Grammatical Framework

Formalizing the Grammars of the World

Aarne Ranta

University of Gothenburg

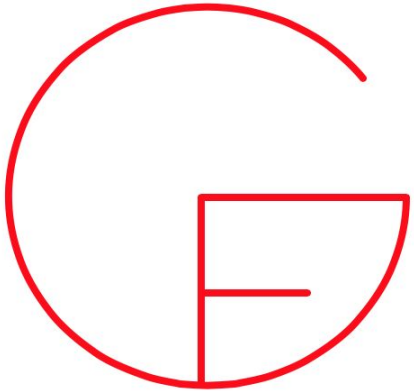
and

Digital Grammars AB

Google Zurich 7 September 2016

Part 1:

Mission



The mission of GF is to formalize the grammars of the world and make them available for computer applications.

http://www.grammaticalframework.org

Since 1998

Grammars…

PhD in linguistics

Grammars…

PhD in linguistics

5 years of work

Grammars…

PhD in linguistics

5 years of work

brittle

Grammars…

PhD in linguistics

5 years of work

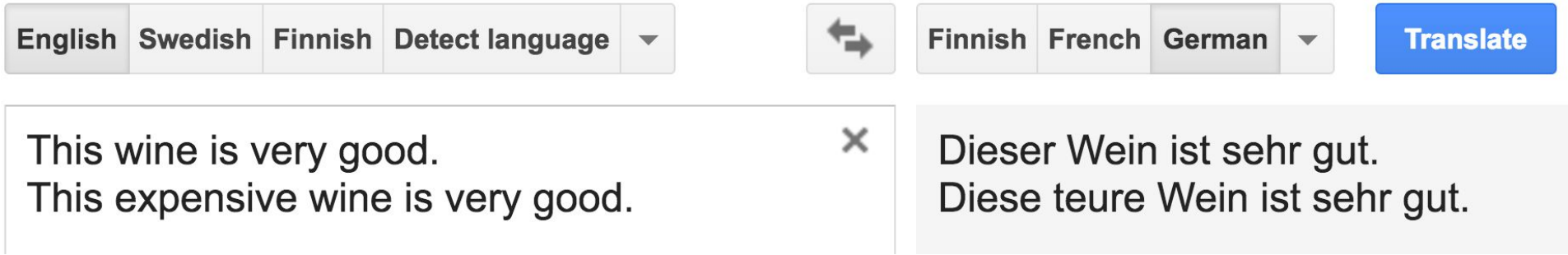
brittle

obsolete

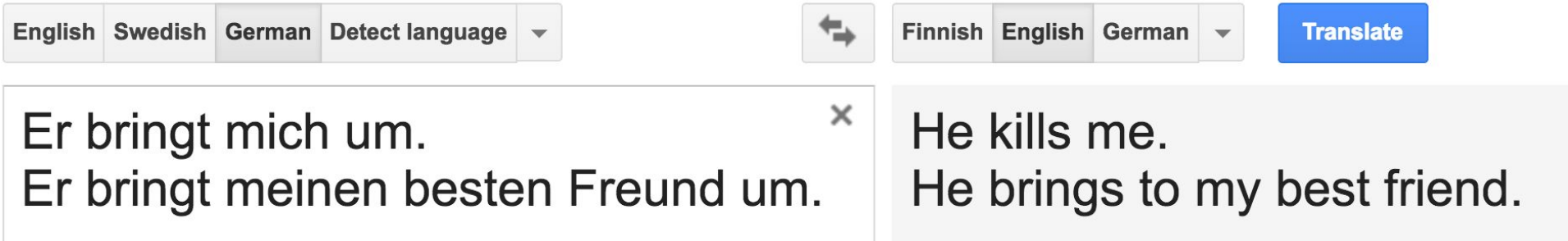
Grammars…	statistics
PhD in linguistics	BSc in computer science
5 years of work	5 weeks of waiting
brittle	robust
obsolete	state of the art

So, why grammars?

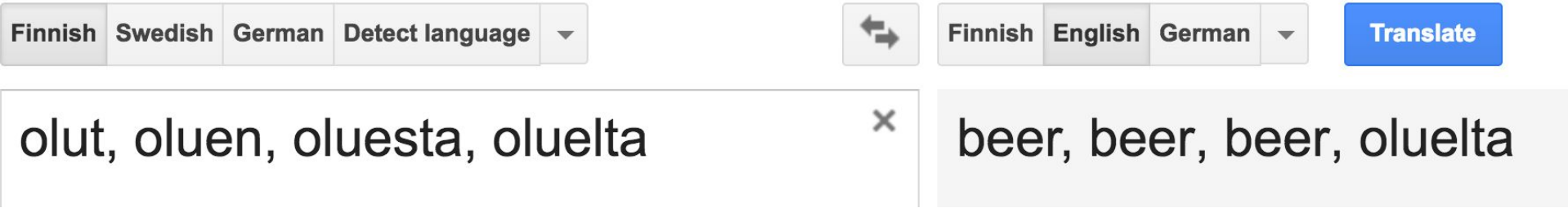
Reason 1: grammaticality



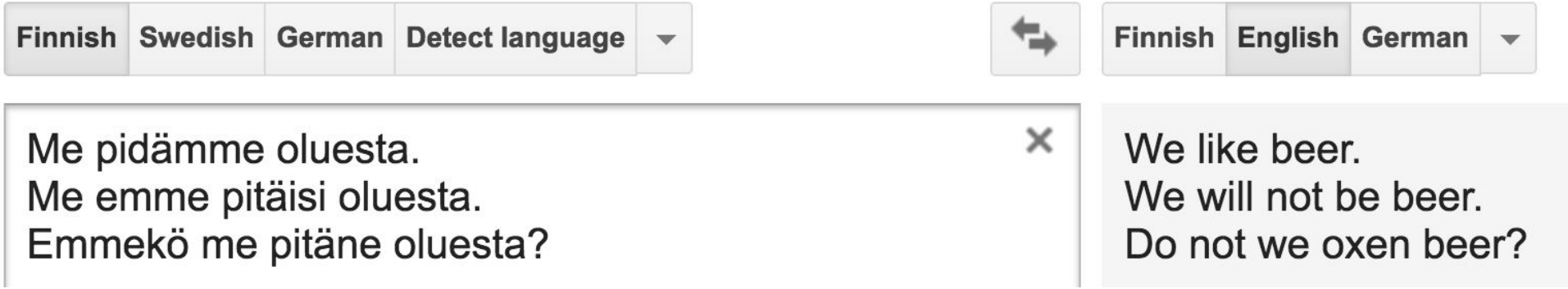
Reason 2: meaning



Reason 3: coverage (morphological)

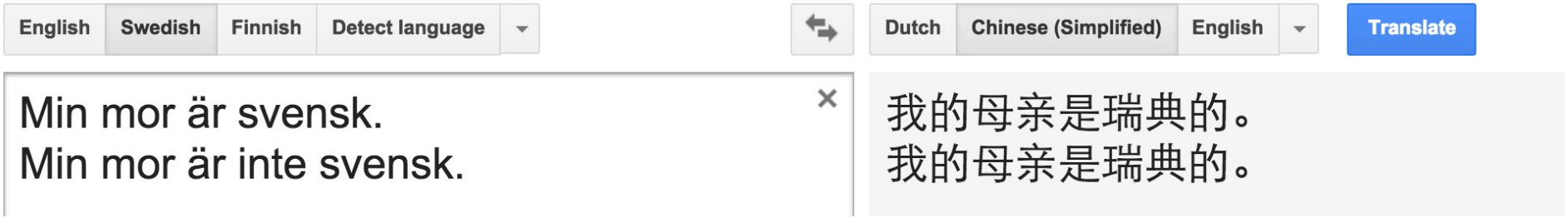


Reason 3: coverage (syntactic)

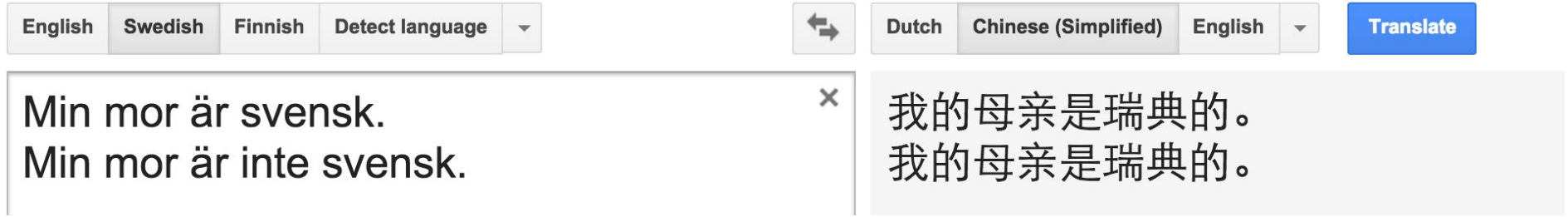


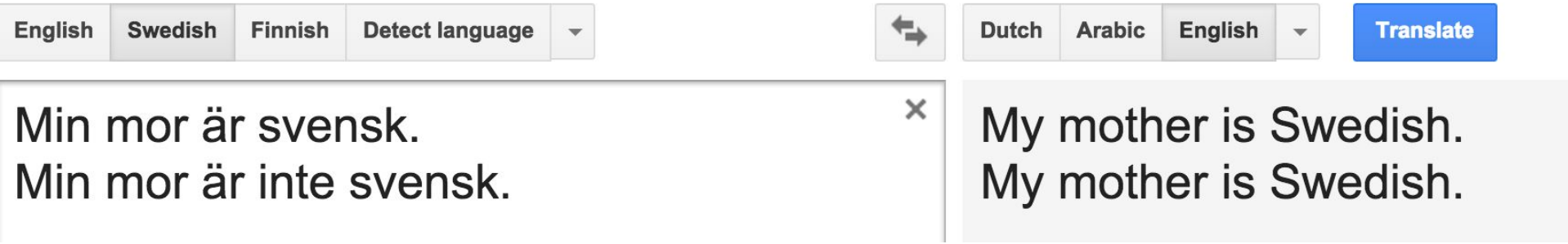
We like beer.
We would not like beer.
Wouldn’t we maybe like beer?

