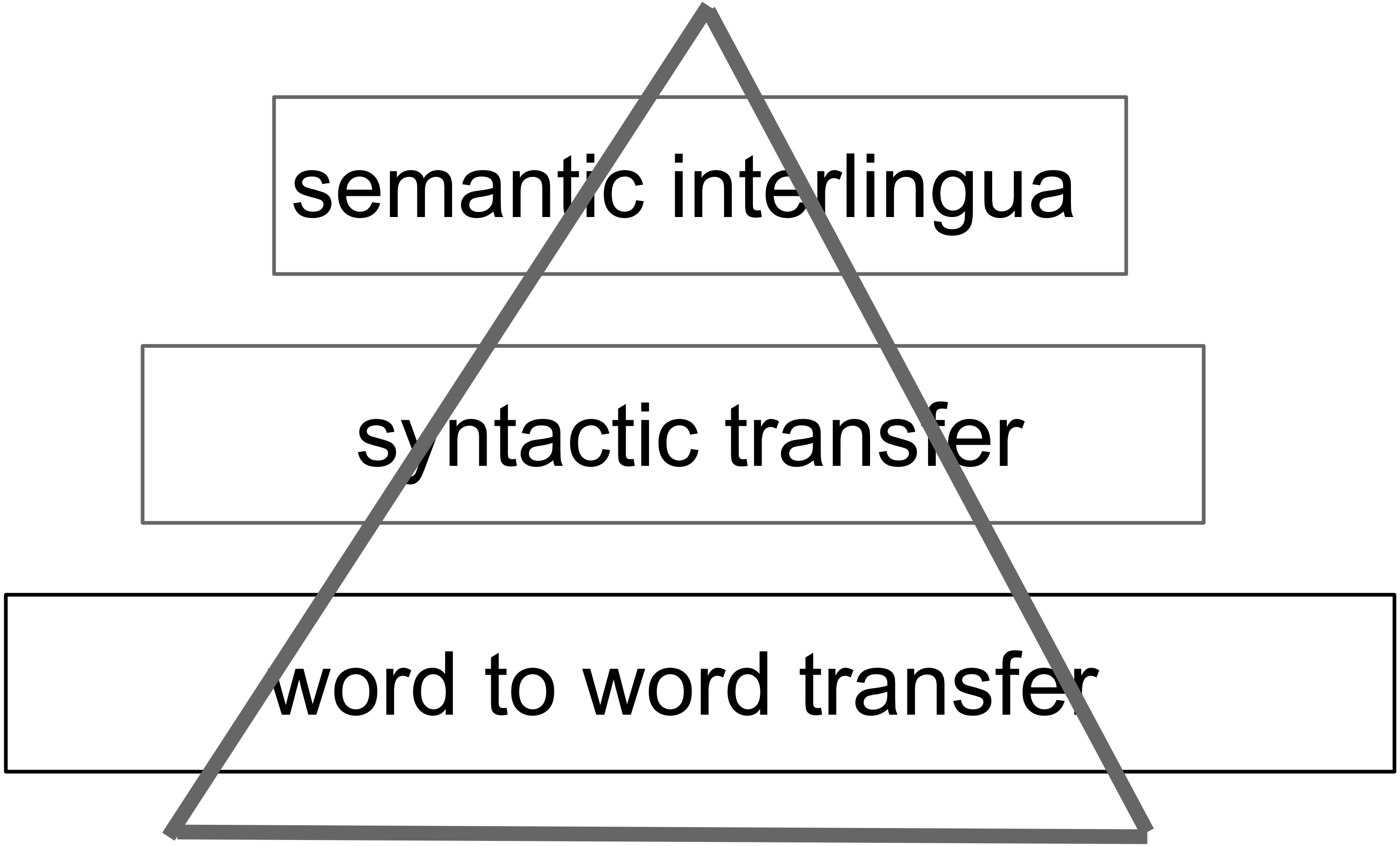
syntax

Little boy eat big snake.

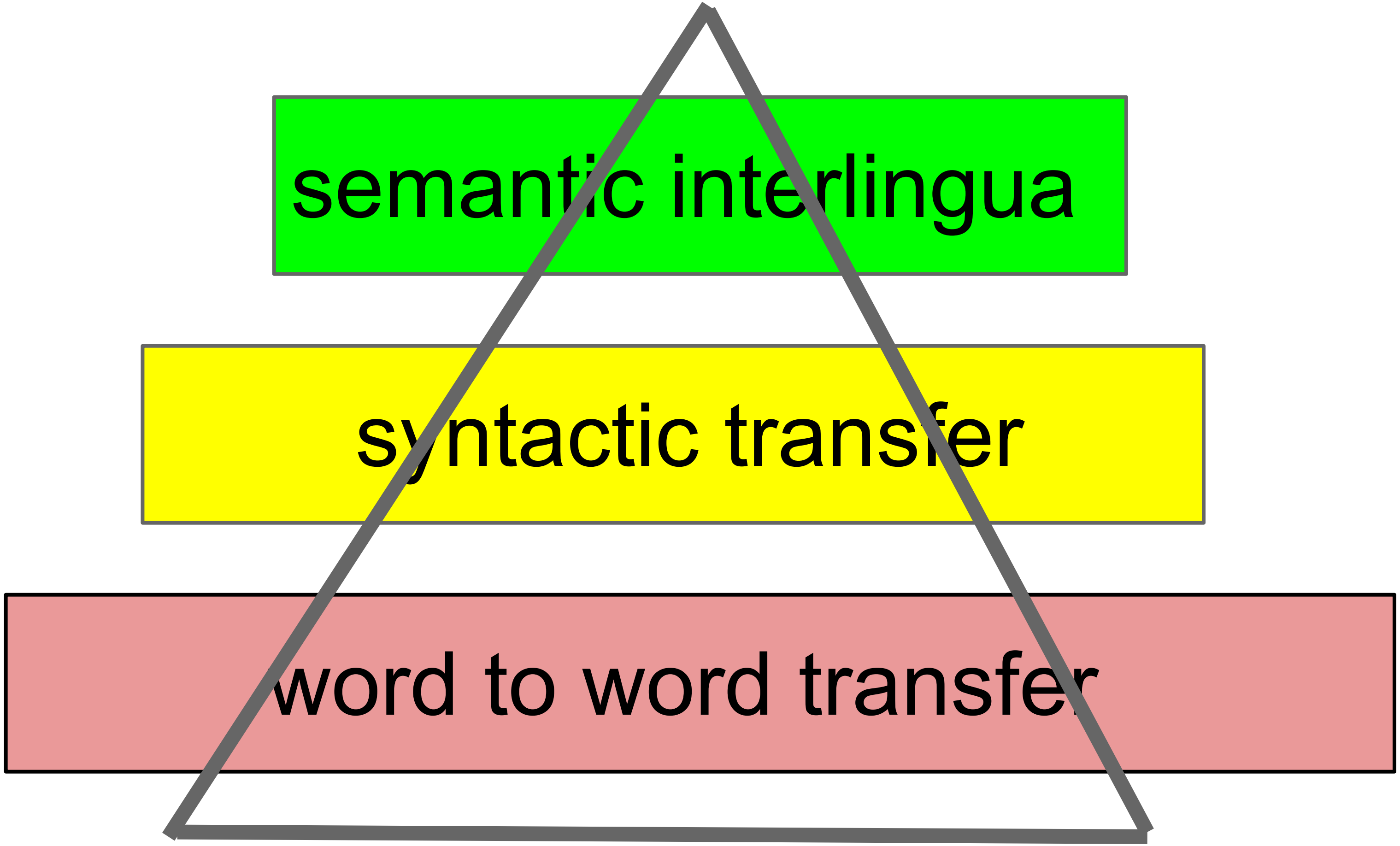
小 男 孩 吃 大 蛇.

