Machine Translation: Green , Yellow , and Red

Aarne Ranta

NLCS/NLSR, Vienna Summer of Logic
18 July 2014

UNIVERSITAS GOTHOBURGENSIS 1891

CLT

REMU

digitalGrammars
Language technology to rely on.

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

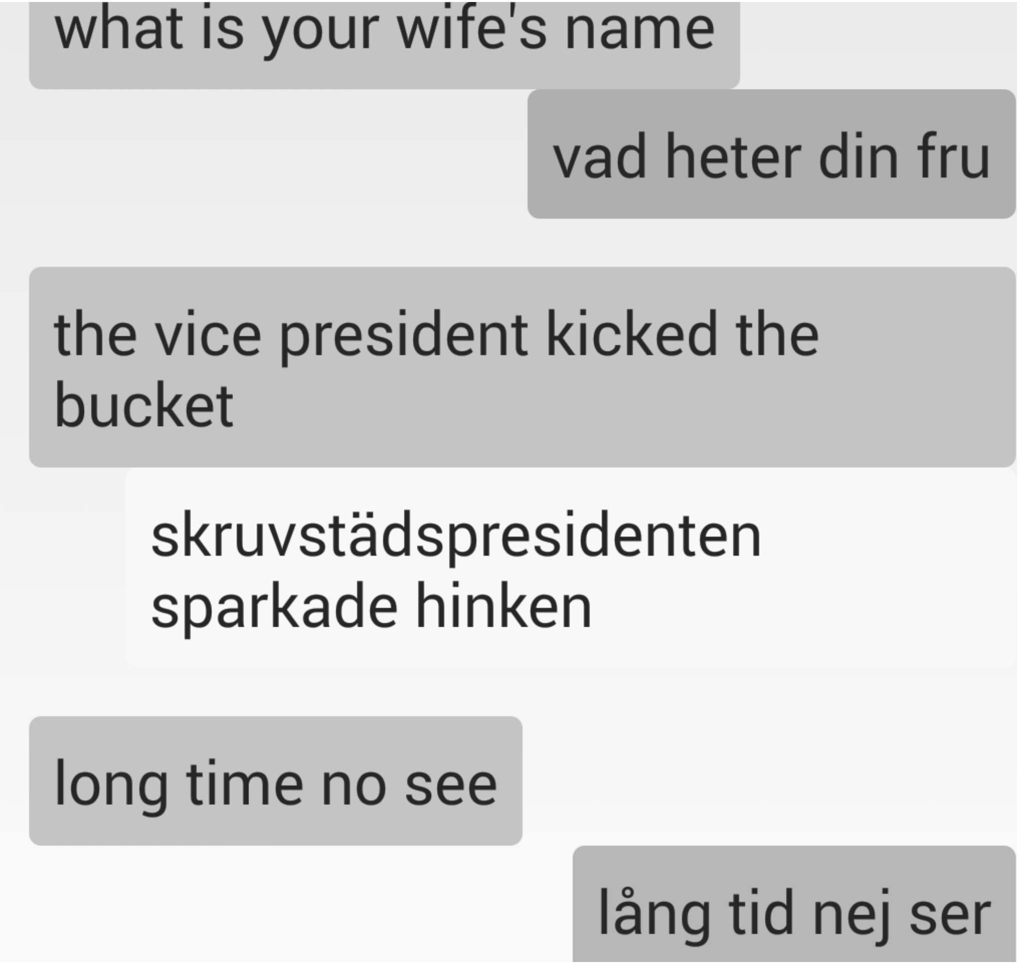
- delivers publication quality in areas where reasonable effort is invested
- degrades gracefully to browsing quality in other areas
- shows a clear distinction between these