Reason 4: precision

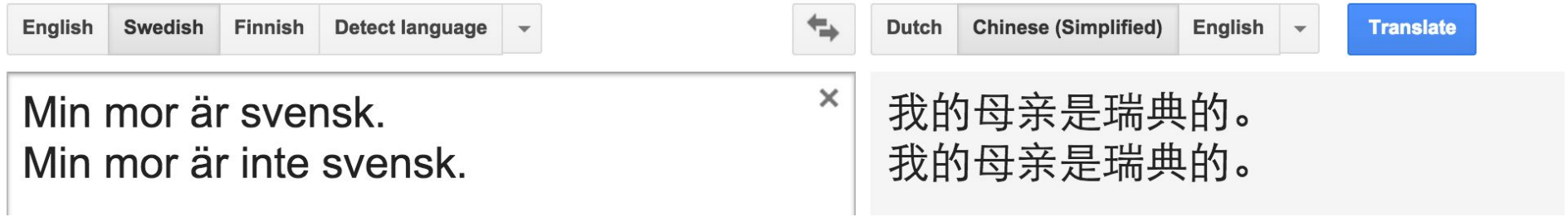


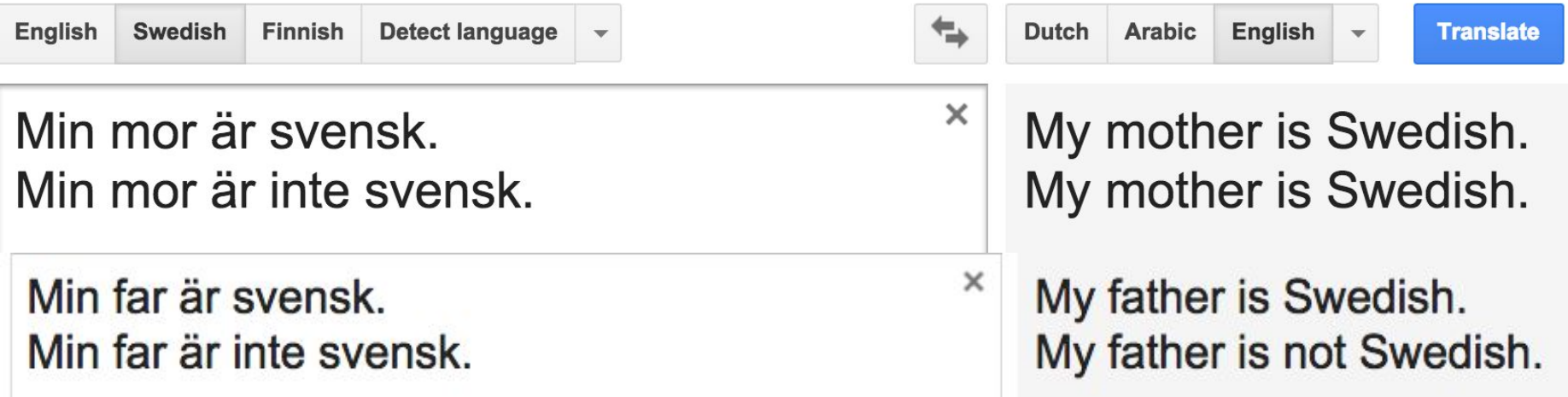
Reason 4: precision





Reason 4: precision



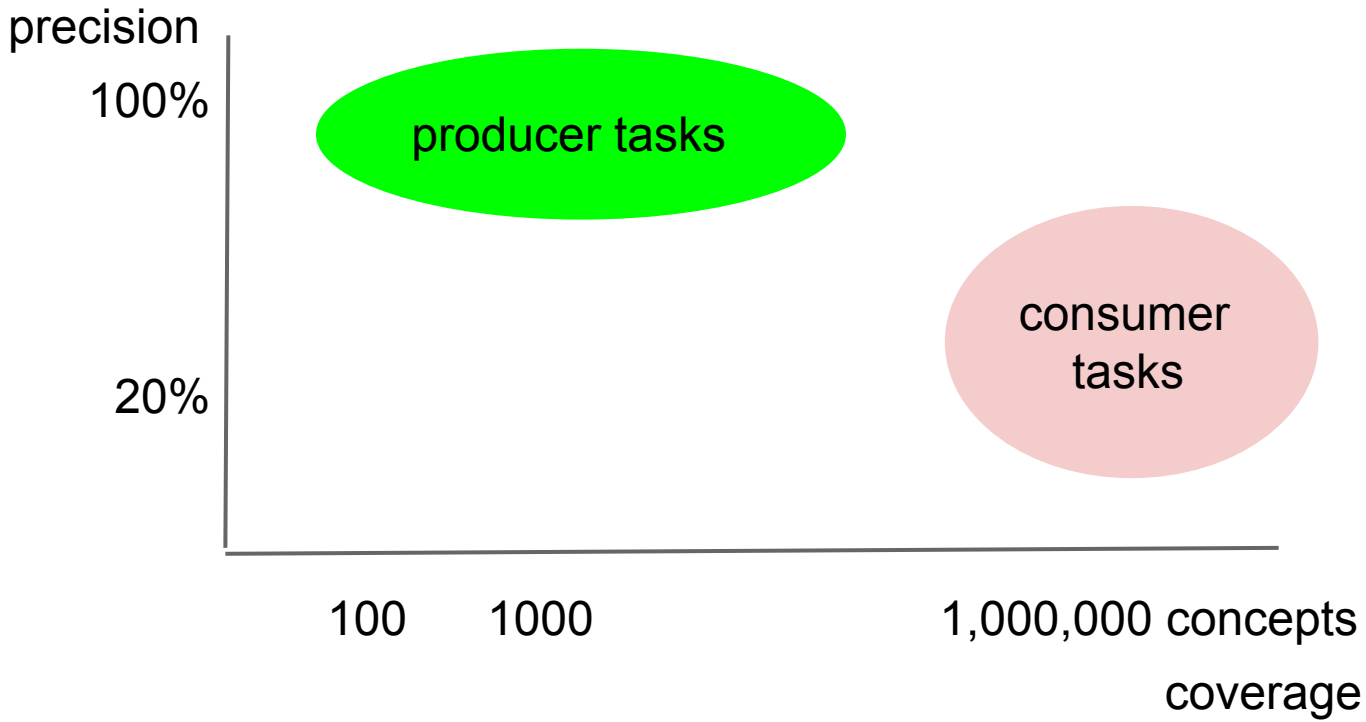


Reason 5: programmability

Grammars are **programs**, which can be

- inspected
- understood
- debugged
- recompiled (in a few minutes)

Reason 6: domain adaptability



TD

digital Grammars

Language technology to rely on.

- *Svängrumsytan utanför dörren lutar 1% i sidled.*
- *The turning space outside the gate tilts 1% sideways.*
- *Kääntymätila oven ulkopuolella kallistuu 1% sivusuunnassa.*
- *Der Schwenkbereich außerhalb der Tür neigt sich um 1% seitlich.*
- *La zona de giro fuera de la puerta se inclina 1% de lado.*

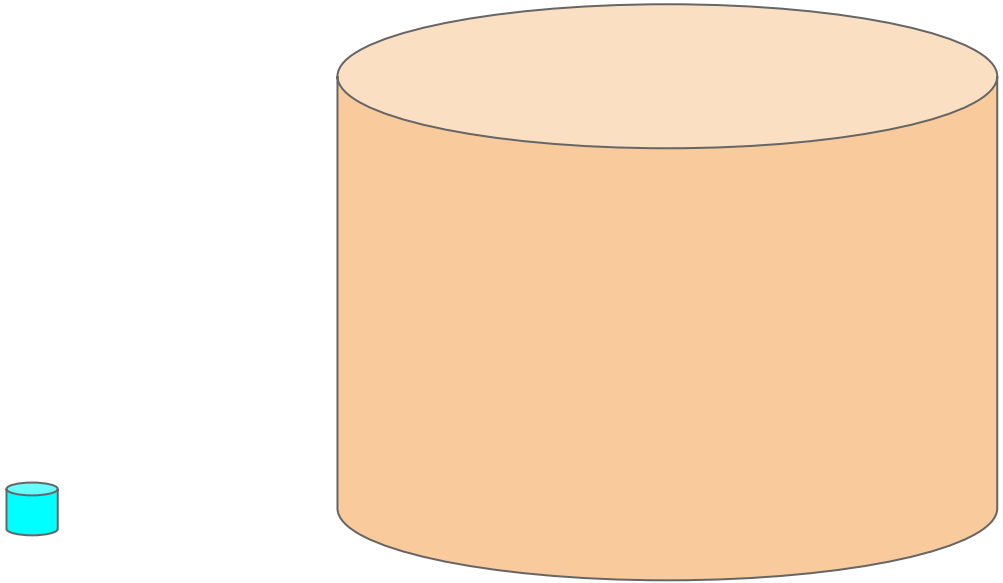
http://www.t-d.se/sv/TD2/

A. Ranta, C. Unger and D. Vidal Hussey, Grammar Enginering for a Customer: A Case Study with Five Languages, ACL-GEAF 2015.