chunks

The Vauquois triangle



The Vauquois triangle

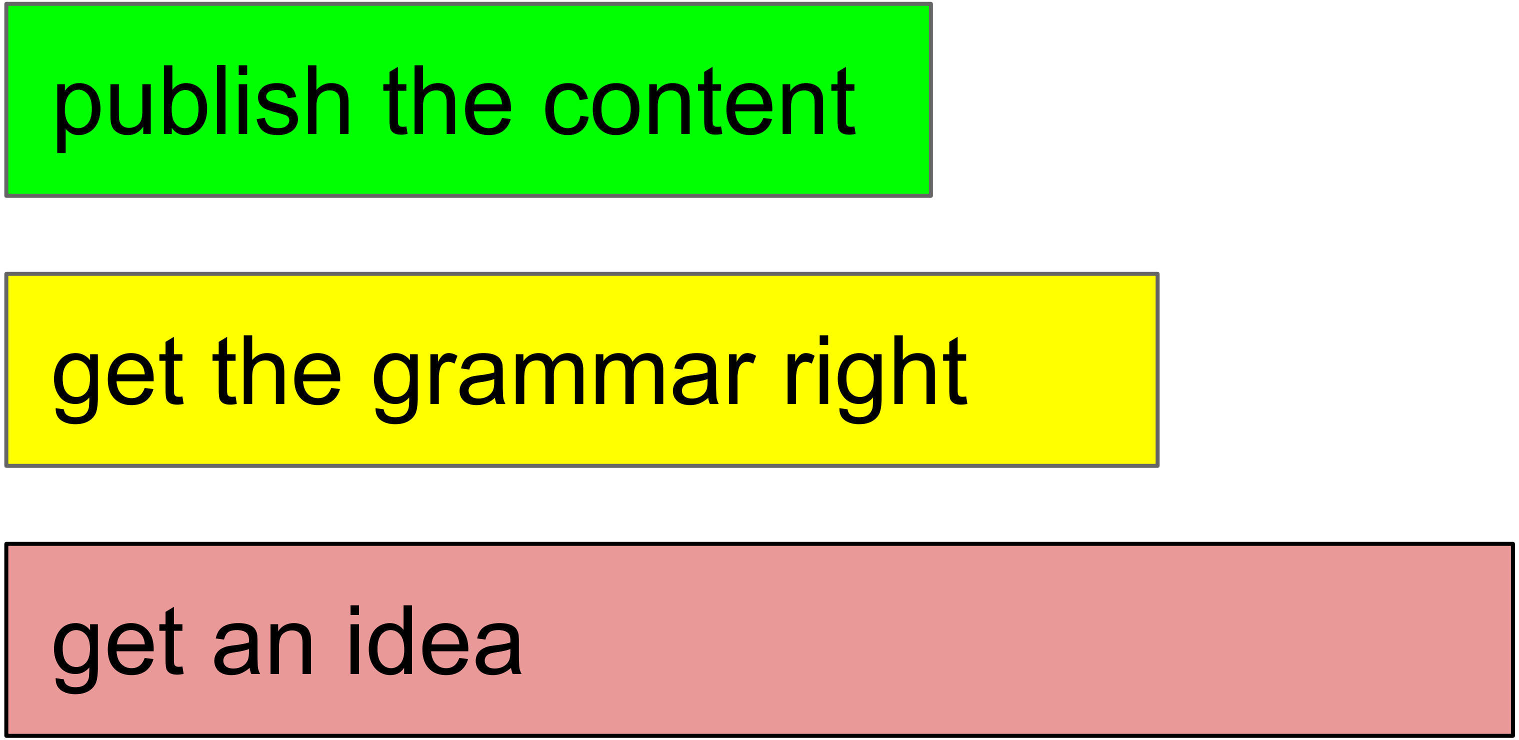


semantic interlingua

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 18 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!

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

Long-distance dependencies

She is happy.	*Elle est heureuse.*
She is usually very happy.	*Elle est généralement très heureux.*

(Google translate 9 November 2014)

Long-distance dependencies

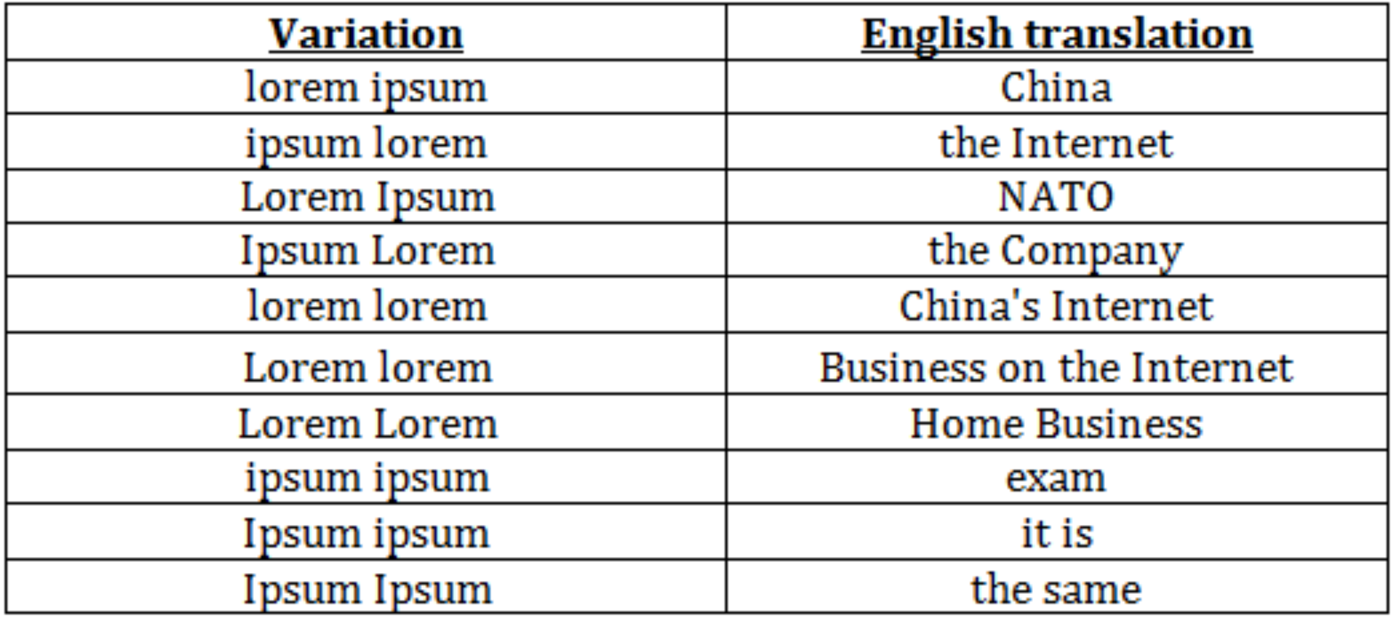
I have five cats	我有五**只猫**
I have five very big cats	我有五**个**非常大的**猫**

Er bringt dich um.	*He is killing you.*
Er ***bringt*** *deinen besten Freund* ***um****.*	*He brings to your best friend.*

A missing word doesn’t cost much

Min far är svensk.	我的父亲是瑞典。
Min far är inte svensk.	我的父亲是瑞典。

Predictability and controllability



Variation	**English translation**
lorem ipsum	China
ipsum lorem	the Internet
Lorem Ipsum	NATO
Ipsum Lorem	the Company
lorem lorem	China's Internet
Lorem lorem	Business on the Internet
Lorem Lorem	Home Business
ipsum ipsum	exam
Ipsum ipsum	it is
Ipsum Ipsum	the same

Google translate mid-2014, reported in
http://krebsonsecurity.com/2014/08/lorem-ipsum-of-good-evil-google-china/

What SMT is good for

Short, common expressions

- idiomacy
- local disambiguation

Acquiring data

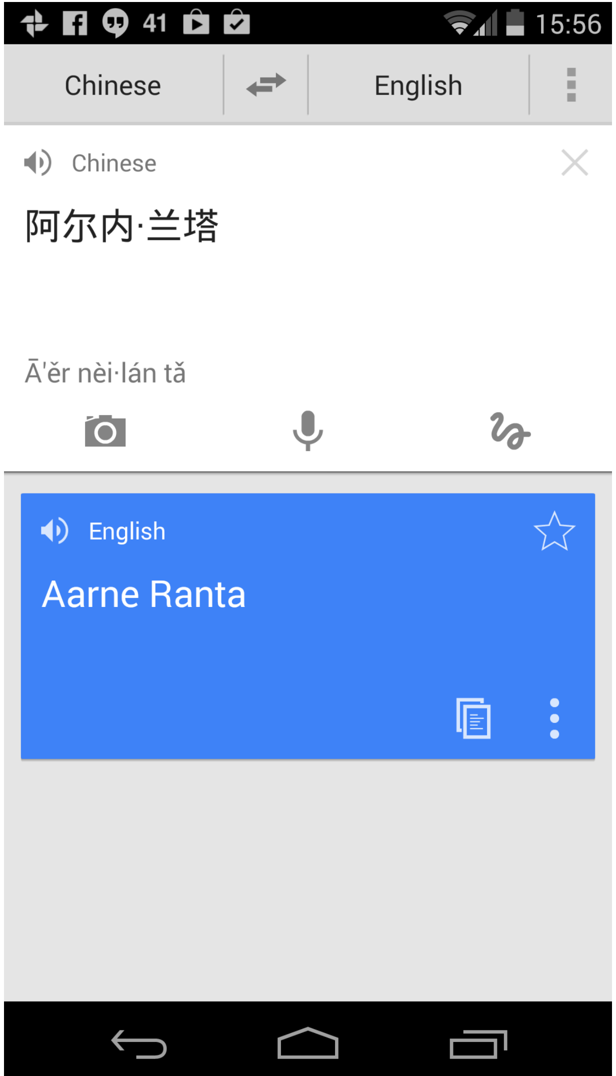
- we can use this data in grammars

NB: Google translate is usually better than GF!

Acquiring data

can be *very* efficient. The Chinese transliteration of my name was created for the Chinese translation of the GF book by prof. Yan Tian, its translator. I could not find it on the web by Google search, but Google translate can have extracted from our gmail exchange or from a private copy of the manuscript in my Google docs. So this is what I got when I tried to use Google translate to find out what the transliteration means!