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

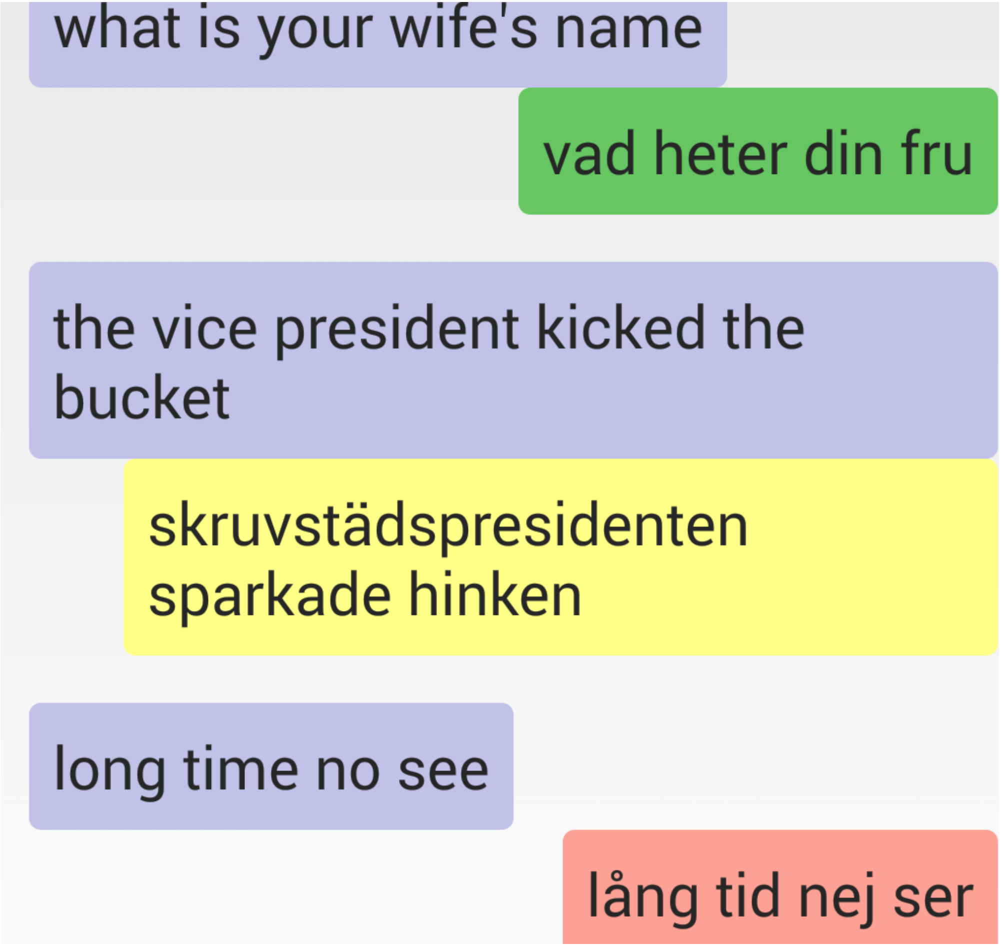
the vice president kicked the bucket

skruvstädspresidenten sparkade hinken

long time no see

lång tid nej ser

GF translation app in greyscale



what is your wife's name

vad heter din fru

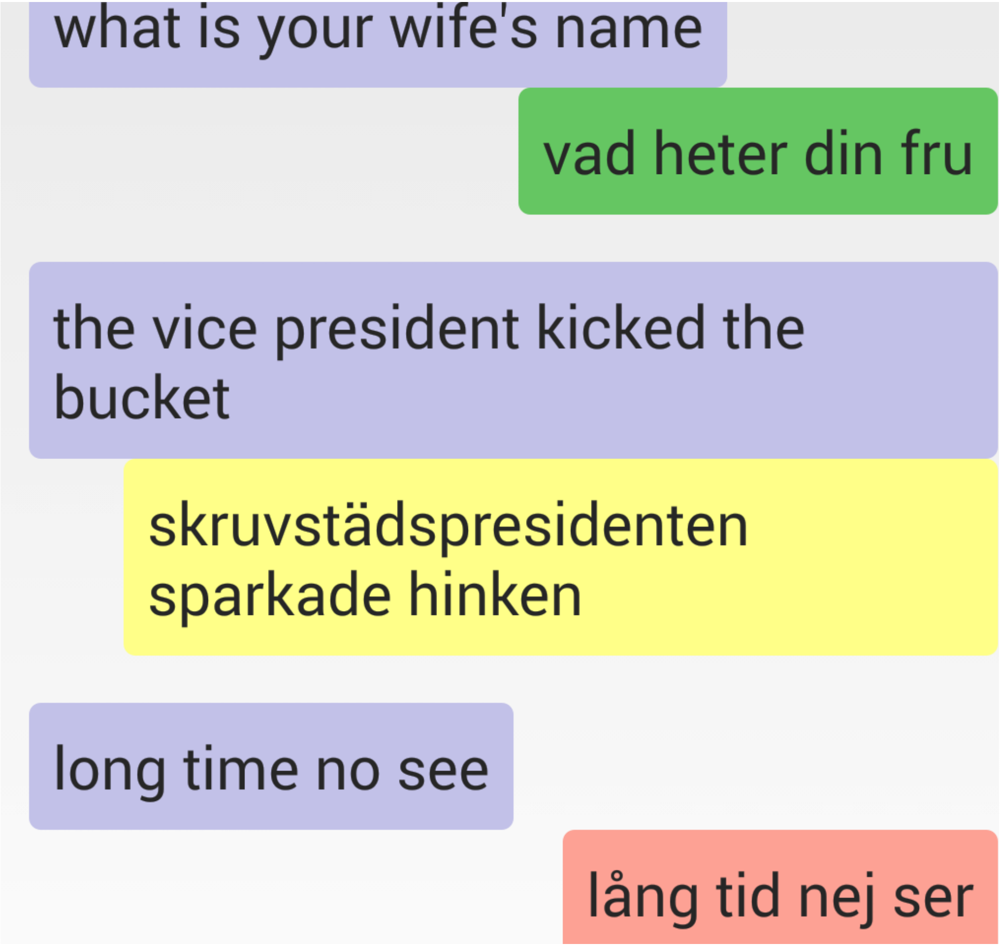
the vice president kicked the bucket

skruvstädspresidenten sparkade hinken

long time no see

lång tid nej ser

GF translation app in full colour



what is your wife's name

vad heter din fru

the vice president kicked the bucket

skruvstädspresidenten sparkade hinken

long time no see

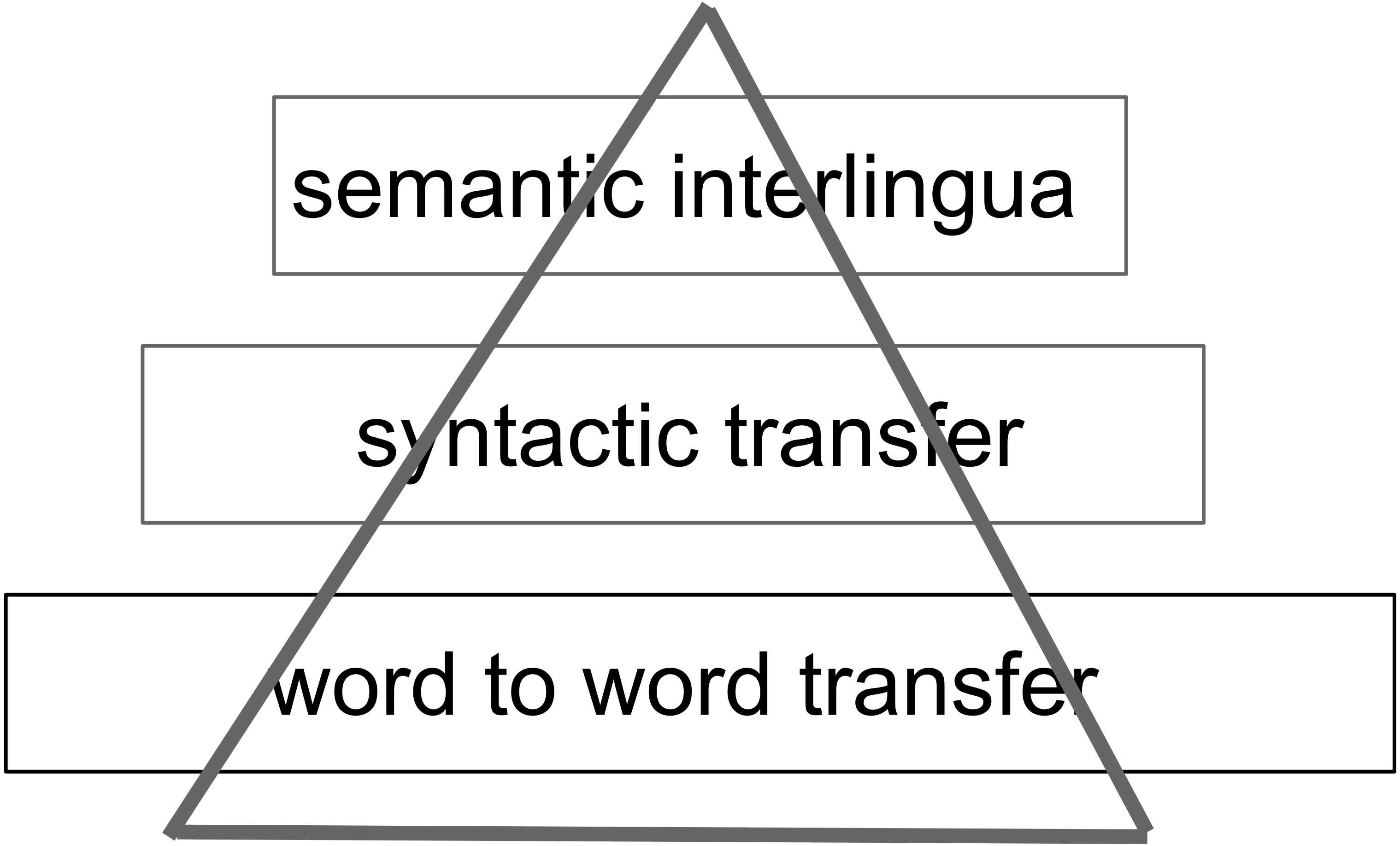