Reason 7: compactness

GF Offline Translator: 16 modules, 35 MB in total

Google/Baidu translate offline: 210 modules, 30-150 MB each



GF Offline Translator

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

https://itunes.apple.com/us/app/gf-offline-translator/id1023328422?mt=8

K. Angelov, B. Bringert & A. Ranta, Speech-enabled hybrid multilingual translation for mobile devices, EACL 2014.

Making grammars work

Grammar formalisms

Tradition (1970-1980's)

Linguistic theory

Definition of grammaticality

Monolith

GF (1998-)

Grammar formalisms

Tradition (1970-1980's)

Linguistic theory

Definition of grammaticality

Monolith

GF (1998-)

Compiler construction

Transduction

Framework

Grammar formalisms

Tradition (1970-1980's)

Linguistic theory

Definition of grammaticality

Monolith

Copy and paste, macros

Programmer's discipline

Experts

GF (1998-)

Compiler construction

Transduction

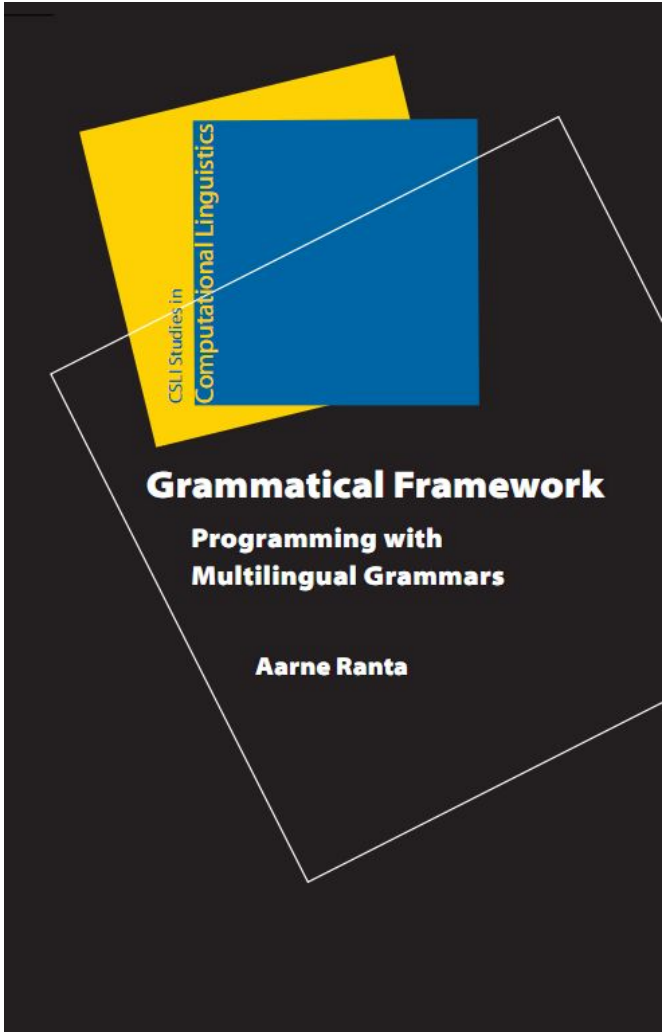
Framework

Grammar formalisms

Tradition (1970-1980's)	**GF (1998-)**
Linguistic theory	Compiler construction
Definition of grammaticality	Transduction
Monolith	Framework
Copy and paste, macros	Functional programming
Programmer's discipline	Static type checking
Experts	Distributed work

Grammar formalisms

Tradition (1970-1980's)	GF (1998-)
Linguistic theory	Compiler construction
Definition of grammaticality	Transduction
Monolith	Framework
Copy and paste, macros	Functional programming
Programmer's discipline	Static type checking
Experts	Distributed work
Monolingual	Multilingual



A. Ranta. *Grammatical Framework: Programming with Multilingual Grammars*, CSLI, Stanford, 2011. Chinese translation by Prof. Yan Tian: 语法框架 为多种自然语言语法编程, Shanghai Jiao Tong University Press, 2014.

Grammar building effort

First grammar on cloud IDE minutes

Grammar building effort

First grammar on cloud IDE	minutes
“Foods” application grammar	hours
Commercial application grammar	days

Grammar building effort

First grammar on cloud IDE	minutes
“Foods” application grammar	hours
Commercial application grammar	days
Mini resource grammar	weeks
Full resource grammar	months

Grammar building effort

First grammar on cloud IDE	minutes
“Foods” application grammar	hours
Commercial application grammar	days
Mini resource grammar	weeks
Full resource grammar	months
Wide coverage grammar	

Grammar building effort

First grammar on cloud IDE	minutes
“Foods” application grammar	hours
Commercial application grammar	days
Mini resource grammar	weeks
Full resource grammar	months
Wide coverage grammar	days

Results

Global community of 200+ members

Nieminen, Camilleri 2013-

Wide coverage:

English Swedish German Dutch French Italian Spanish Catalan Bulgarian Russian Finnish Estonian Japanese Thai Chinese Hindi

Complete Resource Grammar Library:

Norwegian Danish Afrikaans Romanian Polish Russian Latvian Mongolian Urdu Punjabi Sindhi Greek Maltese Nepali Persian

In progress:

Latin Slovene Turkish Basque Icelandic Hebrew Arabic Amharic Swahili ...

Rewarding work attracts developers

Undergraduates, PhD students, professors, translators, hackers…

Computer scientists, mathematicians, linguists...

“Once I got into it it was obviously a lot of fun…”

GF Summer Schools

Gothenburg, Sweden, 2009

Barcelona, Catalonia, 2011

Frauenchiemsee, Bavaria, 2013

Marsalforn, Gozo, 2015

Forthcoming summer schools

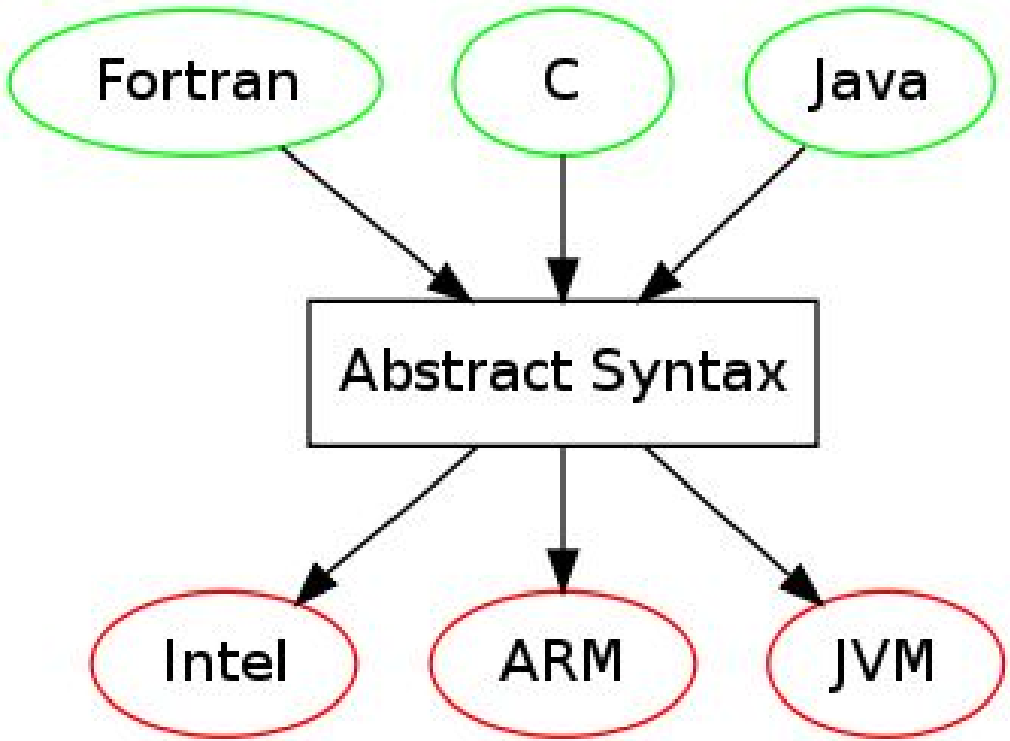
Riga, Latvia: August 2017

Stellenbosch, South Africa: January 2018

Part 2:

How it works

Model: multi-source multi-target compiler

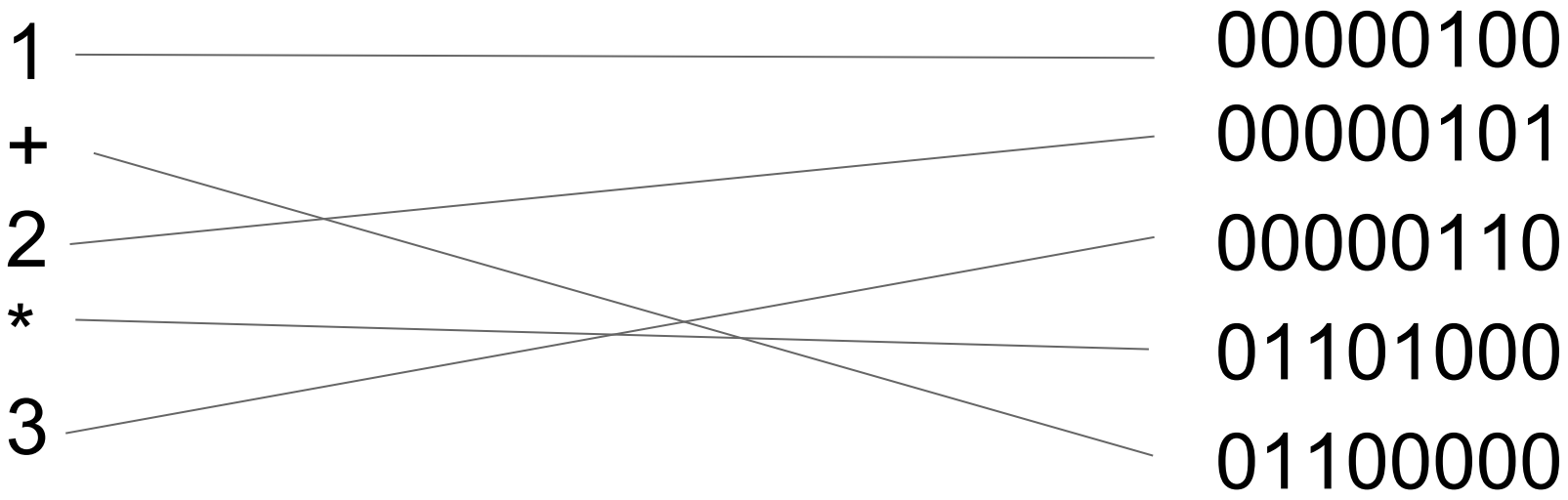


Compiling Java to JVM

1 + 2 * 3

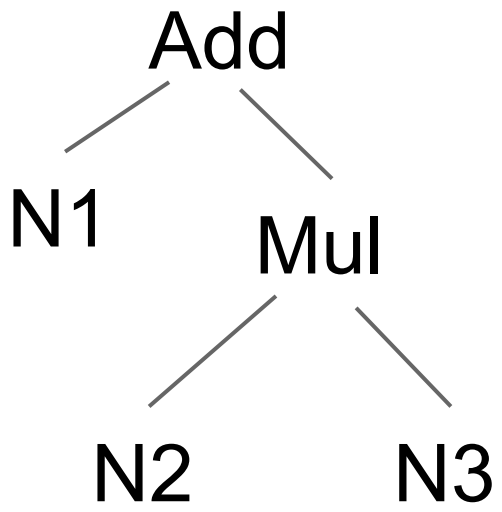
00000100
00000101
00000110
01101000
01100000