Google translate, October 2014



Chinese ⇄ English

Chinese

阿尔内·兰塔

Ā'ěr nèi·lán tǎ

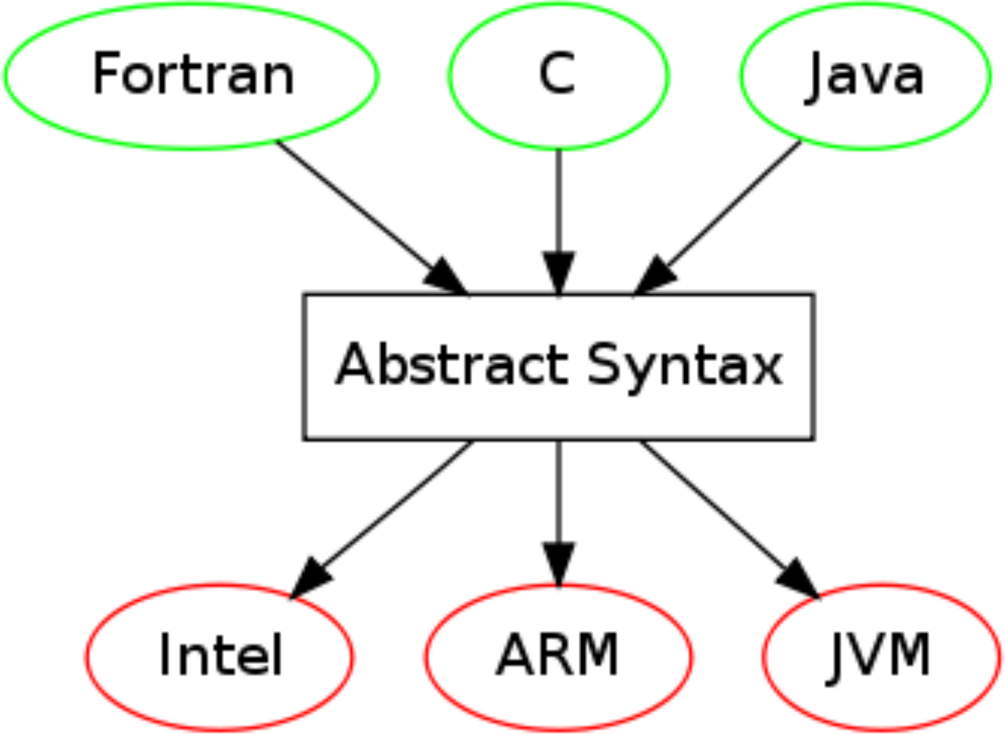
English

Aarne Ranta

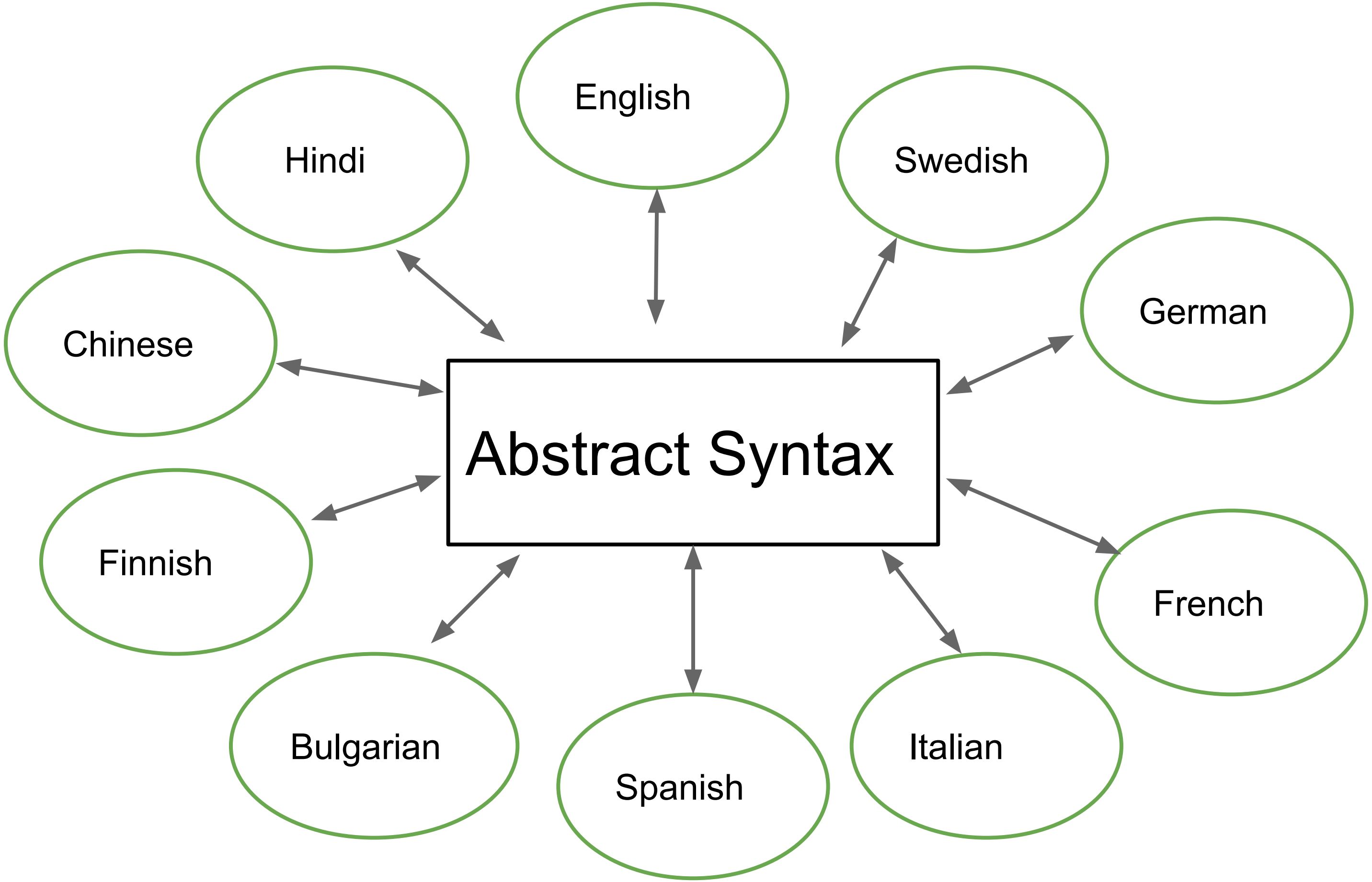
How to do it in GF?

a brief summary

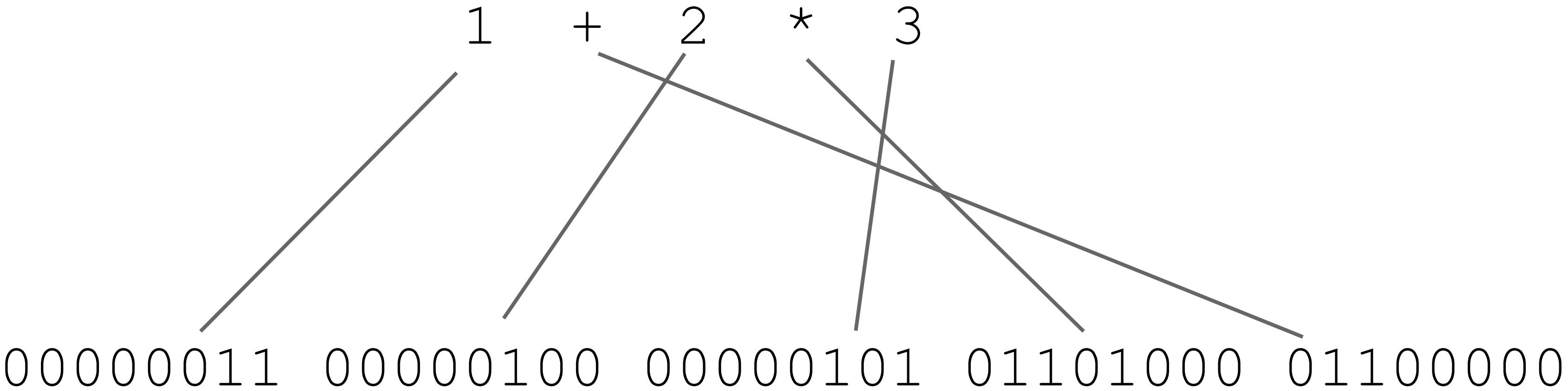
Translation model: multi-source multi-target compiler



Translation model: multi-source multi-target compiler-**decompiler**



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

abstrakt	**Java**	**JVM**
Add x y	*x* “+” *y*	*x y* “01100000”
Mul x y	*x* “*” *y*	*x y* “01101000”
E1	“1”	“00000011”
E2	“2”	“00000100”
E3	“3”	“00000101”

Compiling natural language

Abstract syntax

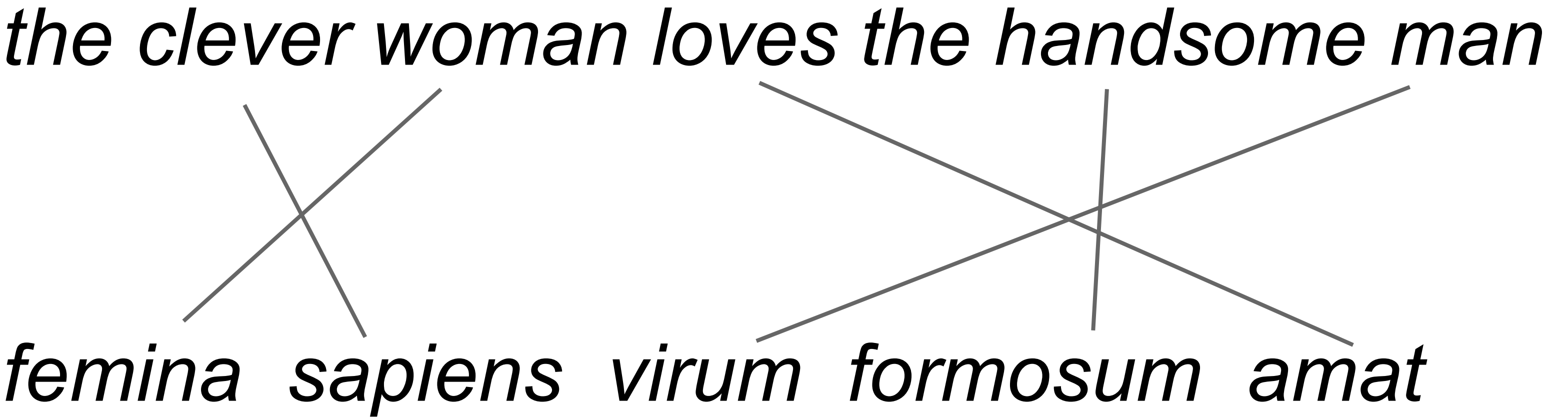
Pred : NP -> V2 -> NP -> S

Mod : AP -> CN -> CN

Love : V2

Concrete syntax:	**English**	**Latin**
Pred s v o	*s v o*	*s o v*
Mod a n	*a n*	*n a*
Love	*“love”*	*“amare”*