lång tid nej ser

translation by **meaning**
- correct
- idiomatic

translation by **syntax**
- grammatical
- often strange
- often wrong

translation by **chunks**
- probably ungrammatical
- probably wrong

The Vauquois triangle

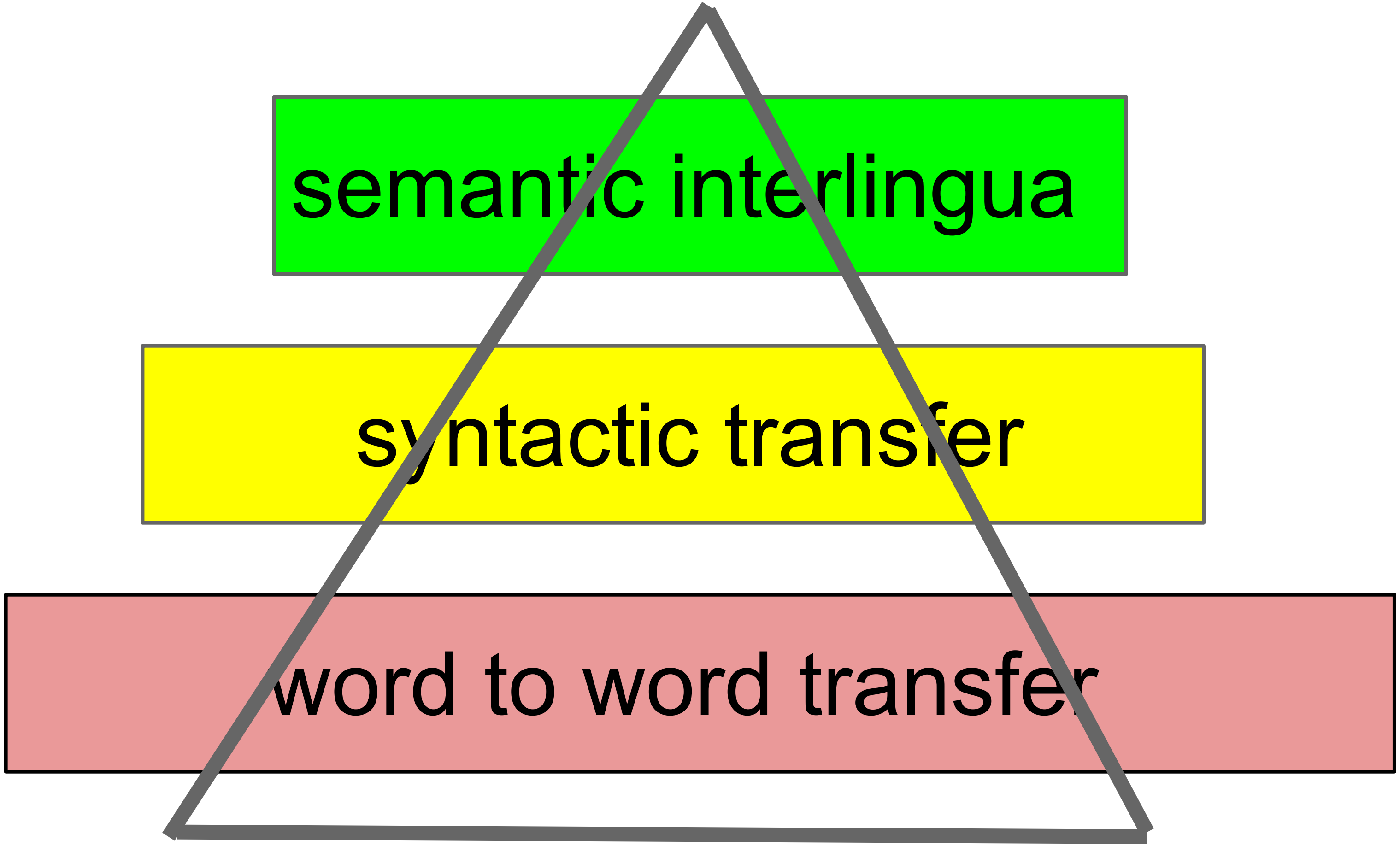


semantic interlingua

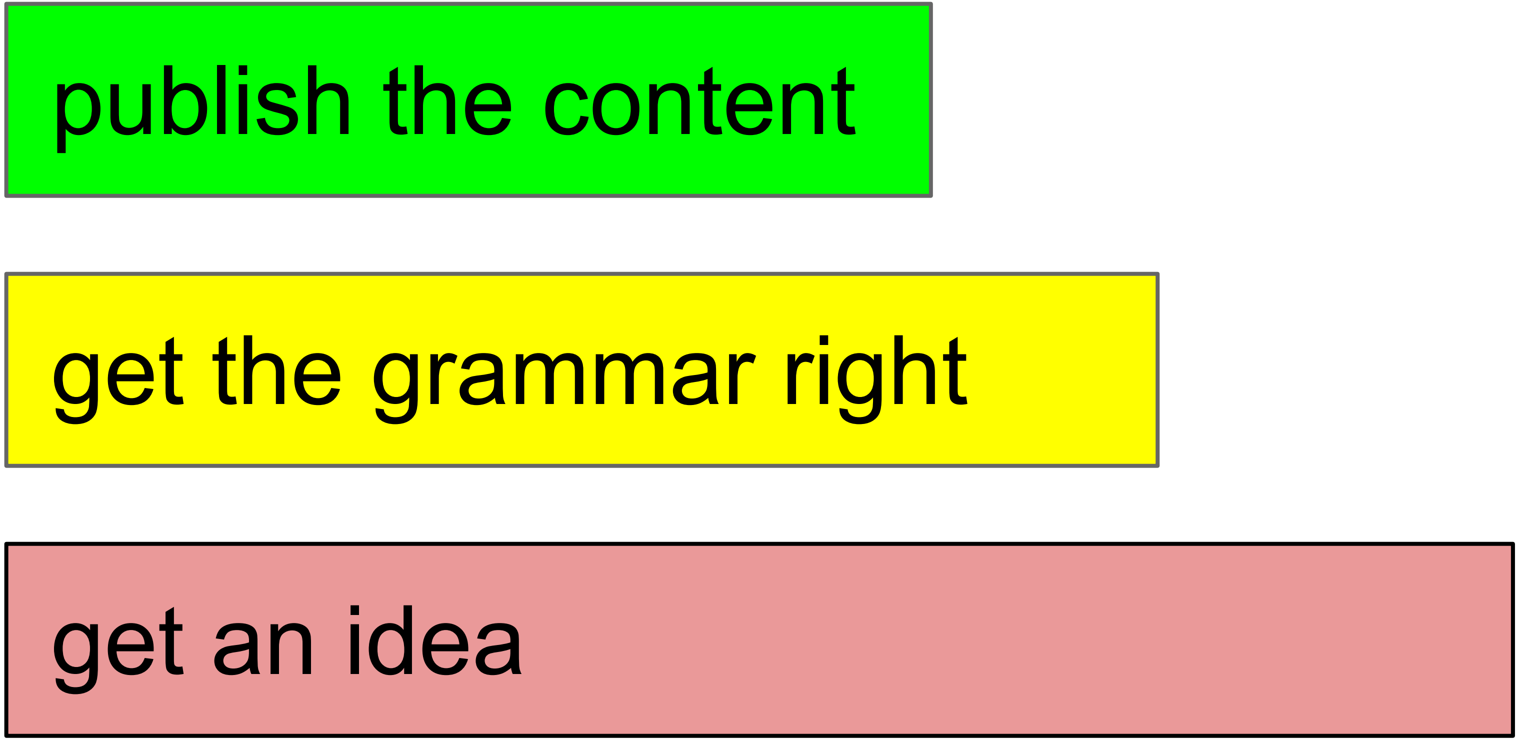
syntactic transfer

word to word transfer

The Vauquois triangle



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”

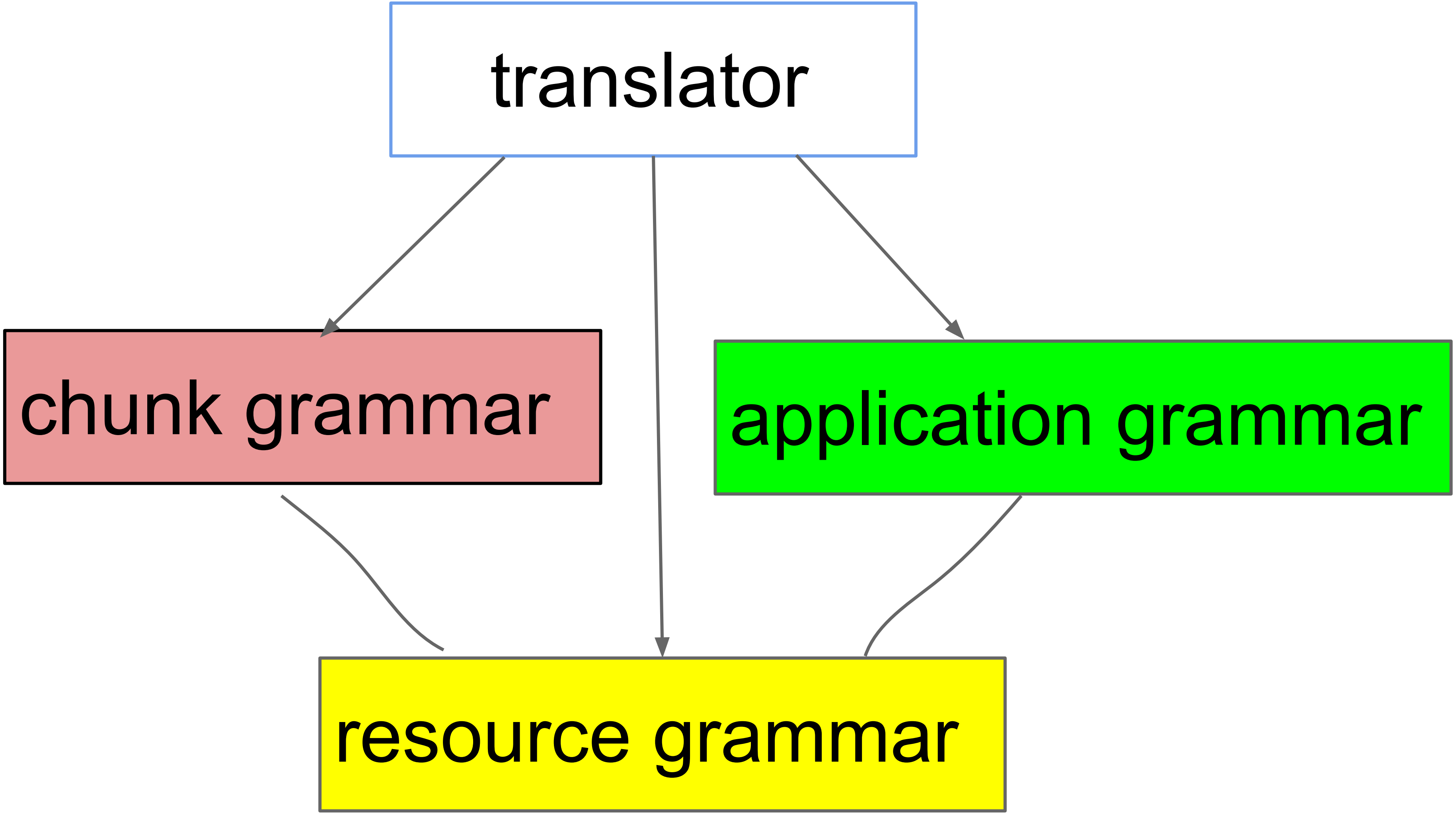
But we do it all in GF!

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
Computational Linguistics 36(4), 2010.

How to do it?