Word alignments



Abstract syntax

1 + 2 * 3



00000100
00000101
00000110
01101000
01100000

How it is defined

Abstract syntax

```
fun Add : Exp -> Exp -> Exp
```

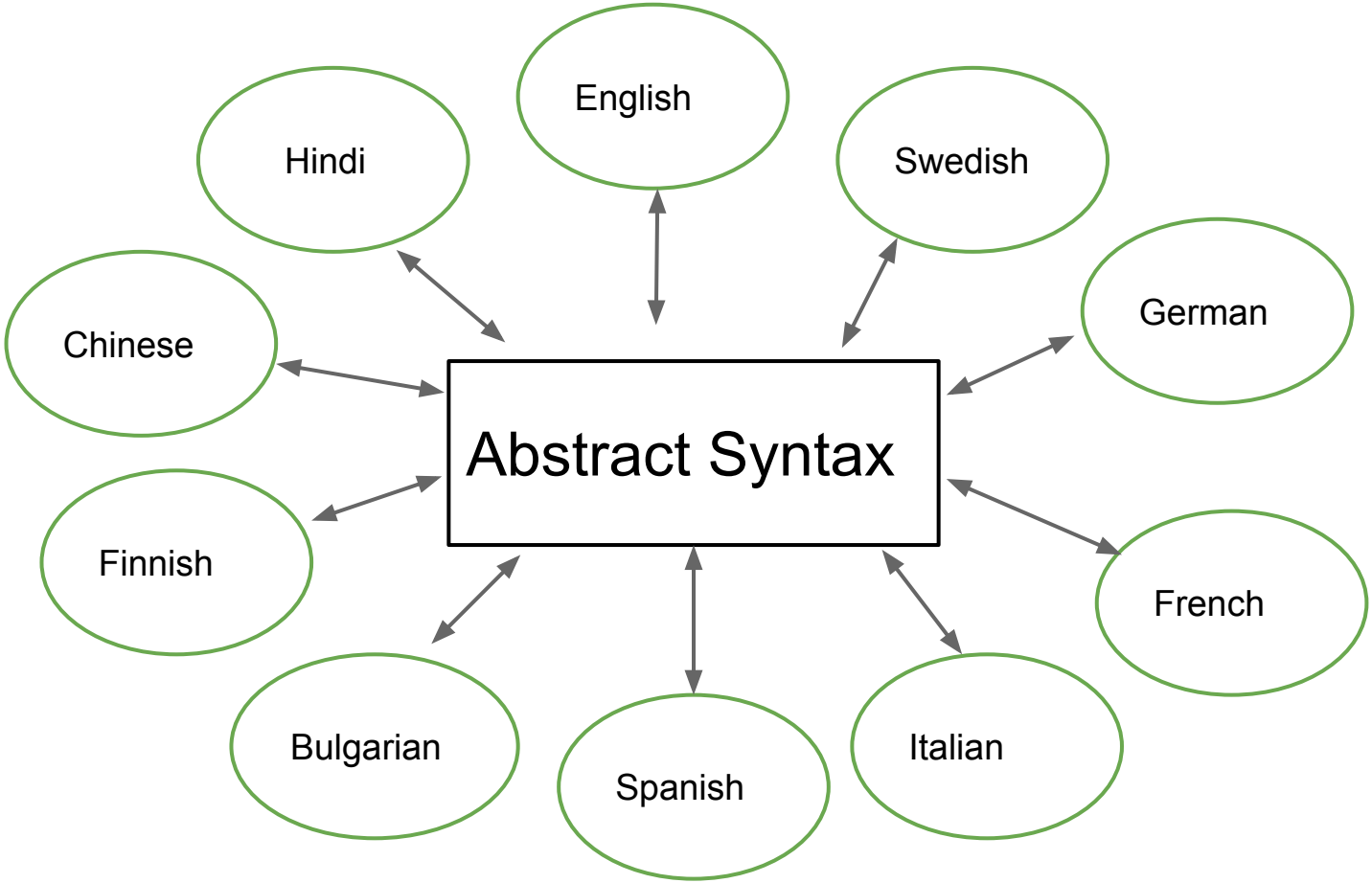
Concrete syntax, Java

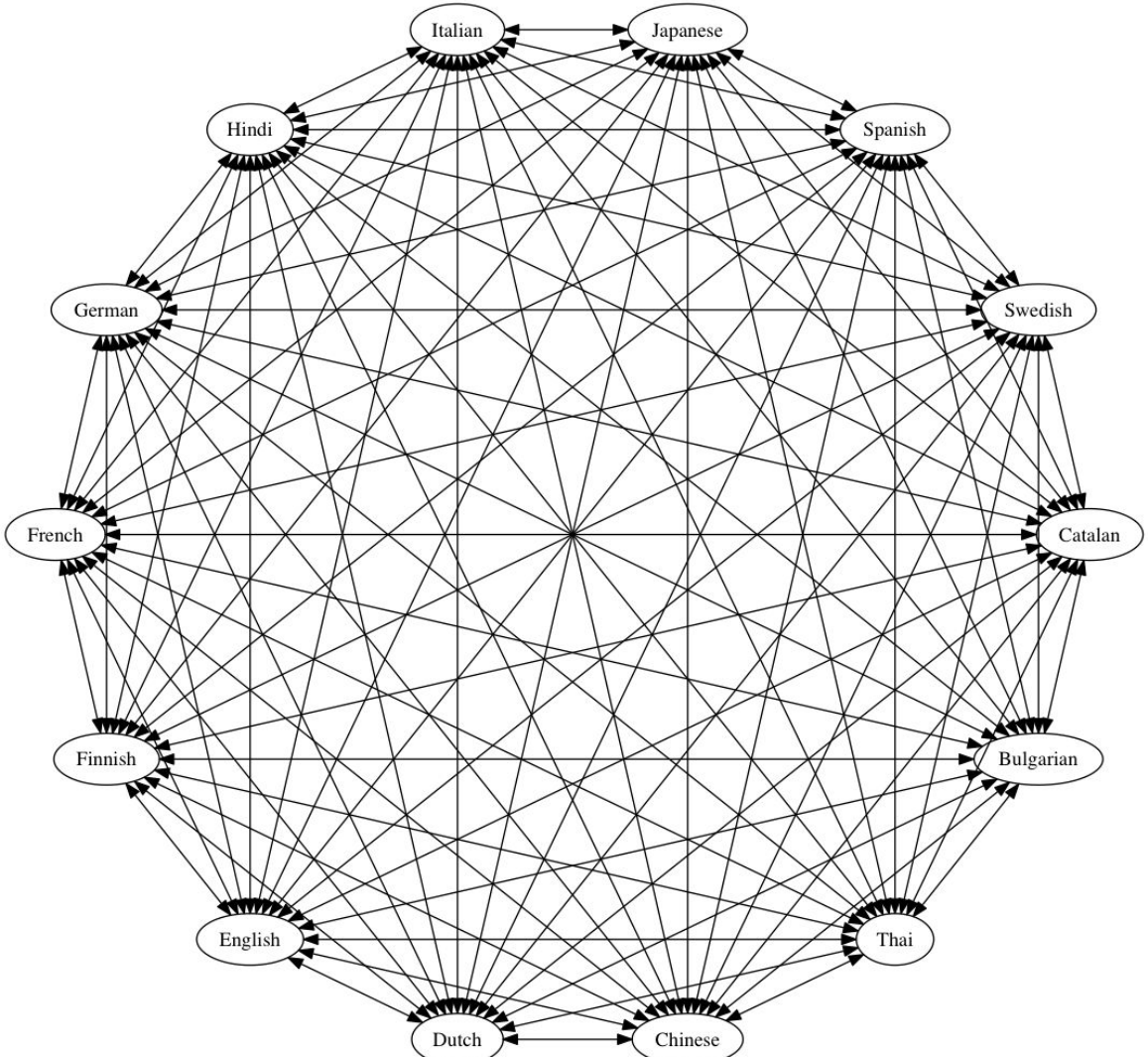
```
lin Add x y = x ++ “+” ++ y
```

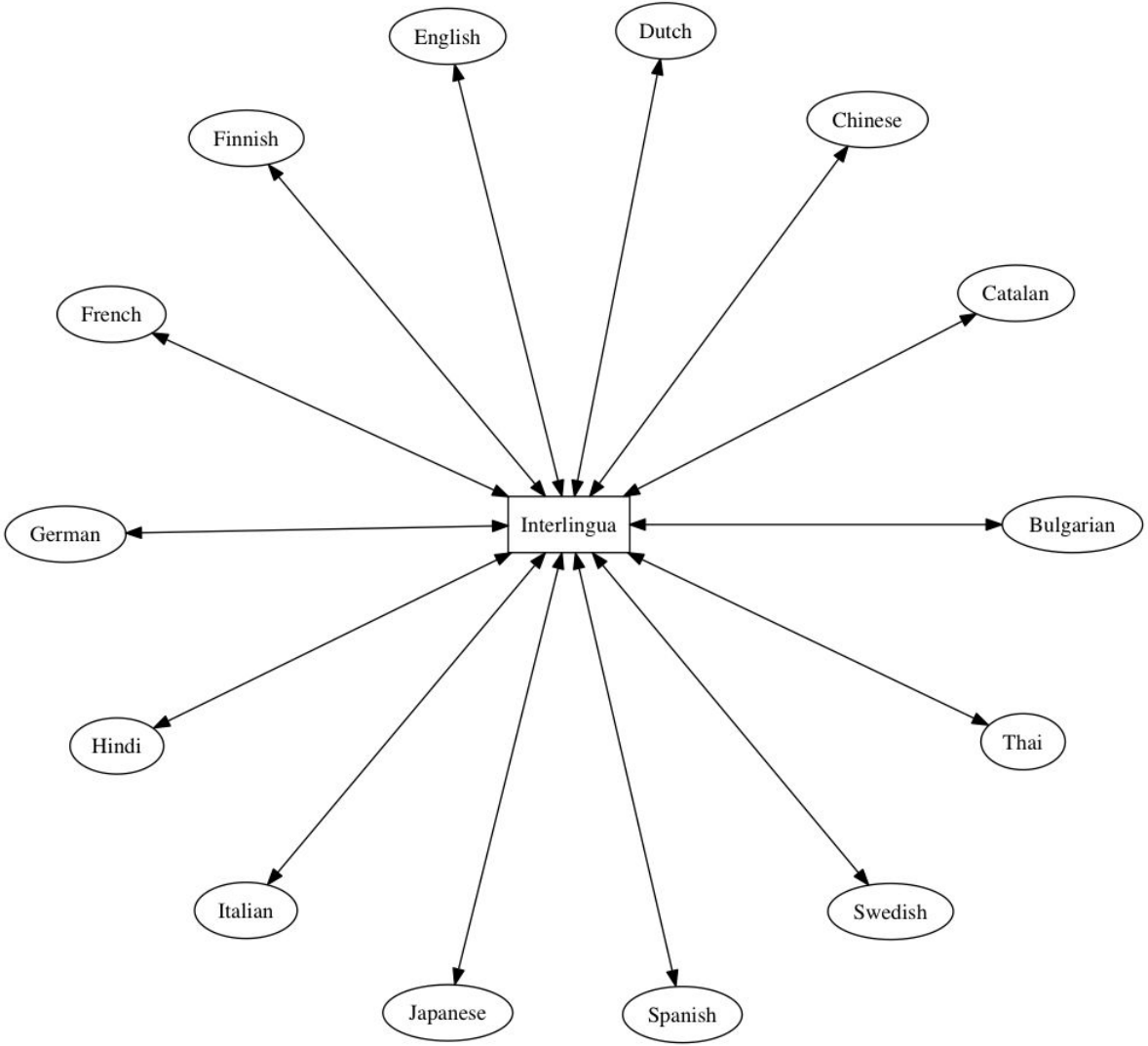
Concrete syntax, JVM

```
lin Add x y = x ++ y ++ “01100000”
```

“Compiling natural language”



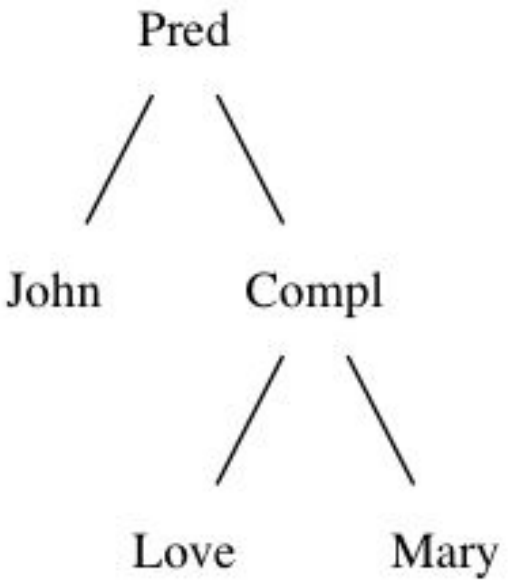

Italian
Japanese
Hindi
Spanish
German
Swedish
French
Catalan
Finnish
Bulgarian
English
Thai
Dutch
Chinese

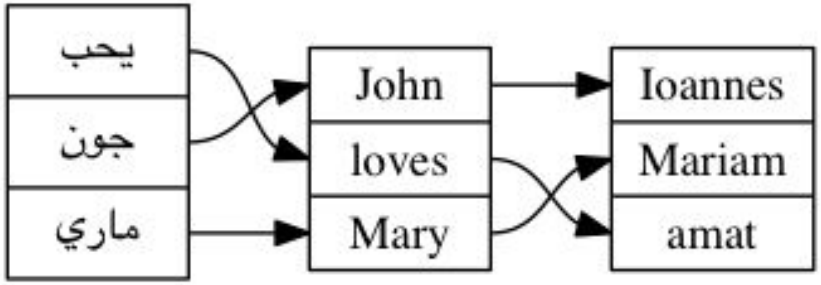

English
Dutch
Finnish
Chinese
French
Catalan
German
Interlingua
Bulgarian
Hindi
Thai
Italian
Swedish
Japanese
Spanish

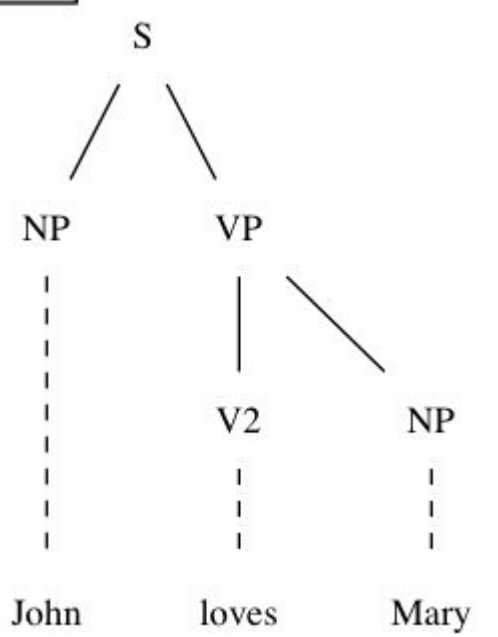
Subject-verb-object predication

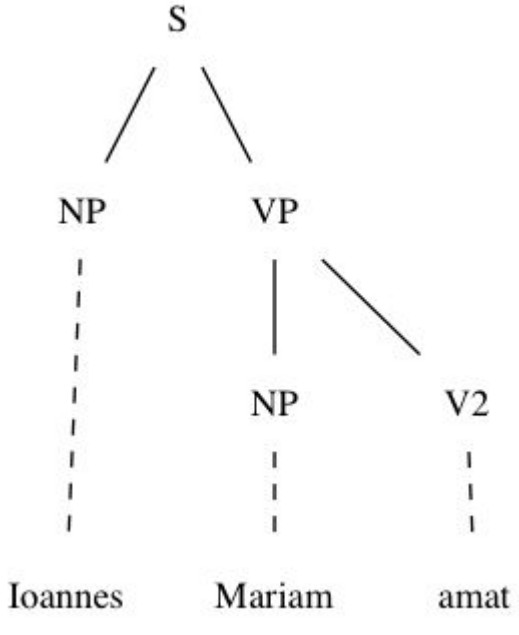
abstract syntax

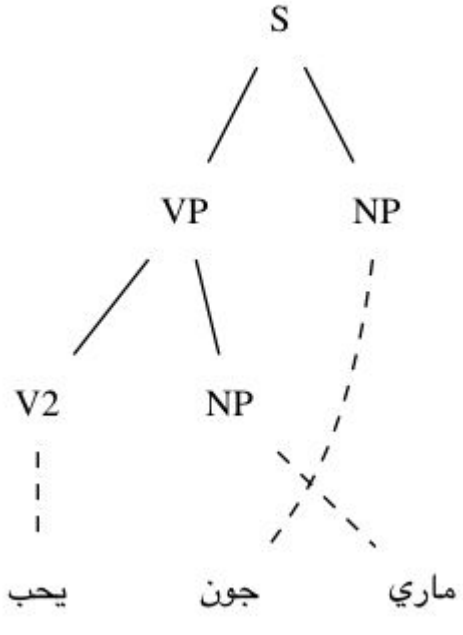
```
fun Pred  : NP -> VP -> S
fun Compl : V2 -> NP -> VP
```











English: SVO

Latin: SOV

Arabic: VSO

Subject-verb-object predication

abstract syntax

```
fun Pred  : NP -> VP -> S
fun Compl : V2 -> NP -> VP
```

concrete syntax

```
SVO: lin Pred  s  vo = s ++ vo
     lin Compl v  o  = v ++ o

SOV: lin Pred  s  vo = s ++ vo
     lin Compl v  o  = o ++ v
```

What about VSO?