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

	English	**Latin**
CN	*{s : Number => Str}*	*{s : Number => Case => Str ; g : Gender}*
AP	*{s : Str}*	*{s : Gender => Number => Case => Str}*

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”)

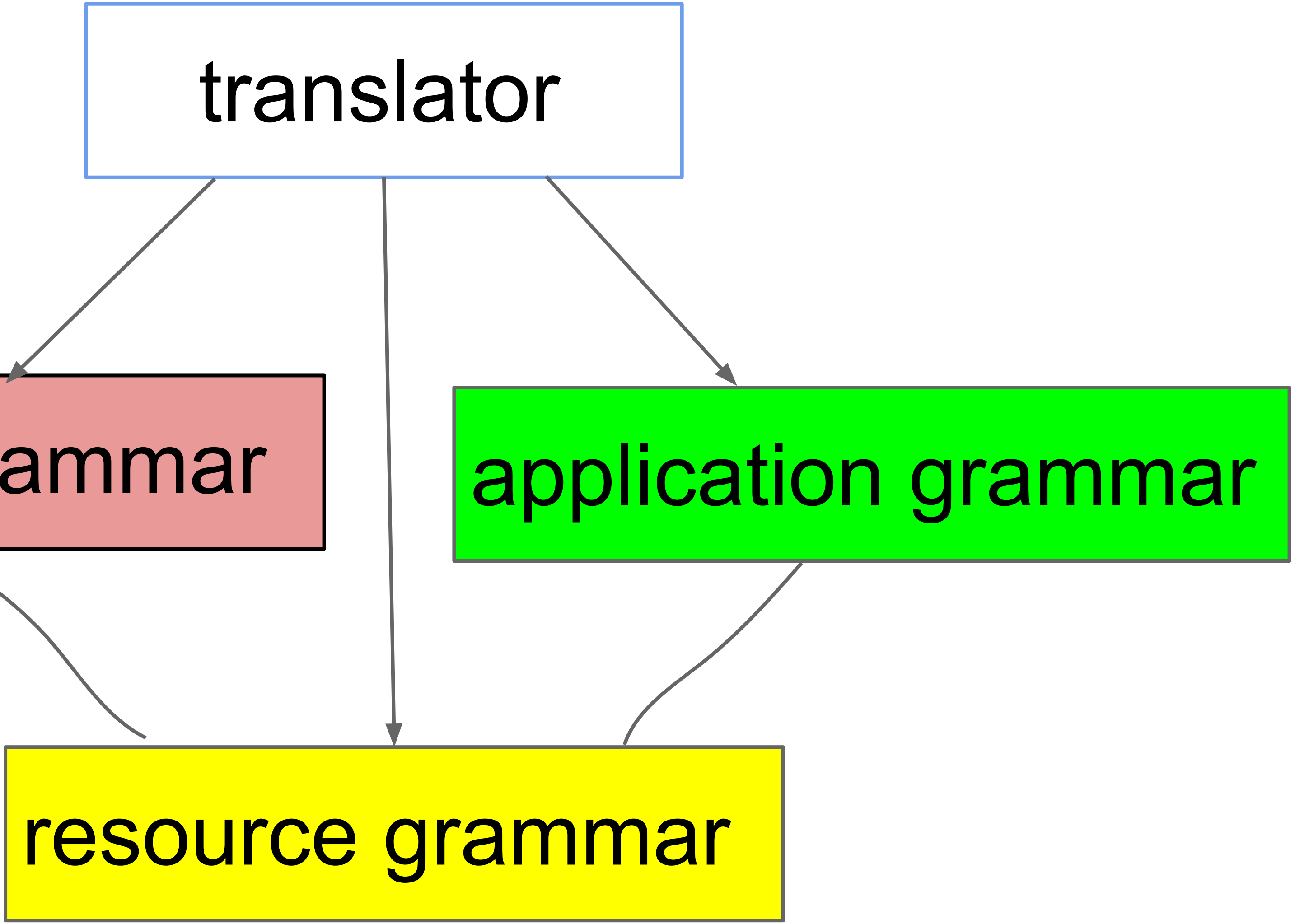
Abstract syntax trees

my name is John

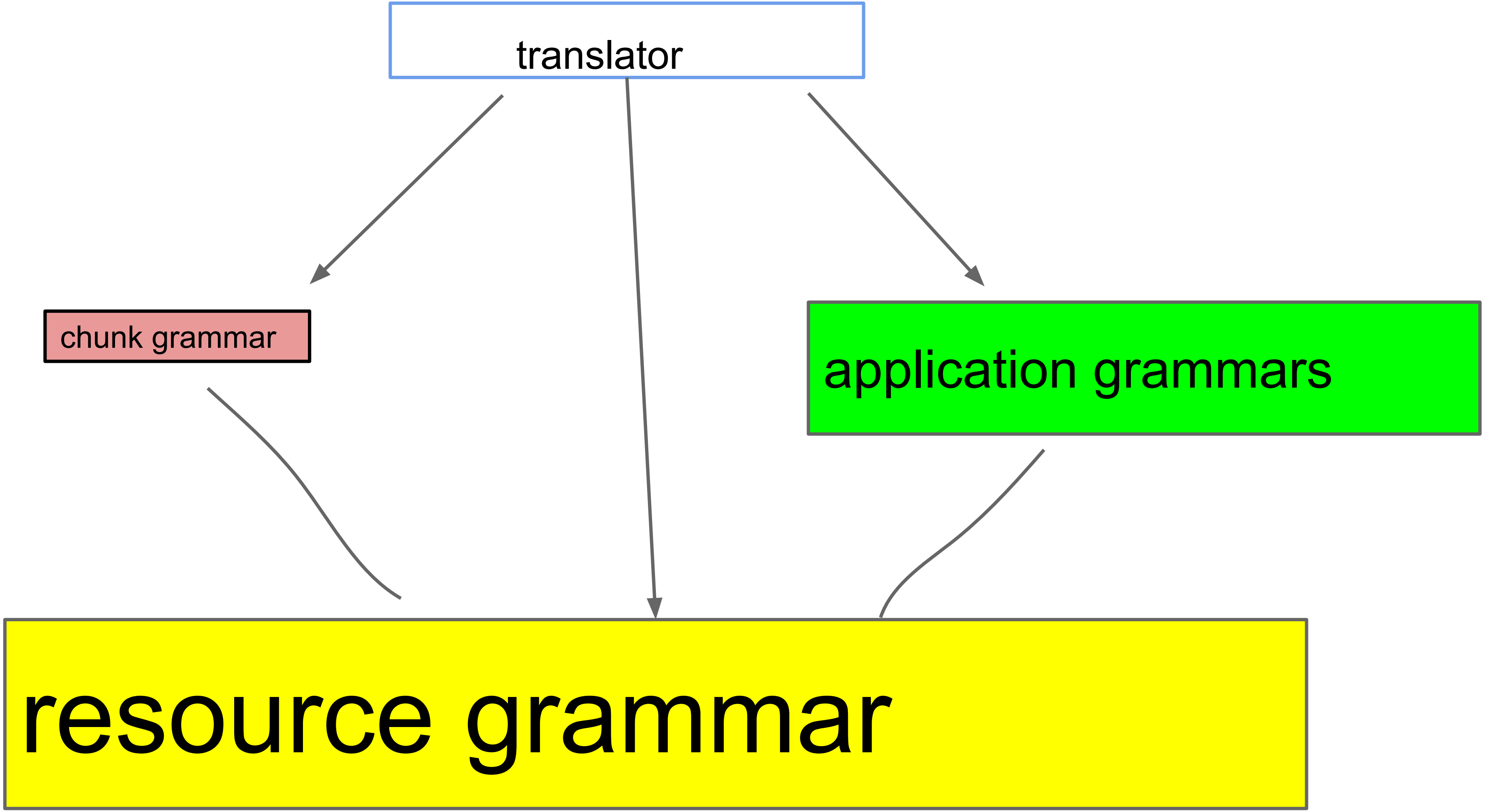
HasName I (Name “John”)

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

[DetChunk (Poss i_NP), NChunk name_N, copulaChunk, NPChunk (NameNP “John”)]