a brief summary



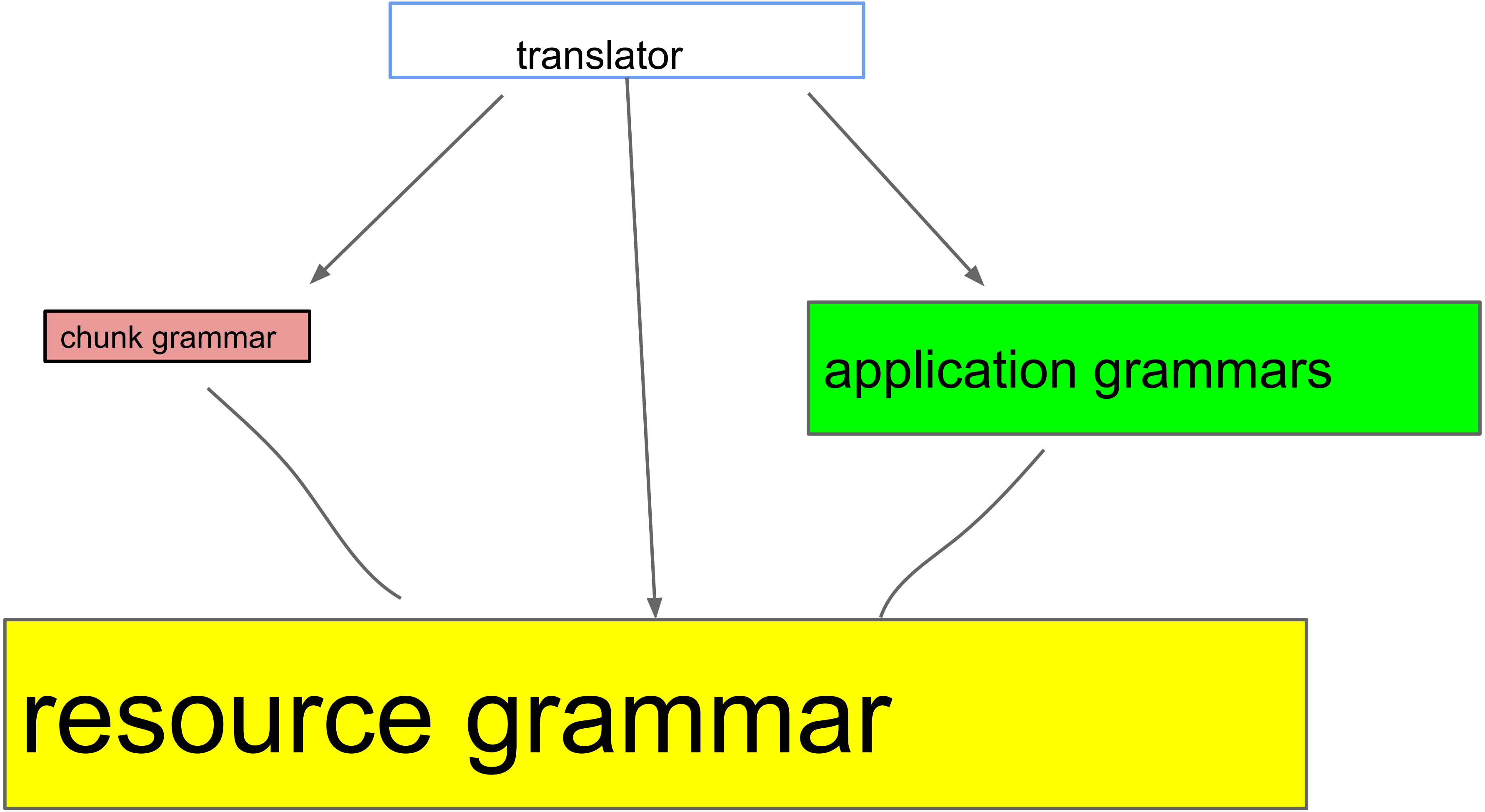
translator

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

chunk grammar

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

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

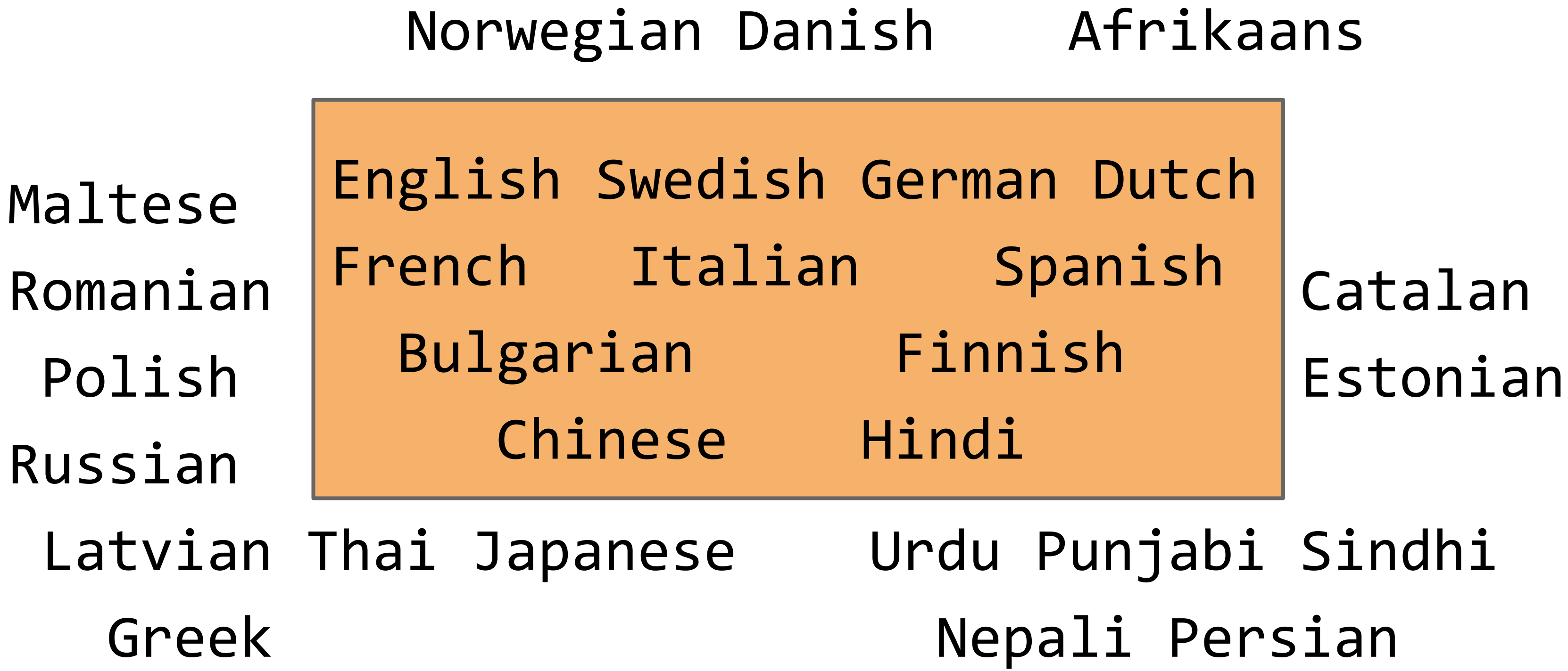
Expressivity: easy to express complex rules