Linearization can produce **records** in addition to strings.

```
lin Pred  s vo = vo.verb ++ s ++ vo.obj

lin Compl v o  = {verb = v ; obj = o}
```

Inflection tables

```
lin Love = table {
  Inf => “love” ;
  Sg3p => “loves” ;
  Past | PastPart => “loved” ;
  PresPart => “loving” ;
  }
```

Inflection tables

```
lin Love = table {
  Ind Pres Sg P1 => “liebe” ;
  Ind Pres Sg P2 => “liebst” ;
  Ind Pres Sg P3 => “liebt” ;
  … -- up to 86 forms
  }
```

Inflection tables

```
lin Love = table {
  Ind Pres Sg P1 => “liebe” ;
  Ind Pres Sg P2 => “liebst” ;
  Ind Pres Sg P3 => “liebt” ;
  … -- up to 86 forms
  }

lin Love = “爱” ;
```

Agreement

```
Pred I   (Compl Love) She          我 爱 她
Pred She (Compl Love) I            她 爱 我
```

Agreement

`Pred I (Compl Love) She`	我 爱 她
`Pred She (Compl Love) I`	她 爱 我
`Pred I (Compl Love) She`	*I love her*
`Pred She (Compl Love) I`	*she loves me*

Agreement defined

```
lin She = {
     s = table {Nom => “she” ; Acc => “her”} ;
     a = Sg3p
     }

lin Pred  s vo = s.s ! Nom ++ vo ! s.a

lin Compl v o  = table {a => v ! a ++ o.s ! Acc}
```

Agreement + German word order

```
lin Pred np vp = table {
  Main => subj ++ verb ++ obj ;
  Inv  => verb ++ subj ++ obj ;
  Sub  => subj ++ obj  ++ verb
  } where {
     subj = np.s ! Nom ;
     verb = vp.verb ! np.a ;
     obj  = vp.obj
     }
```

Multilingual abstraction

Abstract syntax:

- tree structure: constituency, word senses

Concrete syntax:

- words, word order
- inflection, agreement

Expressivity of GF

GF source language

- type theory + functional programming
- static type checking
- module system

PGF machine language

- Portable Grammar Format (binary)
- PMCFG (Parallel Multiple Context-Free Grammar)

P. Ljunglöf, The Expressivity and Complexity of Grammatical Framework, PhD thesis, 2004.

K. Angelov and P. Ljunglöf, Fast statistical parsing with parallel multiple context-free grammars, EACL 2014.

Added expressive power

Synchronous context-free grammars

- operating on strings

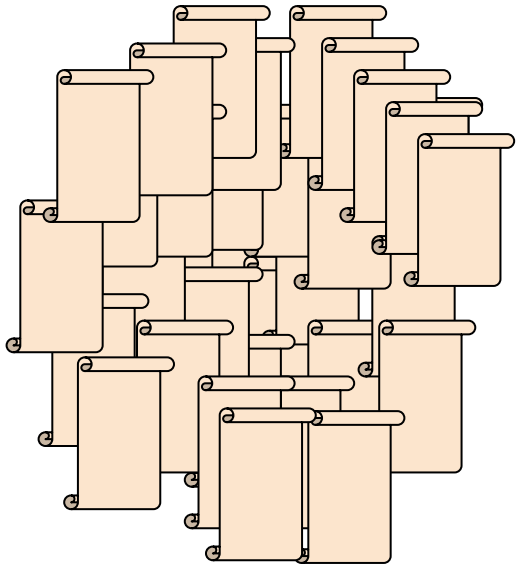
$\longrightarrow$

Synchronous multiple context-free grammars

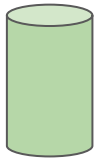
- operating on tuples
  - inflection tables
  - records

PGF = Portable Grammar Format

PGF = Type Theory + PMCFG



GF compiler



PGF interpreter

K.Angelov, B.Bringert, A. Ranta, PGF: A Portable Run-Time Format for Type-Theoretical Grammars, JoLLI 2010.

The Resource Grammar Library

Morphology: Smart Paradigms

```
mkV : Str -> V
```

```
mkV “talk” → talk,talks,talked,talked,talking
mkV “cry”  → cry,cries,cried,cried,crying
mkV “dip”  → dip,dips,dipped,dipped,dipping
```

Both for manual and automatic lexicon building

Lexicon	Forms	Entries	Cost	$m = 1$	$m \leq 2$
Eng N	2	15,029	1.05	95%	100%
Eng V	5	5,692	1.21	84%	95%
Swe N	9	59,225	1.70	46%	92%
Swe V	20	4,789	1.13	97%	97%
Fre N	3	42,390	1.25	76%	99%
Fre V	51	6,851	1.27	92%	94%
Fin N	34	25,365	1.26	87%	97%
Fin V	102	10,355	1.09	96%	99%

G. Détrez and A. Ranta, Smart Paradigms and the Predictability and Complexity of Inflectional Morphology, EACL-2012.

Syntax: combination functions

```
mkCl : NP -> V2 -> NP -> Cl

mkCl he_NP umbringen_V2 i_NP →
```

er bringt mich um
er bringt mich nicht um
(daß) er mich umbringt
hätte er mich umgebracht
(daß) er mich nicht umgebracht haben wird
…

Syntax: combination functions

Function	Type	Example
`genericCl`	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 says that I sleep*
`mkCl`	NP -> VQ -> QS -> Cl	*she wonders who sleeps*
`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*

- API: `mkUtt (mkCl she_NP wonder_VQ (mkQS (mkQCl who_IP sleep_V)))`
- Afr: *sy vra hom af wie slaap*
- Bul: *тя се чудва кой спи*
- Cat: *ella es pregunta qui dorm*
- Chi: *她好奇谁睡觉*
- Dan: *hun undres hvem som sover*
- Dut: *ze vraagt zich af wie slaapt*
- Eng: *she wonders who sleeps*
- Est: *tema arutleb kes magab*
- Fin: *hän ihmettelee , kuka nukkuu*
- Fre: *elle se demande qui dort*
- Ger: *sie wundert sich , wer schläft*
- Gre: *αυτή αναρωτιέται ποιός κοιμάται*

Example: “social media” localization

```
fun Like : Person -> Item -> Fact

lin Like x y = mkCl x (mkV2 “like”) y
lin Like x y = mkCl x (mkV2 “gilla”) y
lin Like x y = mkCl y (mkV2 piacere_V a_Prep) x
```

This generates

John likes your picture	*La tua immagine piace a John*
I like your picture	*La tua immagine mi piace*
I like you	*Mi piaci*
You wouldn’t like this	*Questo non ti piacerebbe*
...	*...*

Demo:
resource grammar with
“fridge magnets”