translator
chunk grammar
application grammar
resource grammar

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

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

chunk grammar

words

suitable word sequences

- local agreement
- local reordering

easily derived from resource grammar

easily varied

minimize hand-hacking

translator

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

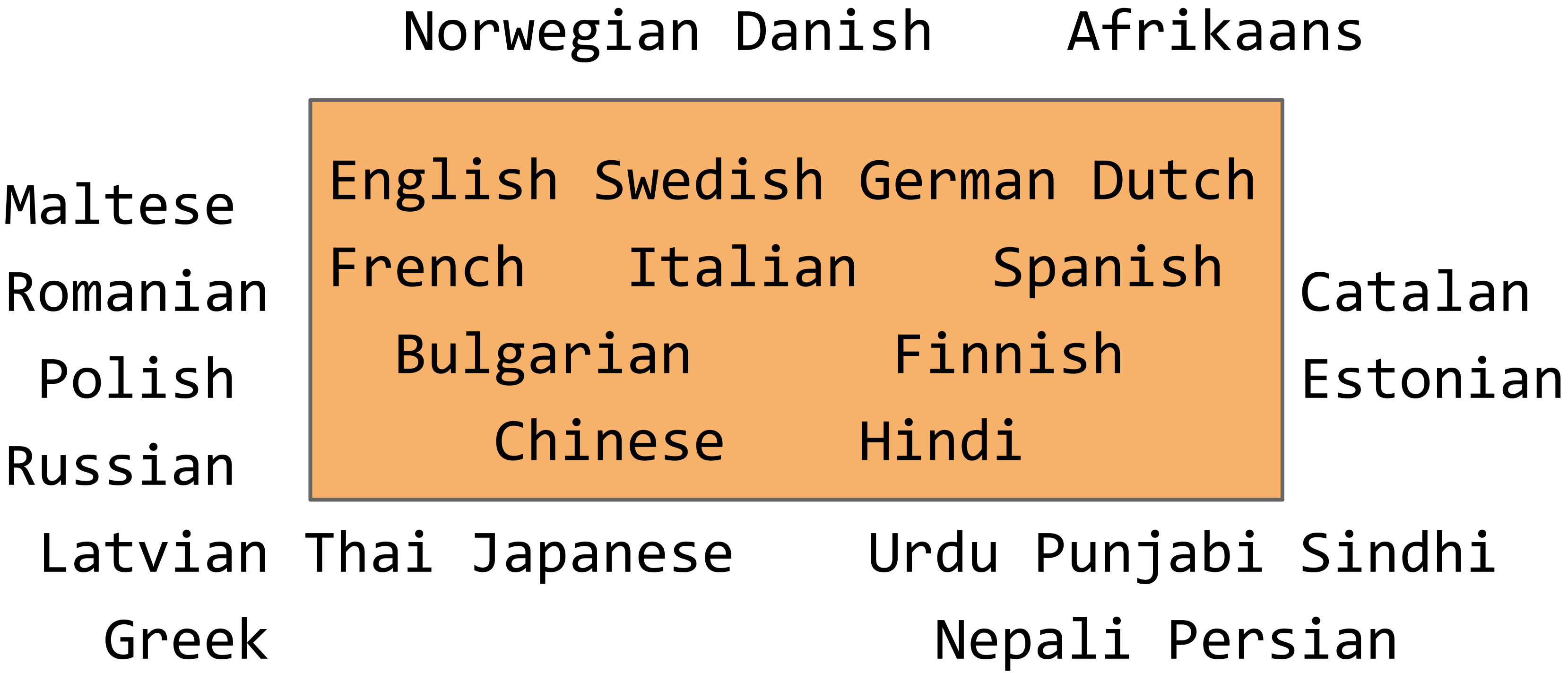
Expressivity: easy to express complex rules

- 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 Finnish

Chinese Hindi

Maltese

Romanian

Polish

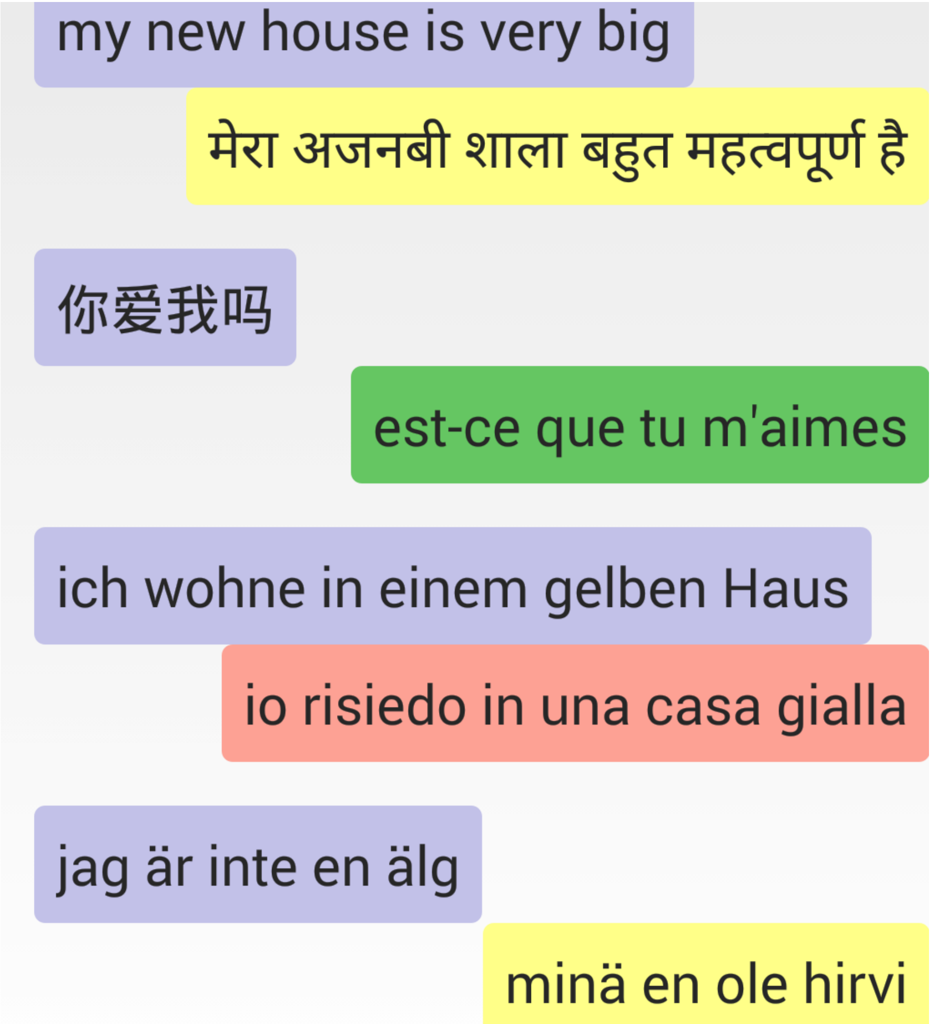
Russian

Catalan

Estonian

Latvian Thai Japanese Urdu Punjabi Sindhi

Greek 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

Demos

Demo 1: MOLTO Phrasebook

Source: **controlled language input**

Always green

Based on **domain semantics**

http://www.grammaticalframework.org/demos/phrasebook/

Demo 2: resource grammar

Source: **predictive input**

Always yellow

Based on **syntactic structure**

http://cloud.grammaticalframework.org/minibar

Demo 3: wide-coverage translation

Source: any text

Can be green, yellow, or red.

Based on **semantics, grammar, or chunks.**

http://cloud.grammaticalframework.org/wc.html

Demo 4: mobile translation app

Source: **text or speech** in any language

Can be green, yellow, or red.

Based on **semantics, grammar,** or **chunks.**

https://play.google.com/store/apps/details?id=org.grammaticalframework.ui.android

http://www.grammaticalframework.org/~aarne/App11.apk

How to do it?

some more details

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 (Konstruktikon)

Extraction from SMT phrase tables

- **example-based grammar writing**

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 heissen_V2 y
    ich heisse 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 a new language is a matter of a few hours

MOLTO’s goal was to make this possible.

- EU project 2010-2013: Multilingual Online Translation

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)
    - Reliable Multilingual Translation, Swedish Research Council project 2013-2017

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 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:

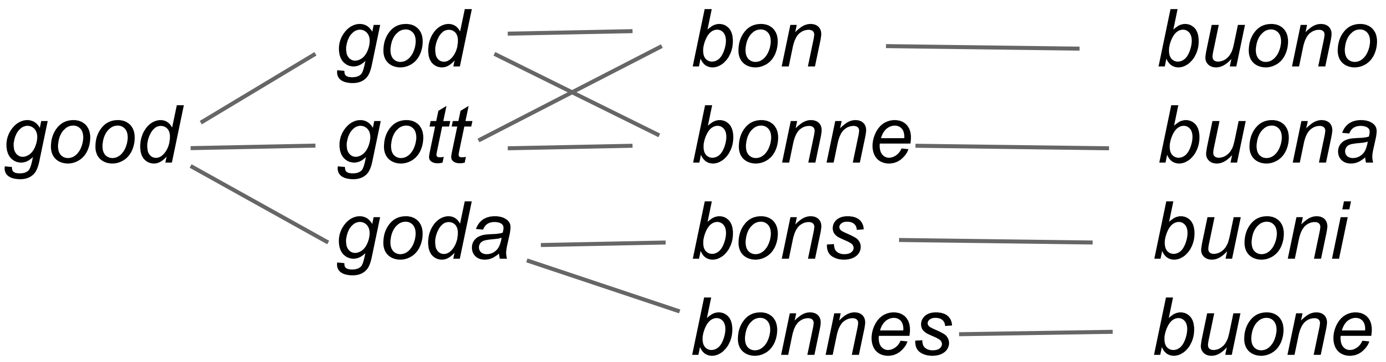
[this yellow house] - [det här gula huset]

[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.



good

god

gott

goda

bon

bonne

bons

bonnes

buono

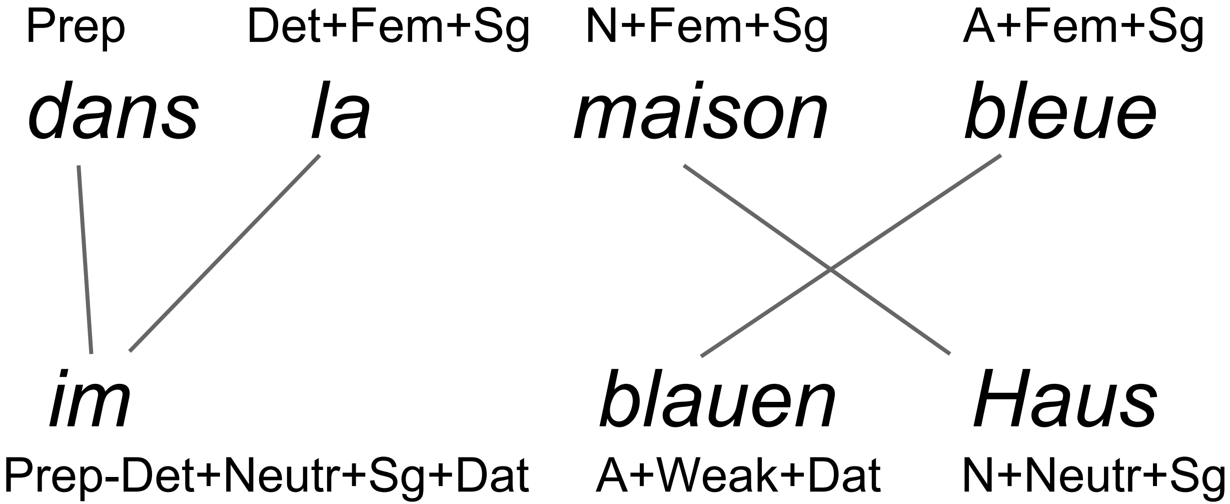
buona

buoni

buone

Apertium does this for every language pair.