- agreement
- word order
- discontinuity

Abstractions: easy to manage complex code

Interlinguality: easy to add new languages

Resources: basic and bigger



Norwegian Danish Afrikaans

Maltese

Romanian

Polish

Russian

English Swedish German Dutch

French Italian Spanish

Bulgarian Finnish

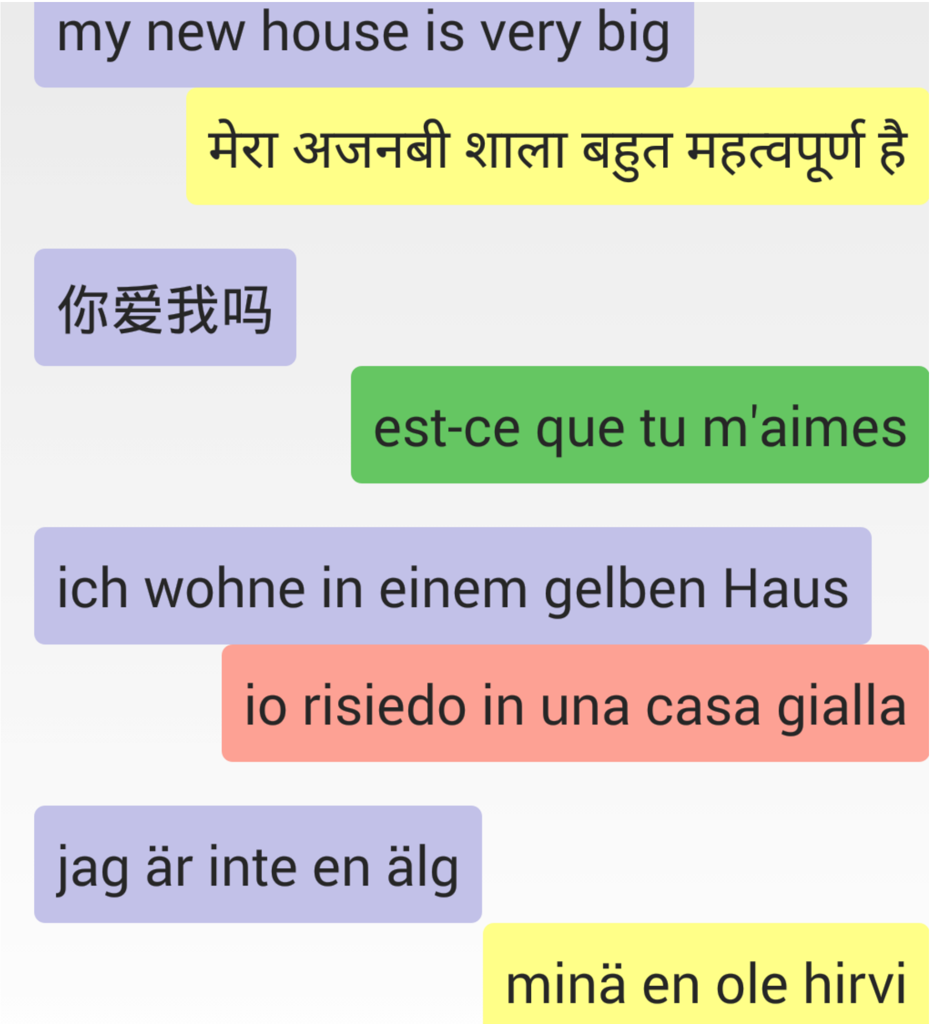
Chinese Hindi

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

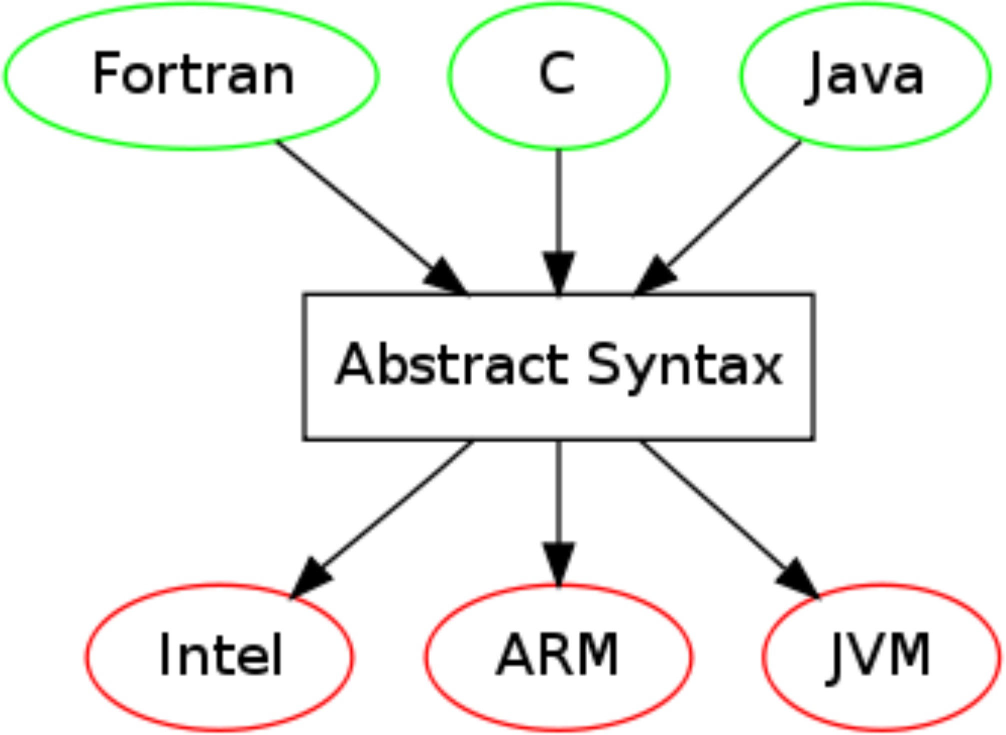
jag är inte en älg

minä en ole hirvi

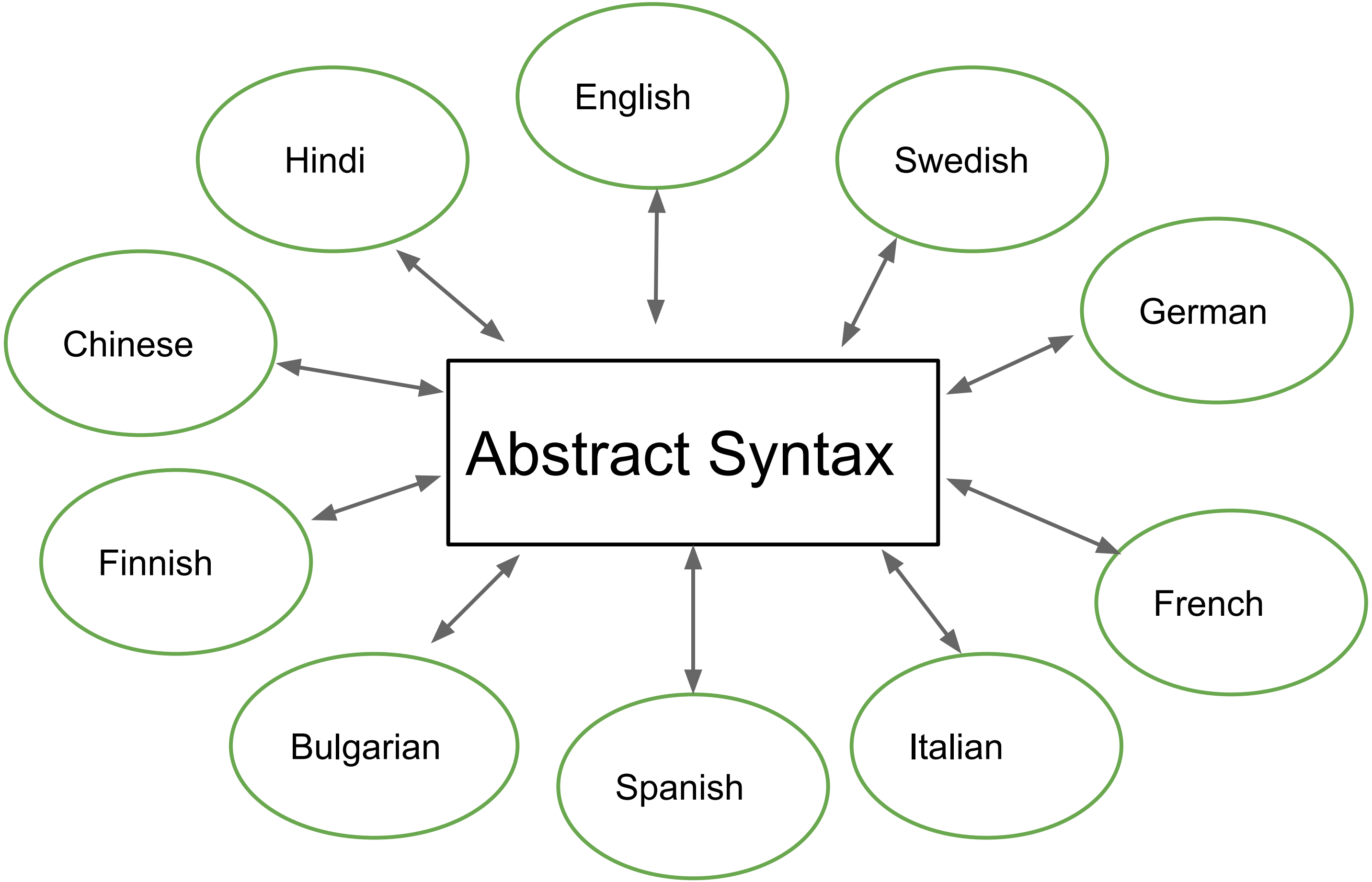
How to do it?

some more details

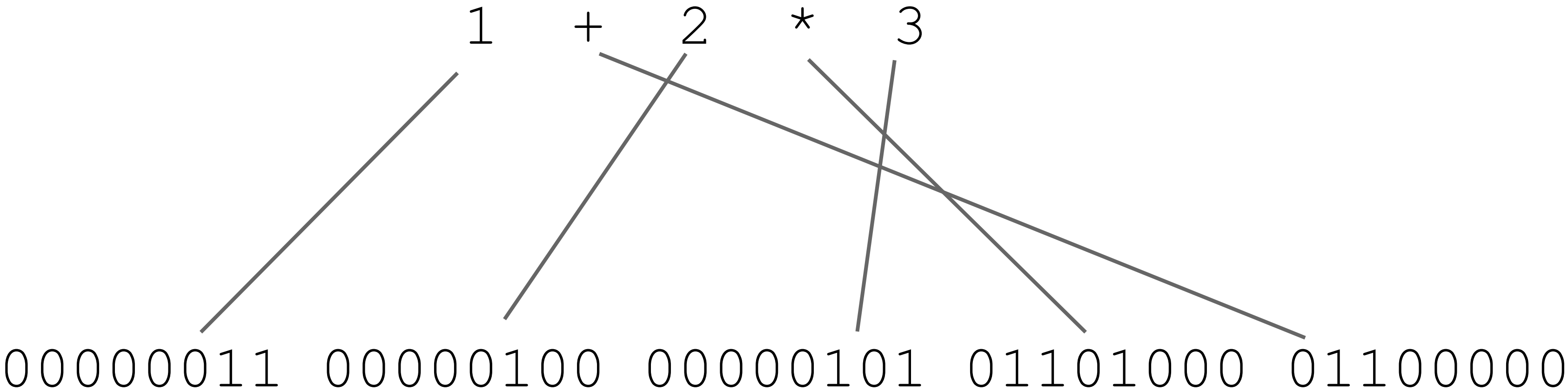
Translation model: multi-source multi-target compiler



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

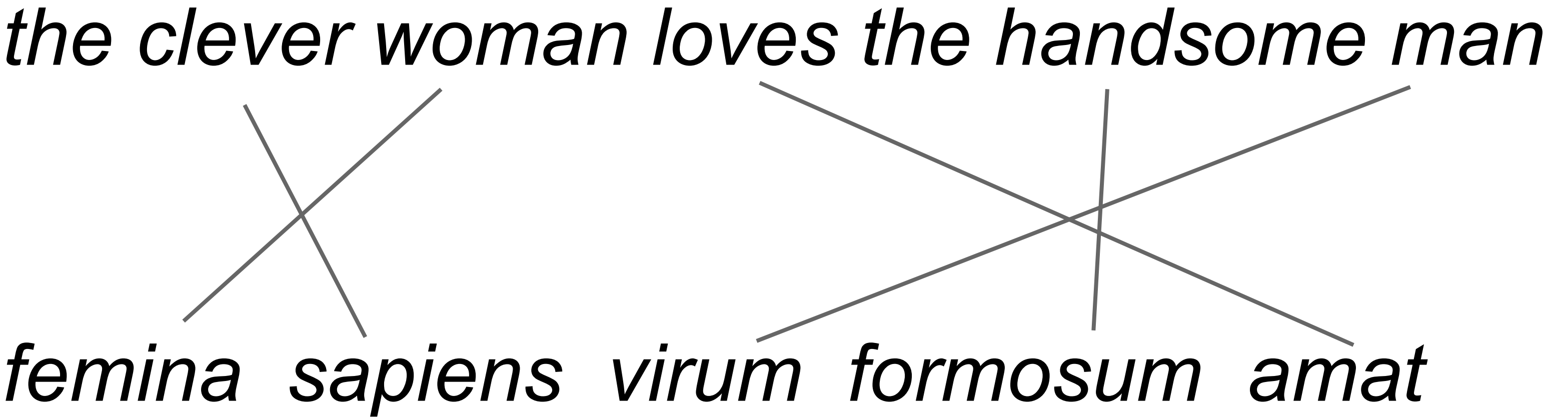
Pred : NP -> V2 -> NP -> S

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”)]