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

AST = Abstract Syntax Tree

Language-independent

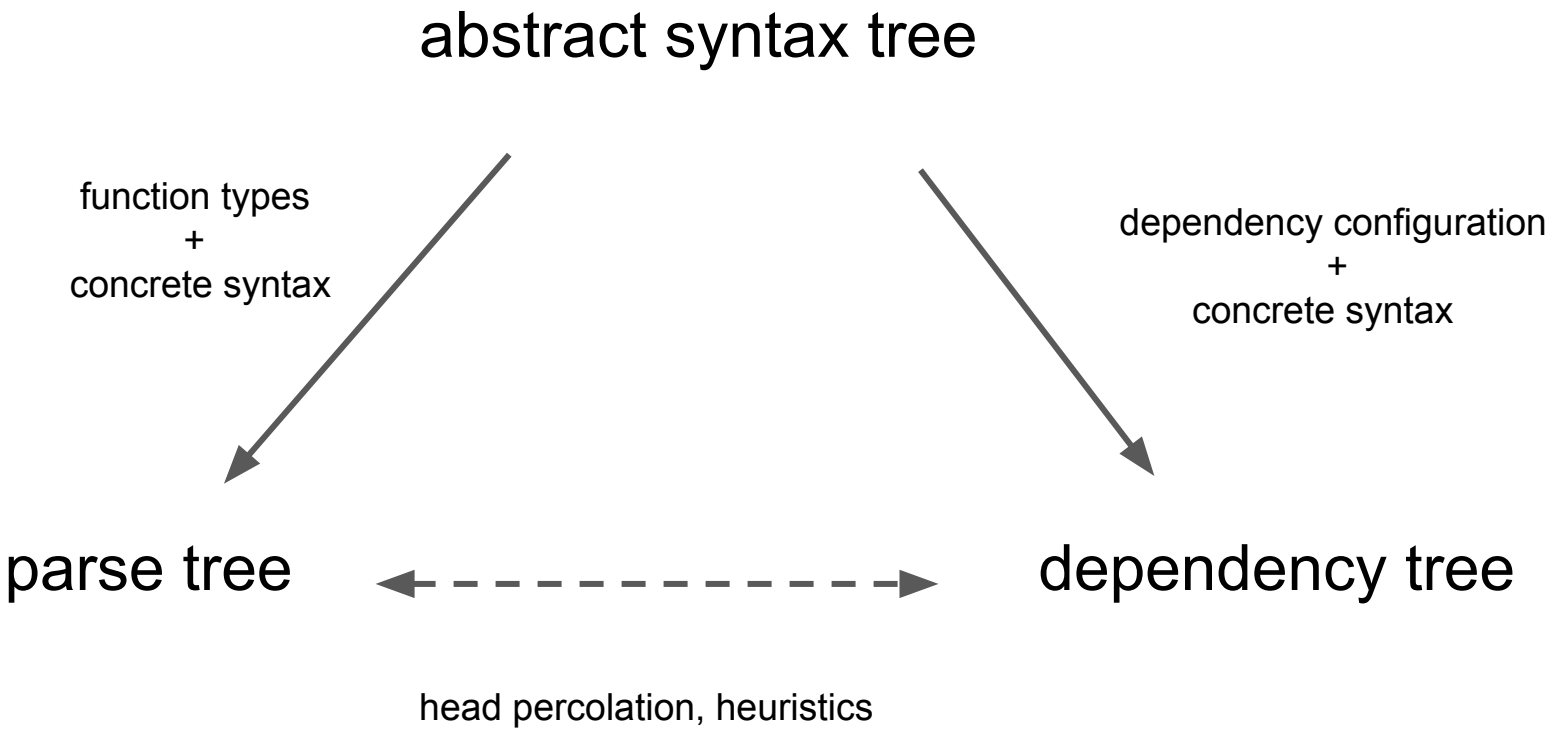
- ignores morphology and word order

Semantic

- type-theoretic lambda term

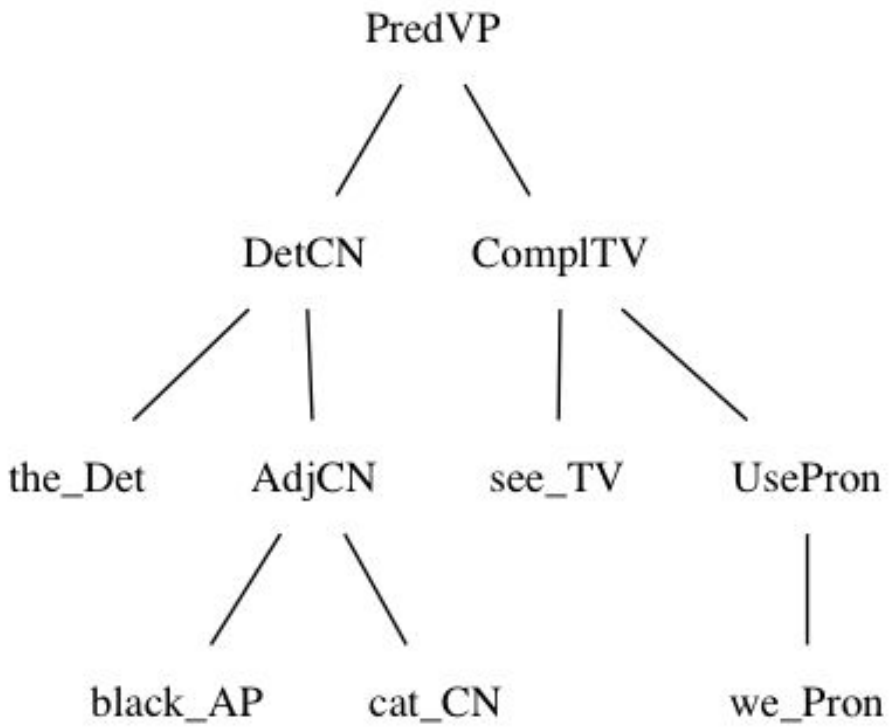
Non-lossy

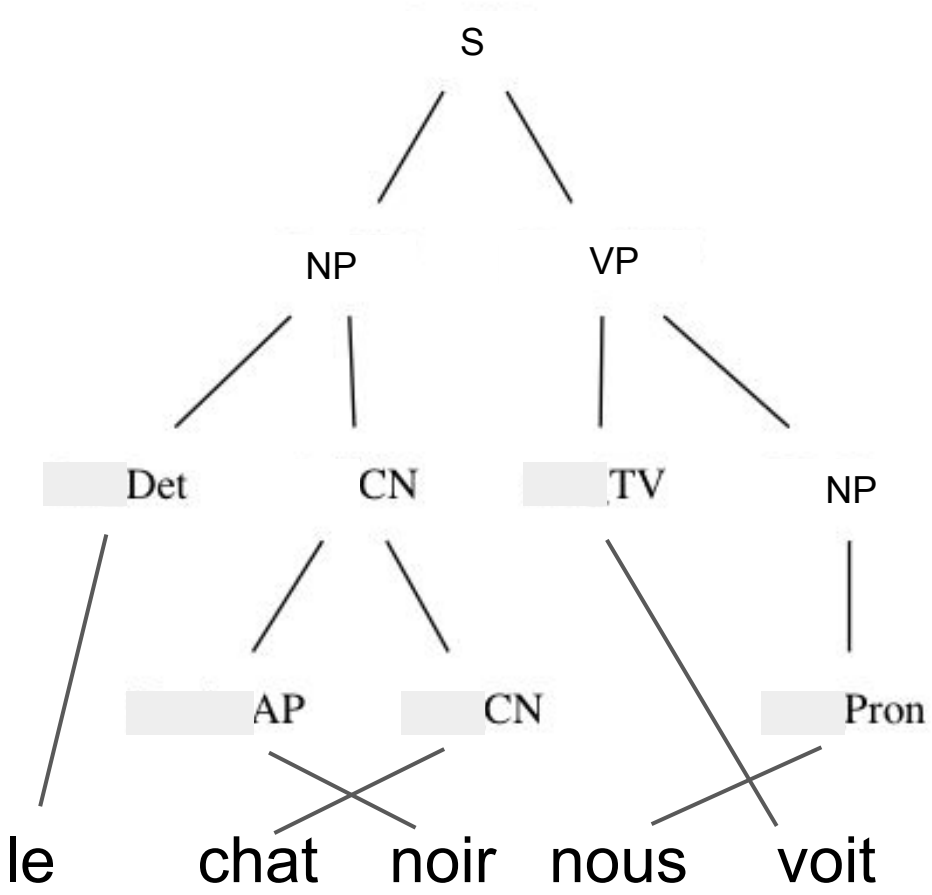
- can generate other trees and surface forms

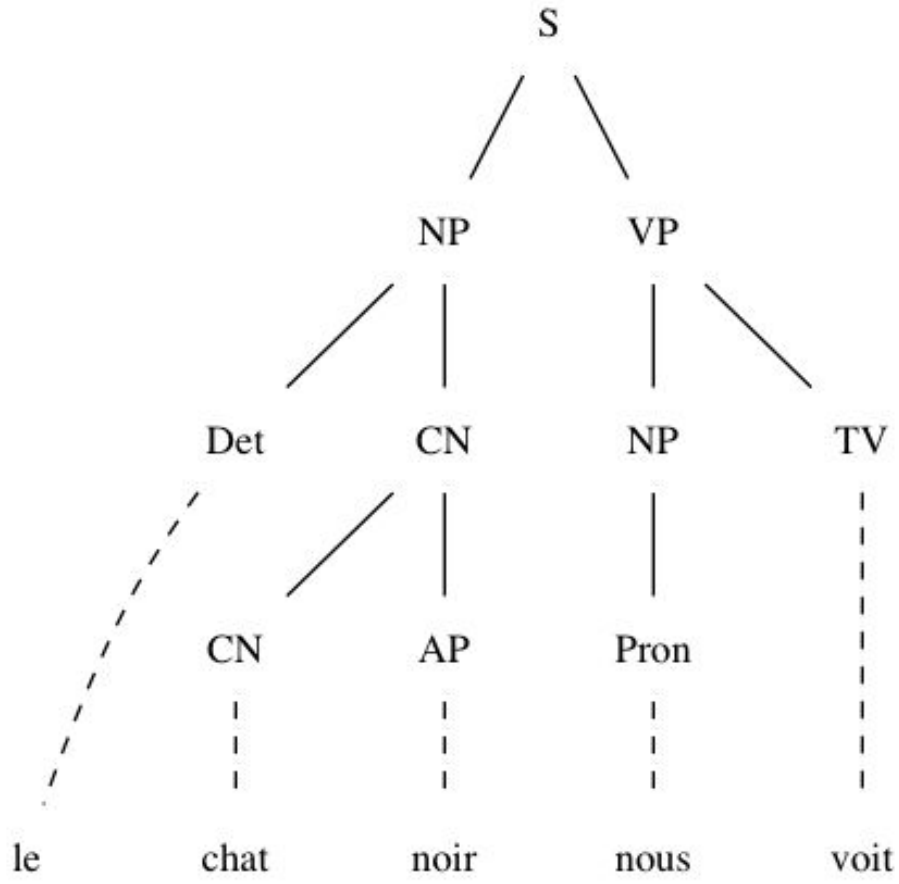

abstract syntax tree
function types
+
concrete syntax
dependency configuration
+
concrete syntax
parse tree
dependency tree
head percolation, heuristics