Resource grammar chunks of course come with reordering and internal agreement



Prep *dans* Det+Fem+Sg *la* N+Fem+Sg *maison* A+Fem+Sg *bleue*

im Prep-Det+Neutr+Sg+Dat *blauen* A+Weak+Dat *Haus* N+Neutr+Sg

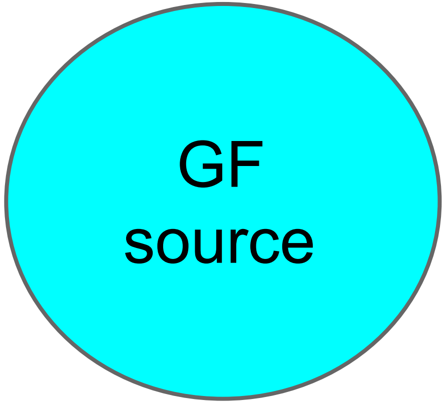
Recall: chunks are just a by-product of the real grammar.

Their size span is

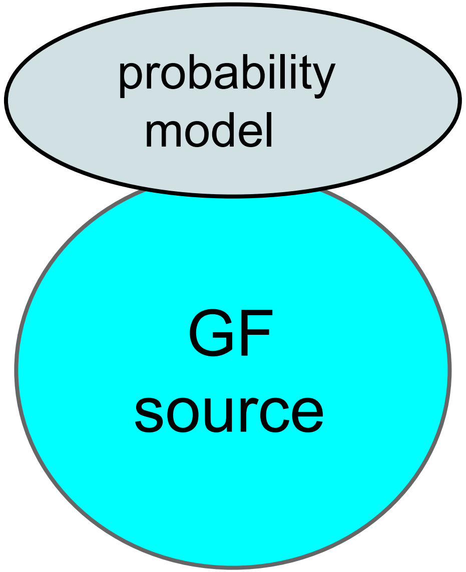
single words <---> entire sentences

A wide-coverage chunking grammar can be built in a couple of hours **by using the RGL**.