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

```
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:

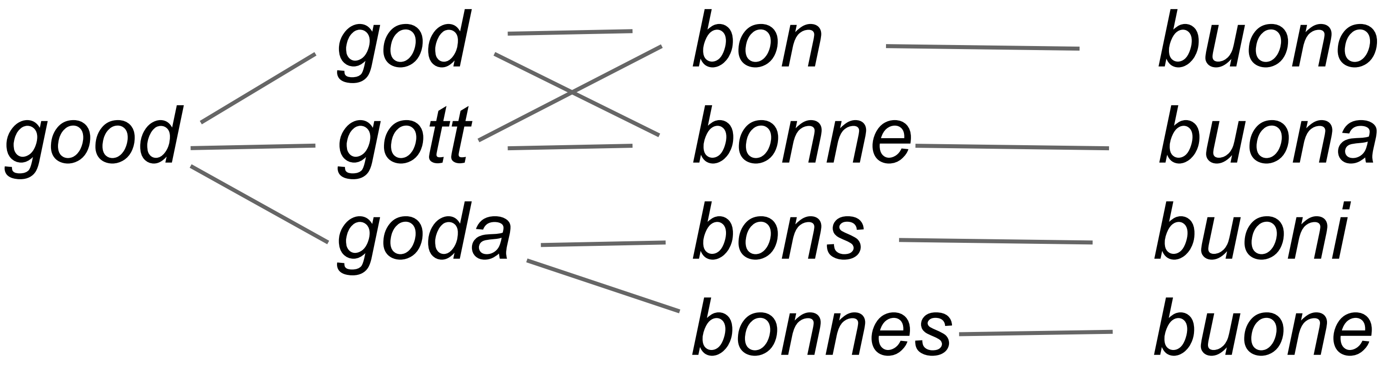
[this yellow house] - [det här gula huset]

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

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

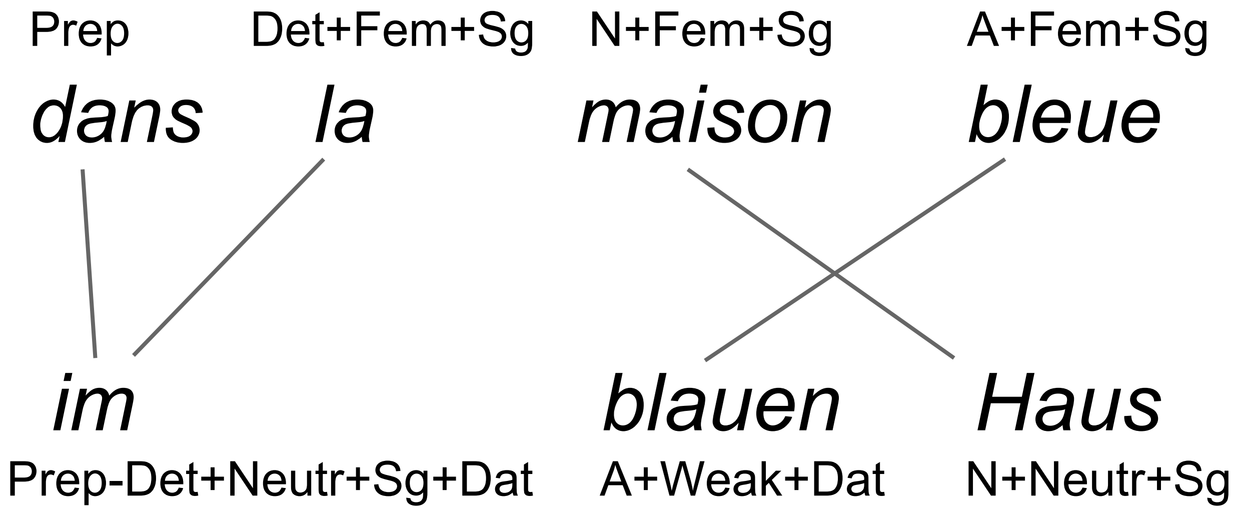
Limiting case: whole sentences as chunks.

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



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**

```
  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

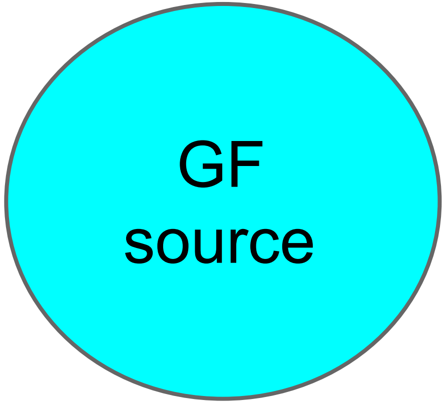
- formal methods help (REMU project)

These grammars are a source of

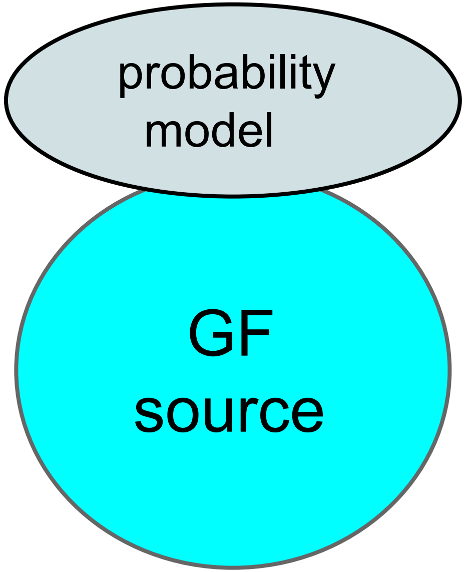
- “non-compositional” translations
- compile-time transfer
- idiomatic language
- translating meaning, not syntax

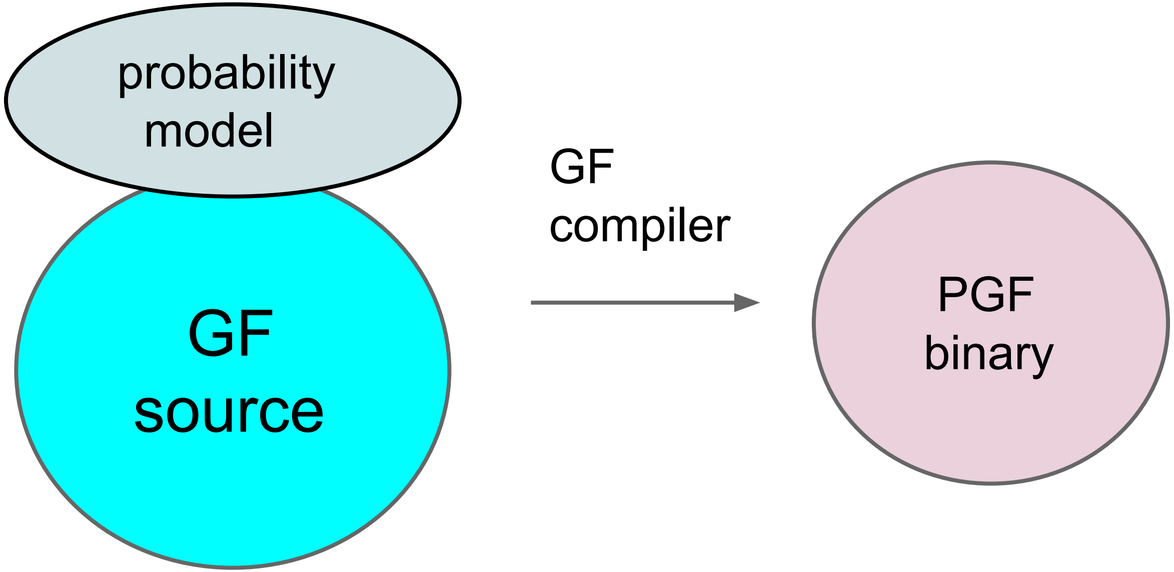
Constructions are the generalized form of this idea, originally domain-specific.

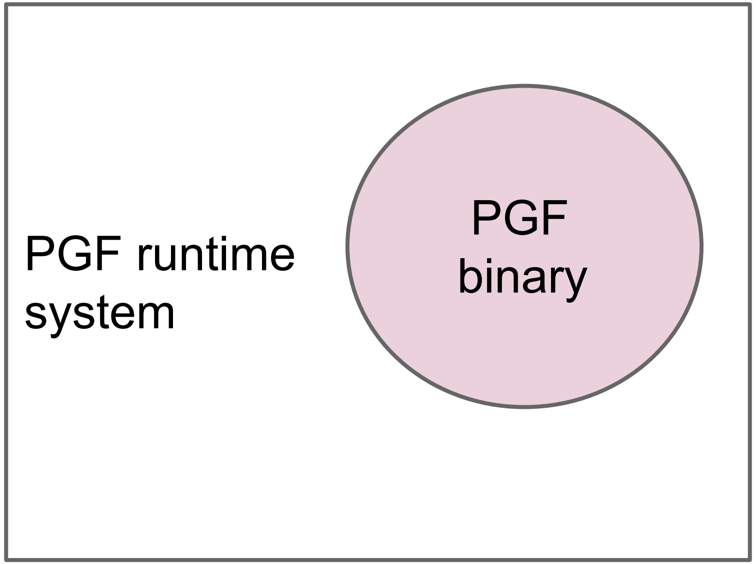
Building the translation system



GF
source

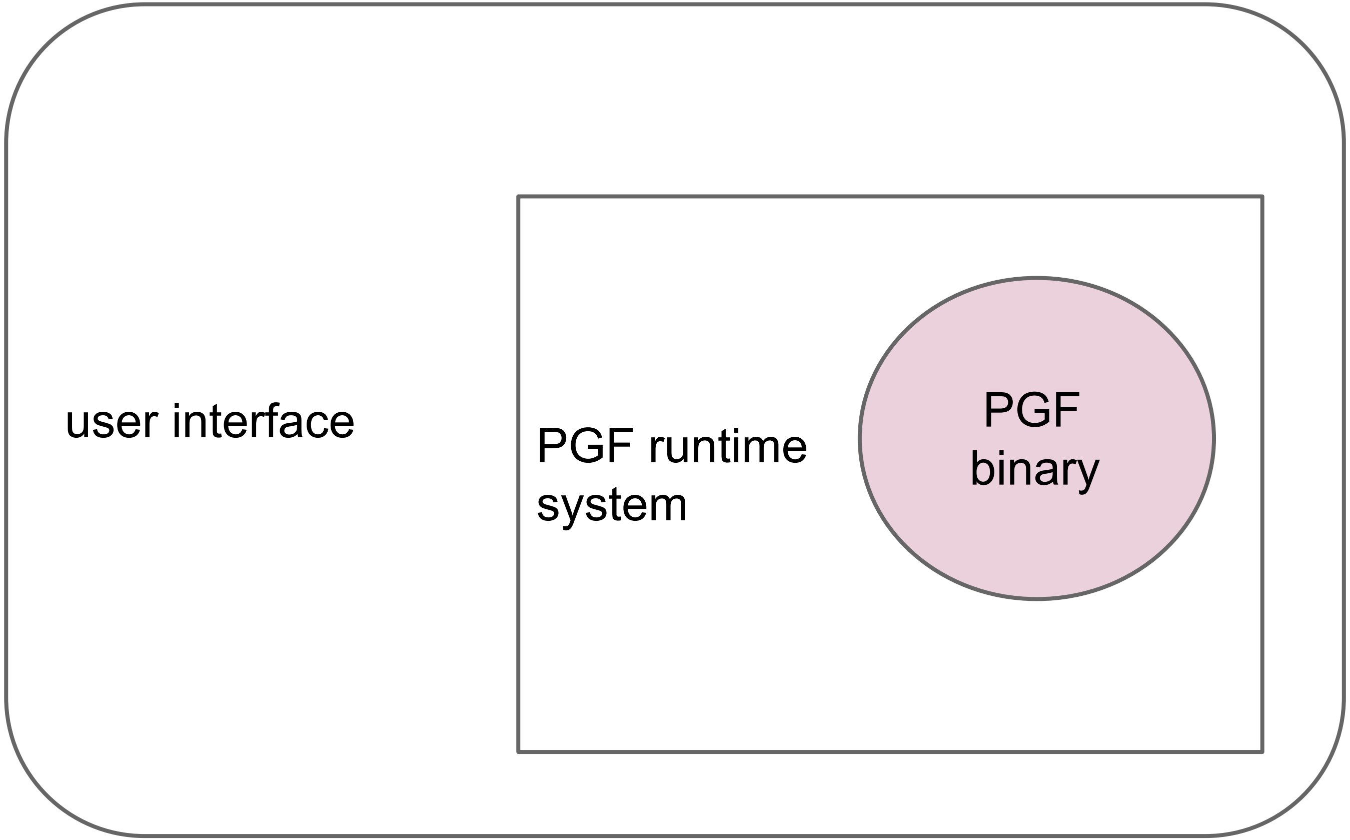

probability model
GF source


probability model
GF source
GF compiler
PGF binary



PGF runtime system

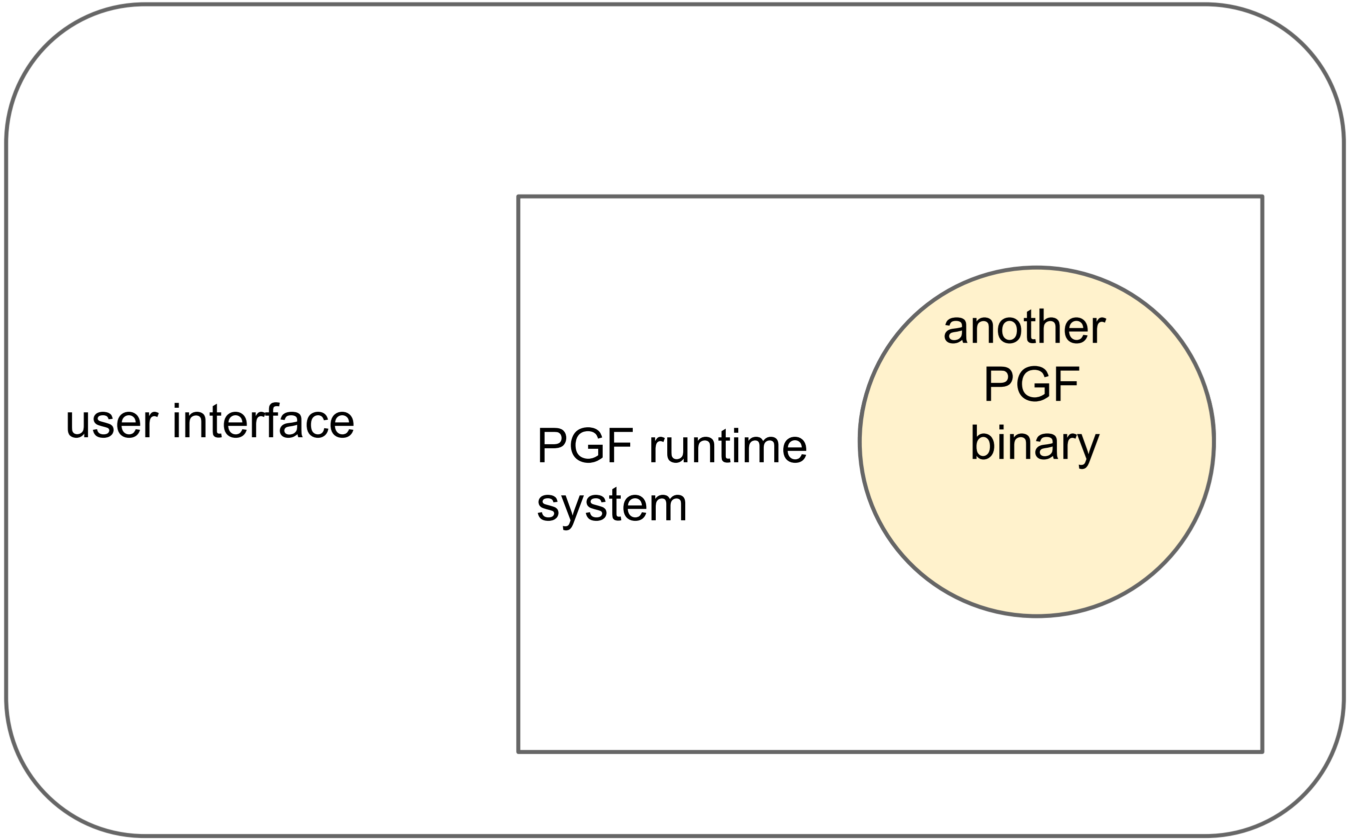
PGF binary



user interface

PGF runtime system

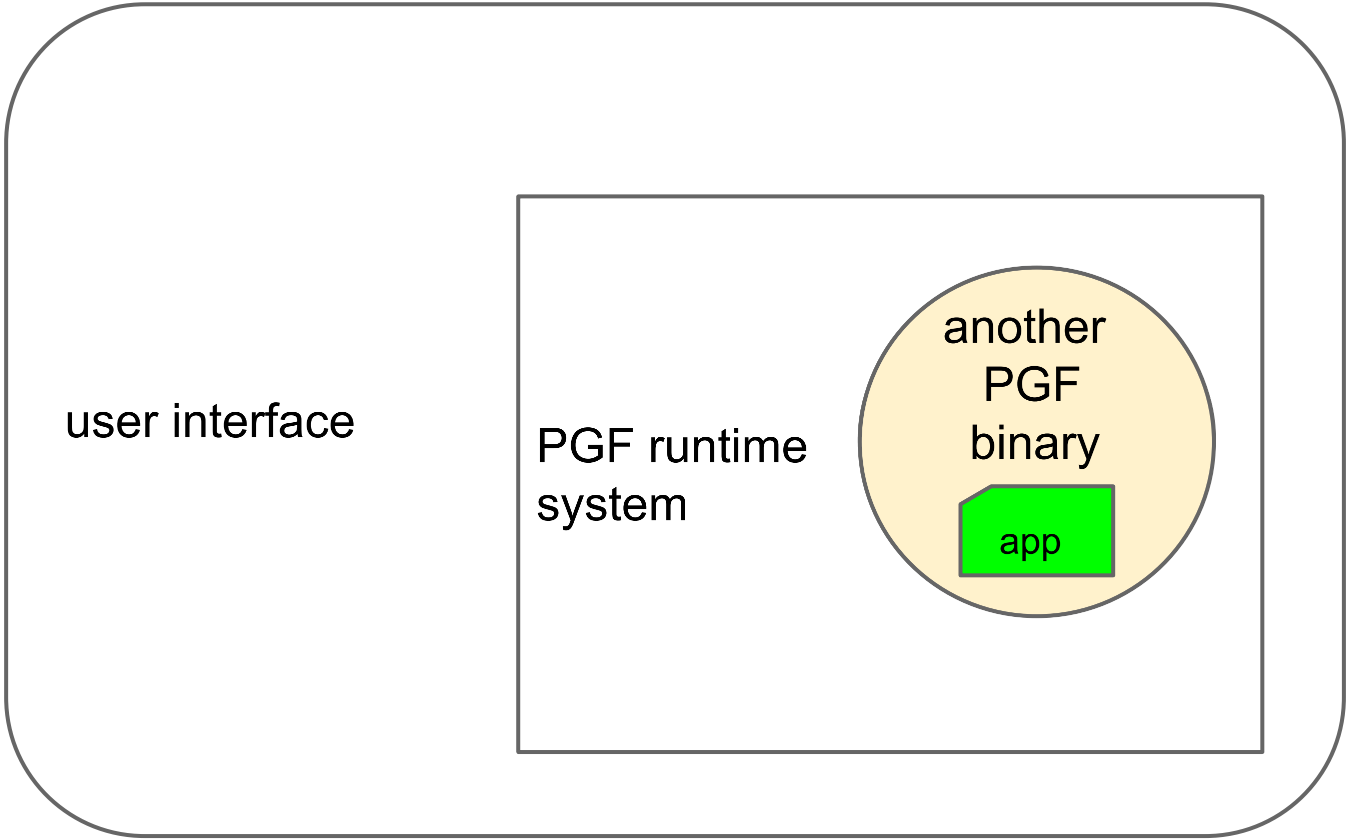
PGF binary



user interface

PGF runtime system

another PGF binary

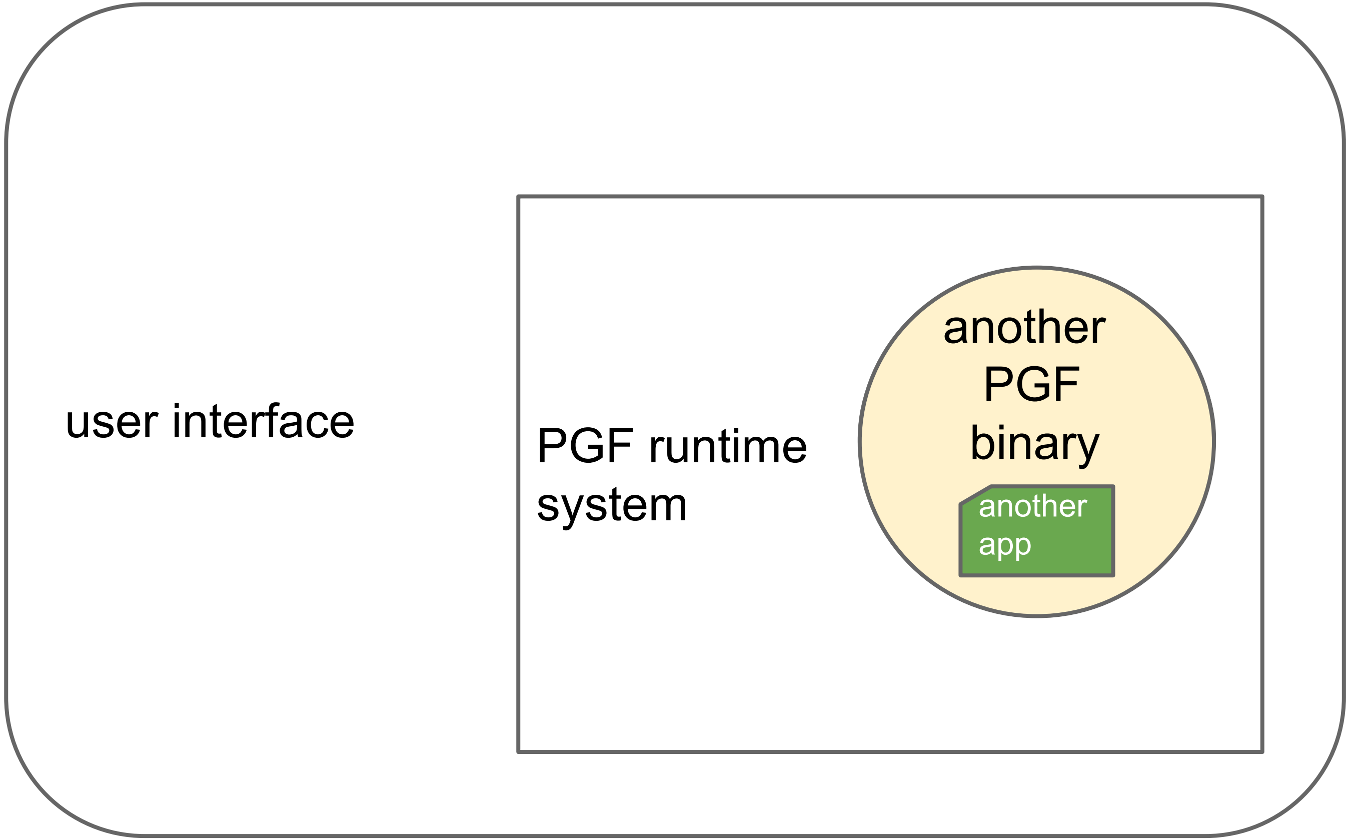