Building the translation system

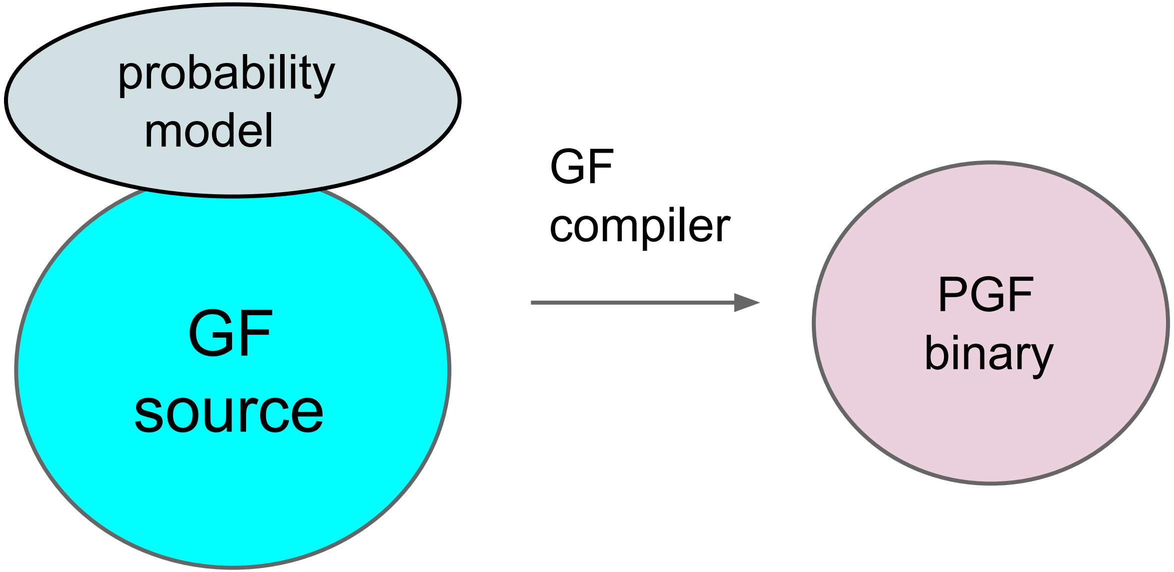


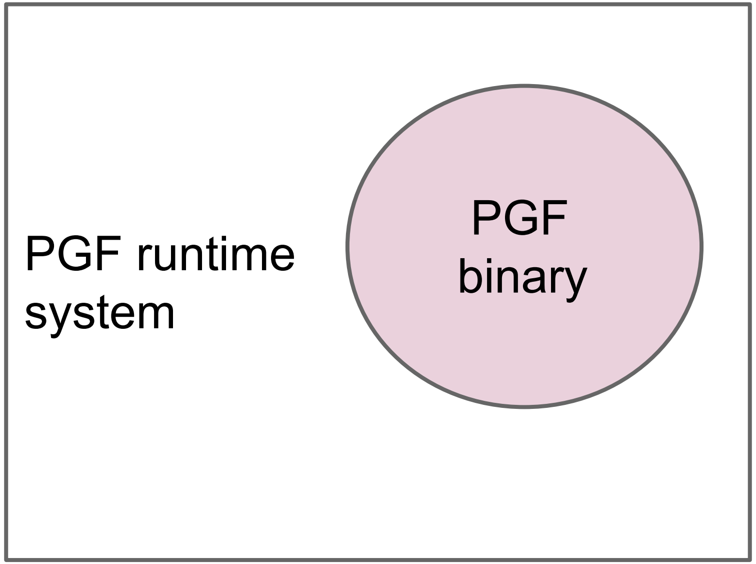
GF
source



probability model

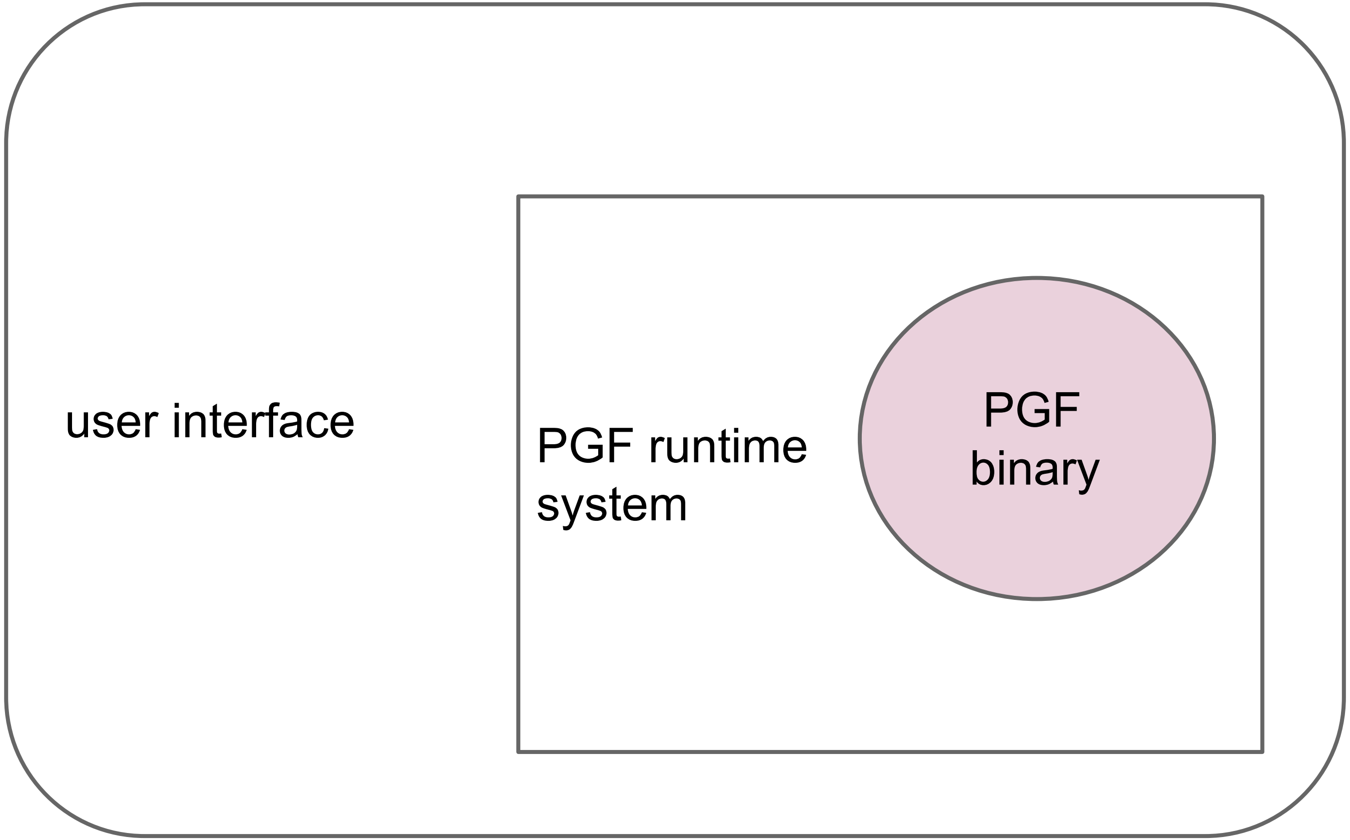
GF source


probability model
GF source
GF compiler
PGF binary

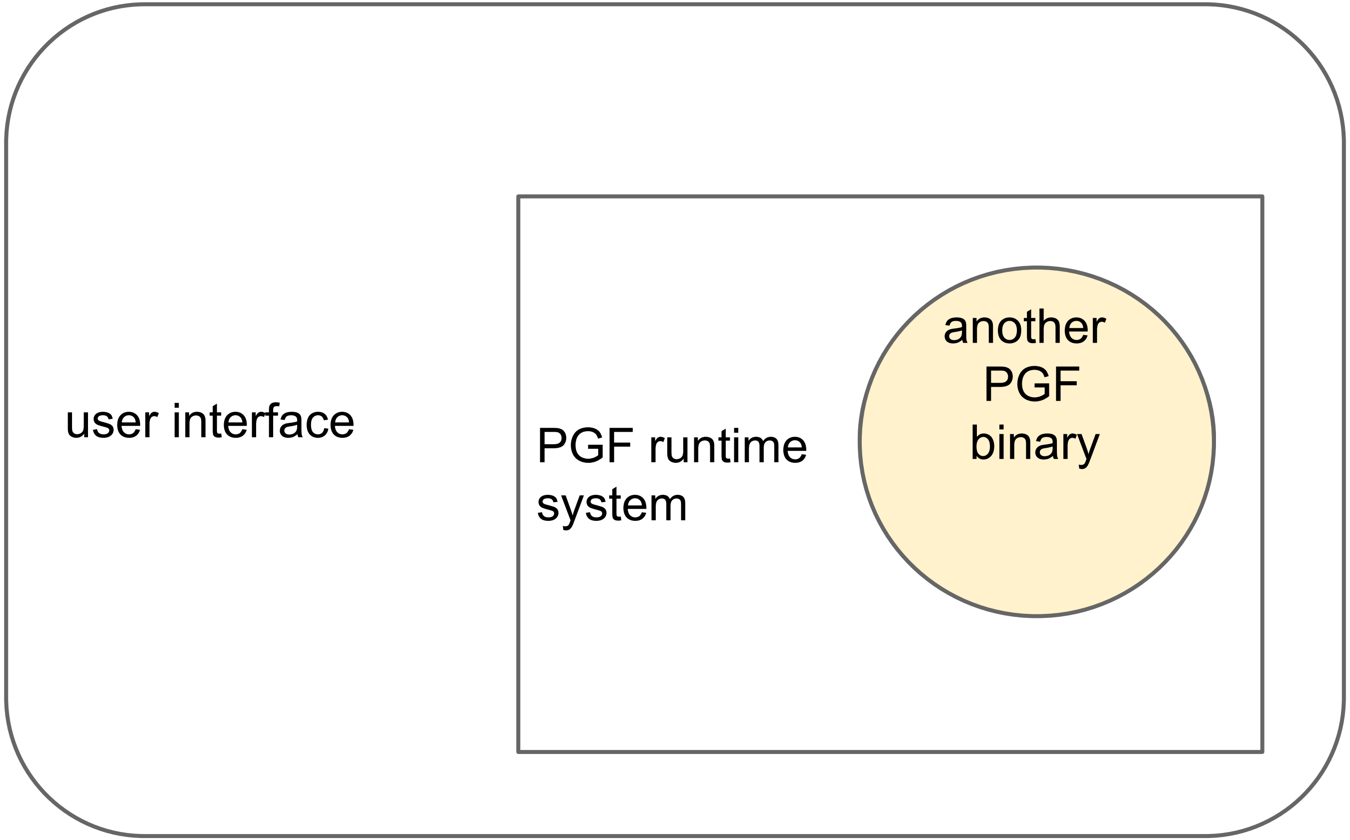


PGF runtime system

PGF binary



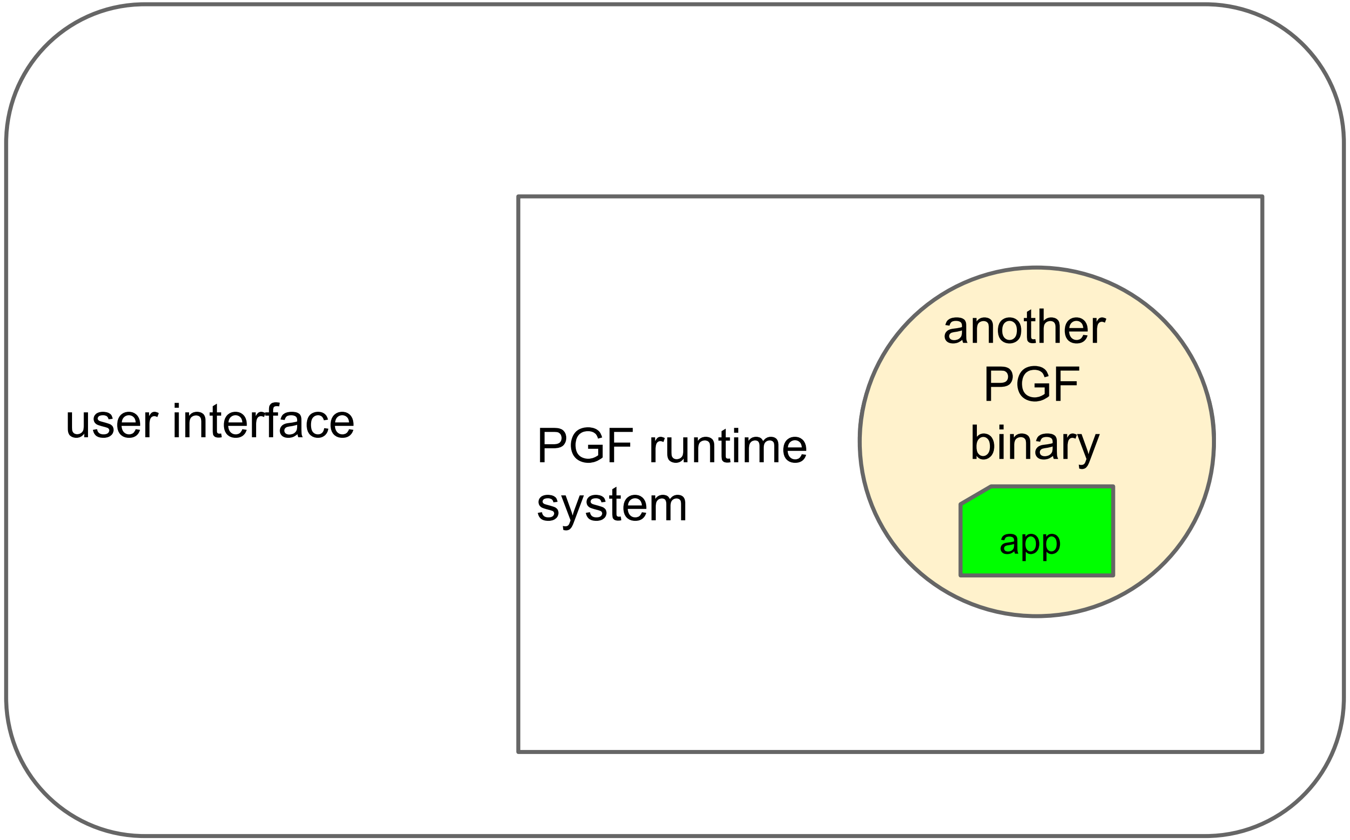
user interface
PGF runtime system
PGF binary