user interface

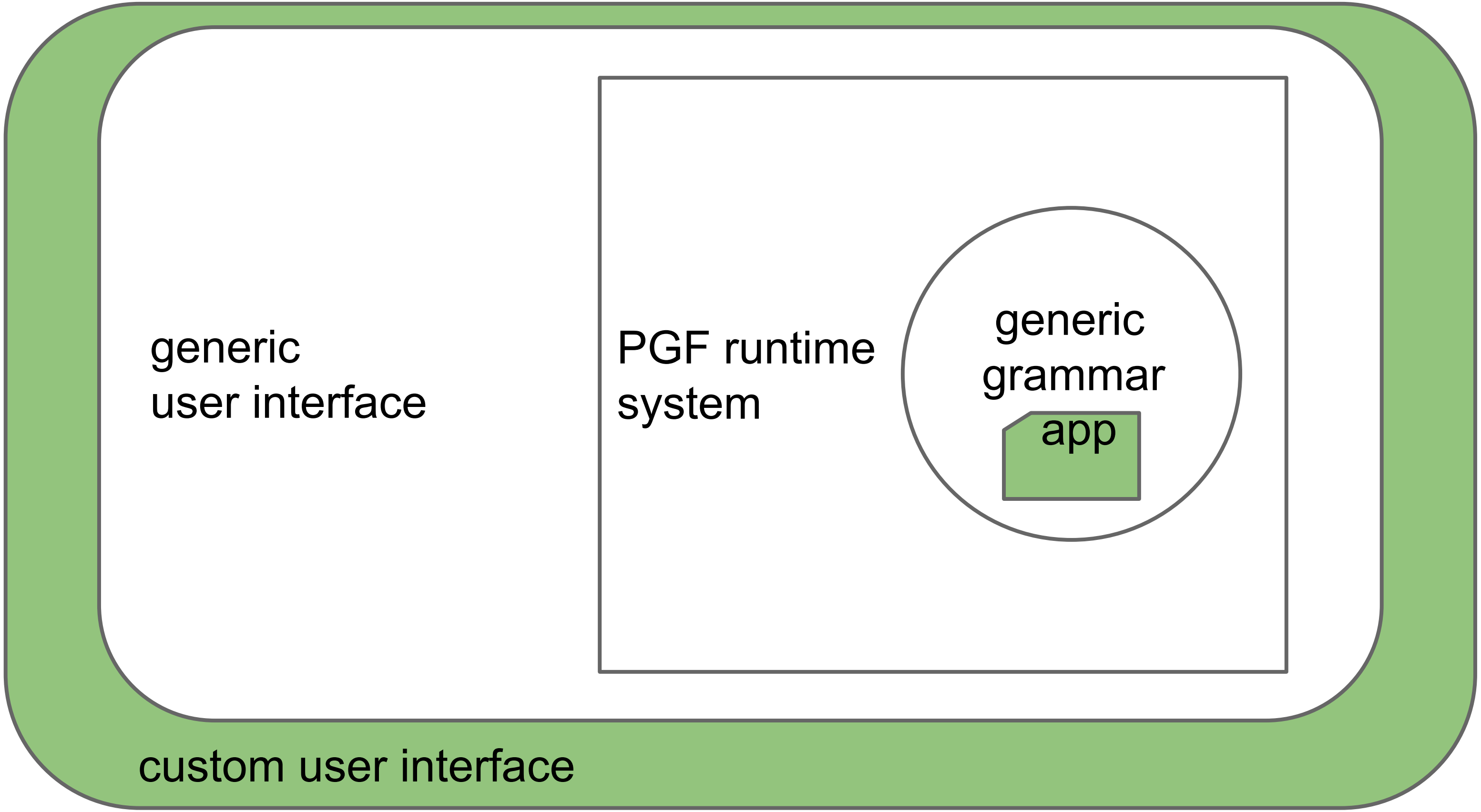
PGF runtime system

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

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

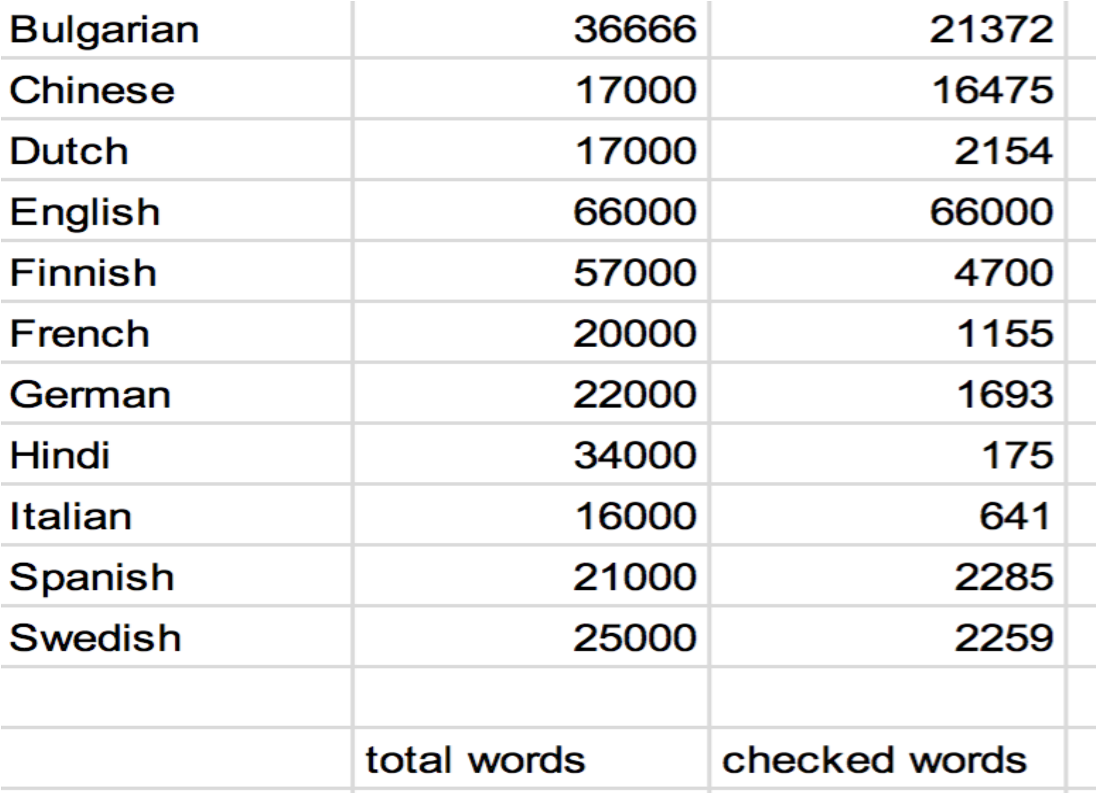
Split senses

Improve disambiguation

Introduce constructions

Design and perform evaluation

Current dictionary coverage



Bulgarian	36666	21372
Chinese	17000	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