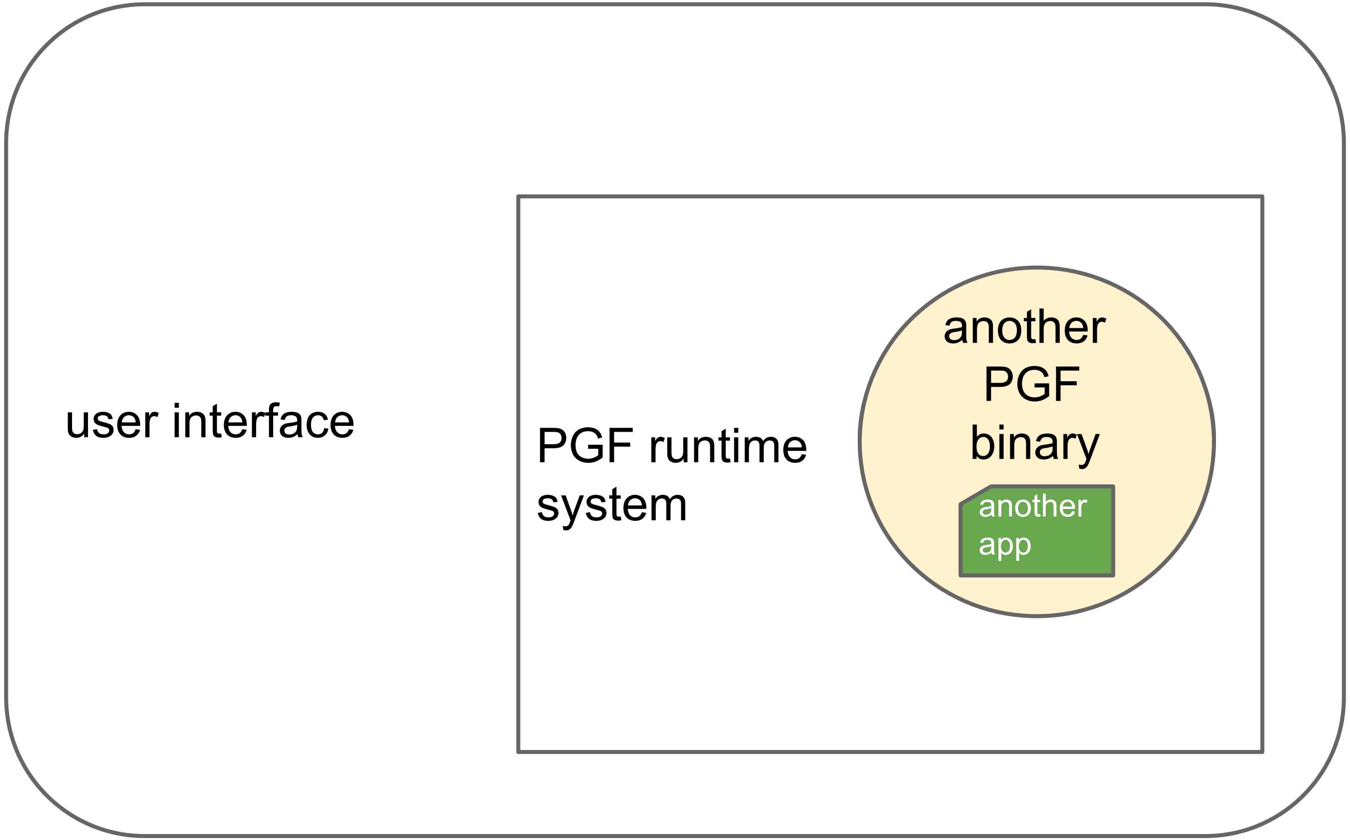
user interface

PGF runtime system

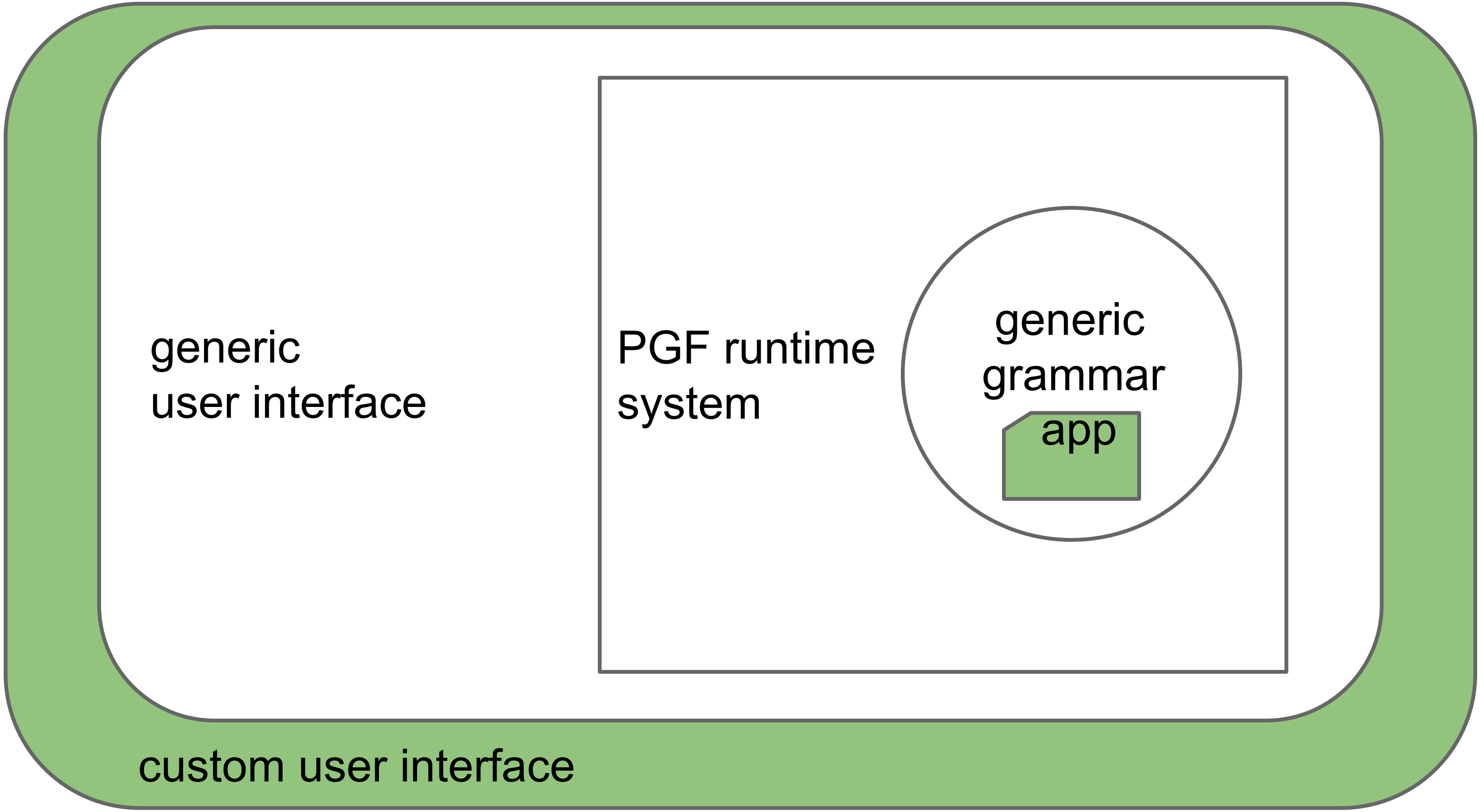
another PGF binary



user interface
PGF runtime system
another PGF binary
app


user interface
PGF runtime system
another PGF binary
another app

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



User interfaces

command-line

shell

web server

web applications

mobile applications

Agenda for future work

Improve the lexicon

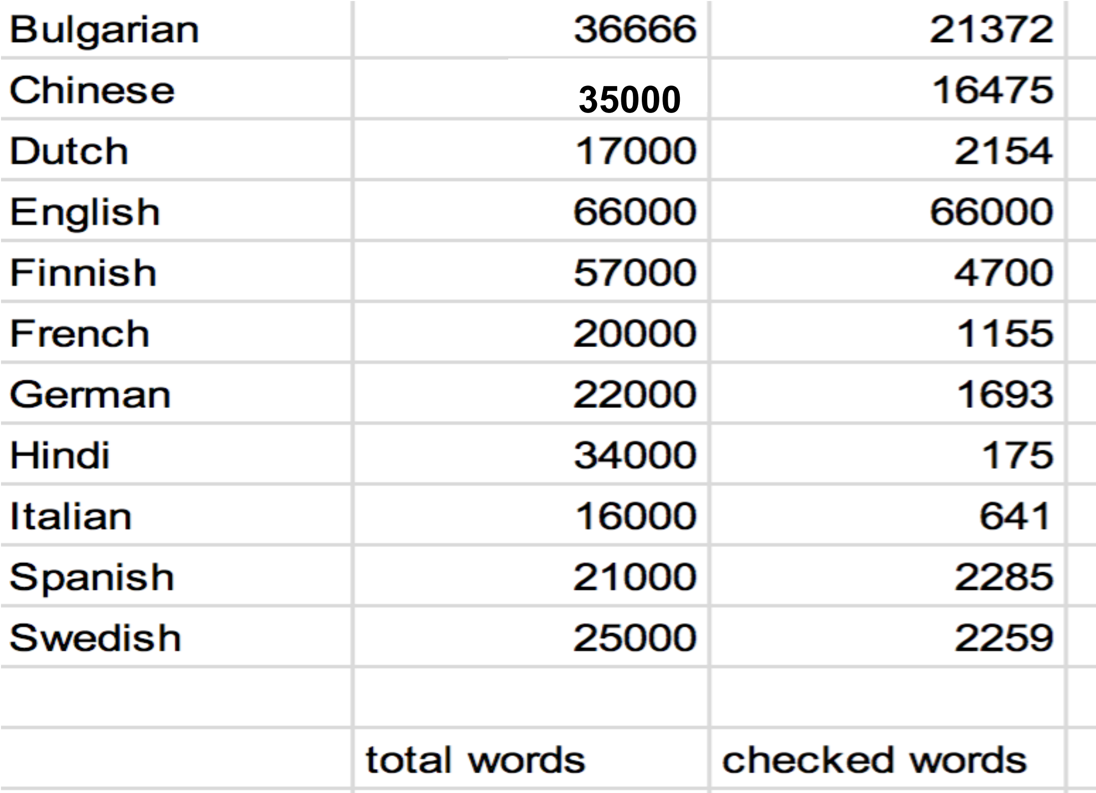
Split senses

Improve disambiguation

Introduce constructions

Design and perform evaluation

Current dictionary coverage



Bulgarian	36666	21372
Chinese	35000	16475
Dutch	17000	2154
English	66000	66000
Finnish	57000	4700
French	20000	1155
German	22000	1693
Hindi	34000	175
Italian	16000	641
Spanish	21000	2285
Swedish	25000	2259
	total words	checked words

Splitting senses

time

Splitting senses

time_N

time_V

Splitting senses

Zeit

time_N

Mal

Splitting senses

time_1_N	Zeit
time_2_N	Mal

Splitting senses

time_1_N	Zeit	temps
time_2_N	Mal	fois

Splitting senses

weather_N	Wetter	
time_1_N	Zeit	temps
time_2_N	Mal	fois

Disambiguation

Current model, for abstract trees:

$$P(C\ t_1 \ldots t_n) = P(C) * P(t_1) * \ldots * P(t_n)$$

where *P(C)* for each tree constructor *C* is estimated from its frequency in a corpus.

The context-free tree model

Surprisingly good for syntactic constructors

But almost useless for word senses

*This **time** we will have more **time**.*

Alternative models

Run-time (in “decoding”):
verb + arguments “n-grams” (on tree level)

Compile-time (in grammars):
include constructions and multiwords in lexicon

See also: 4th GF Summer School

July 2015 in Marsalforn, Malta