From AST to parse tree

- link words to subtrees and hide functions

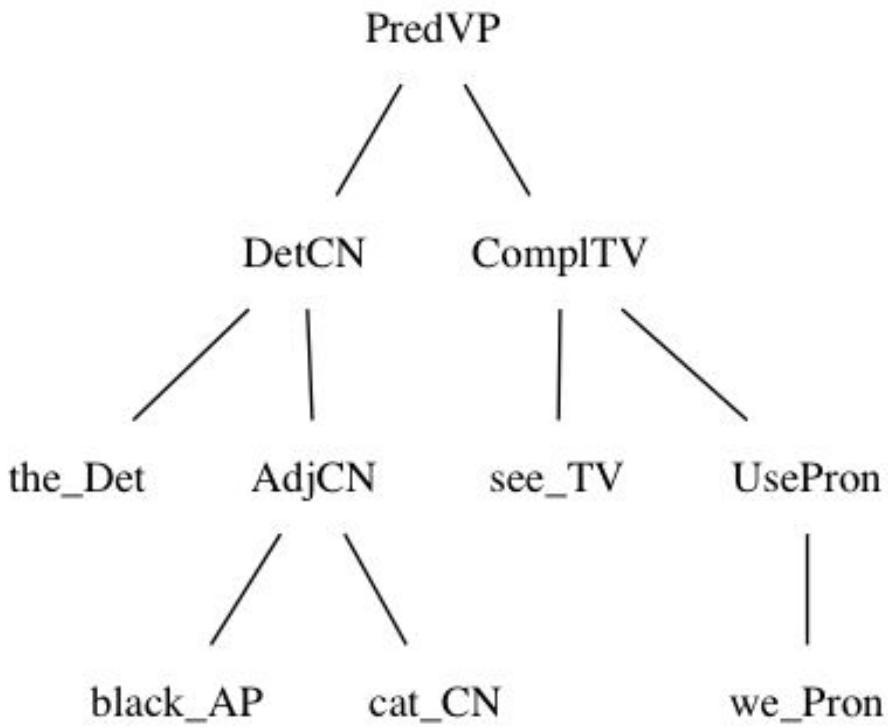

PredVP
DetCN
ComplTV
the_Det
AdjCN
see_TV
UsePron
black_AP
cat_CN
we_Pron

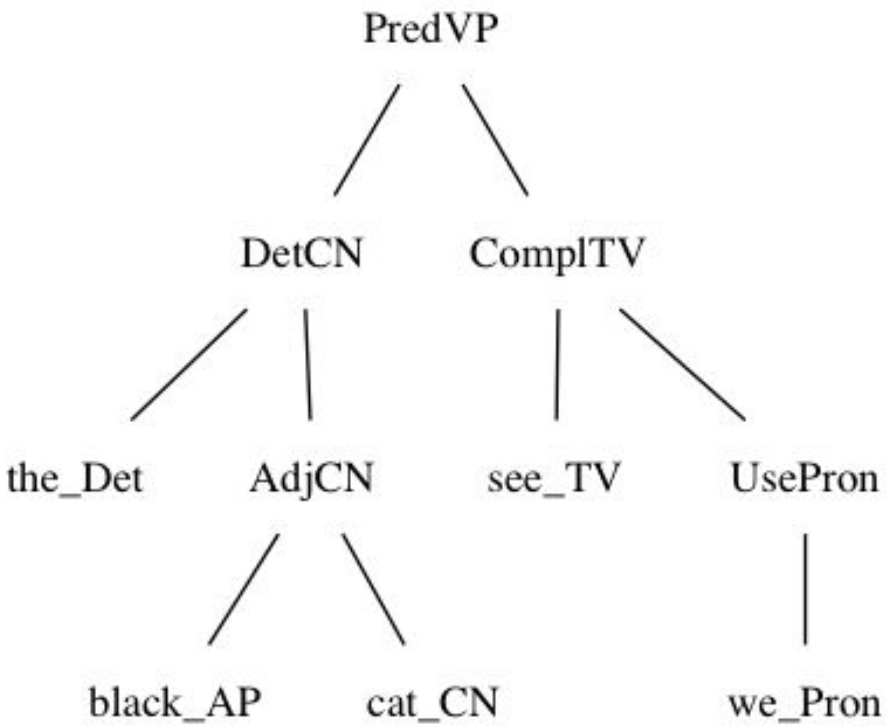

S
NP
VP
Det
CN
TV
NP
AP
CN
Pron
le
chat
noir
nous
voit


S
NP
VP
Det
CN
NP
TV
CN
AP
Pron
le
chat
noir
nous
voit

From AST to dependency tree

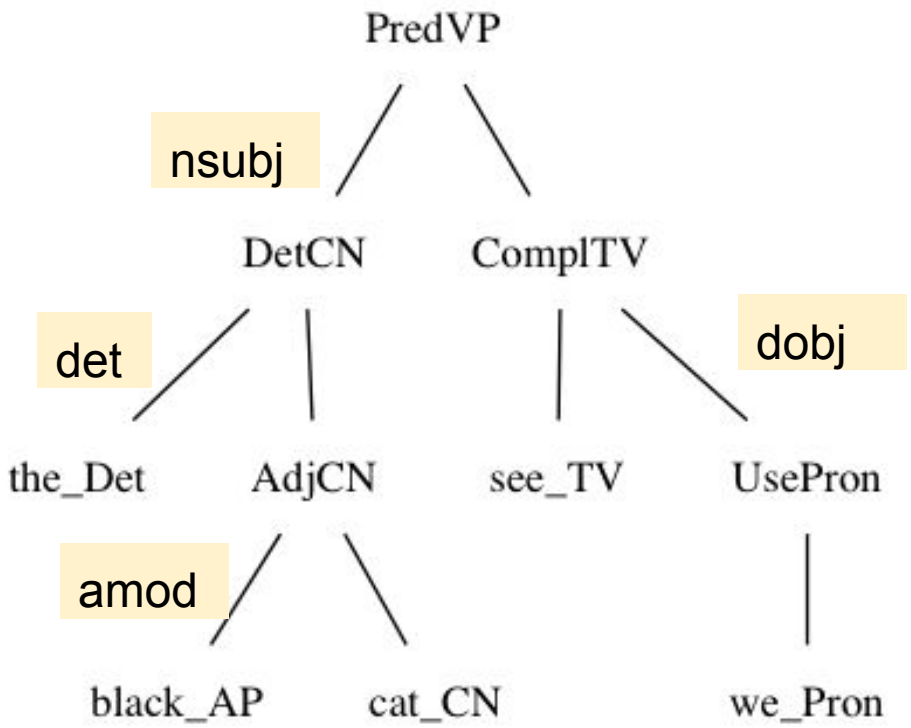
- put labels on subtrees and hide functions


PredVP
DetCN
ComplTV
the_Det
AdjCN
see_TV
UsePron
black_AP
cat_CN
we_Pron



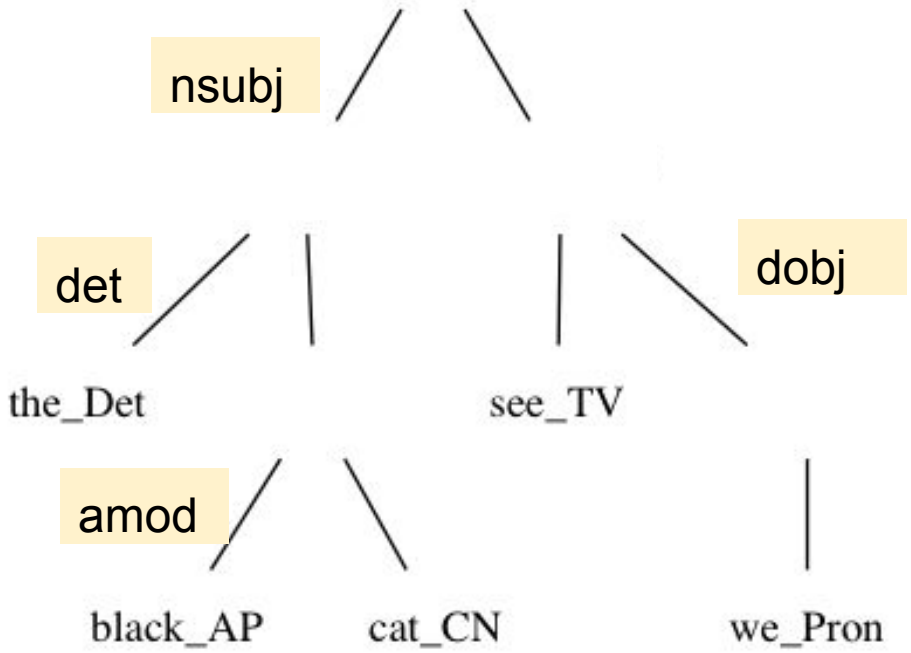
Dependency configuration

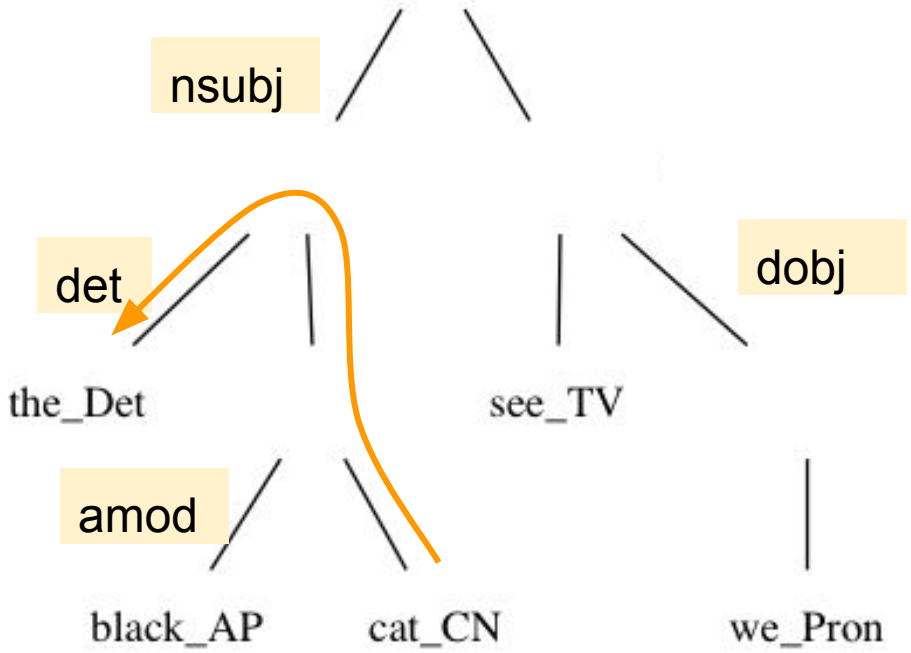
```
PredVP      nsubj head
ComplTV     head  dobj
DetCN       det   head
AdjCN       amod  head
```

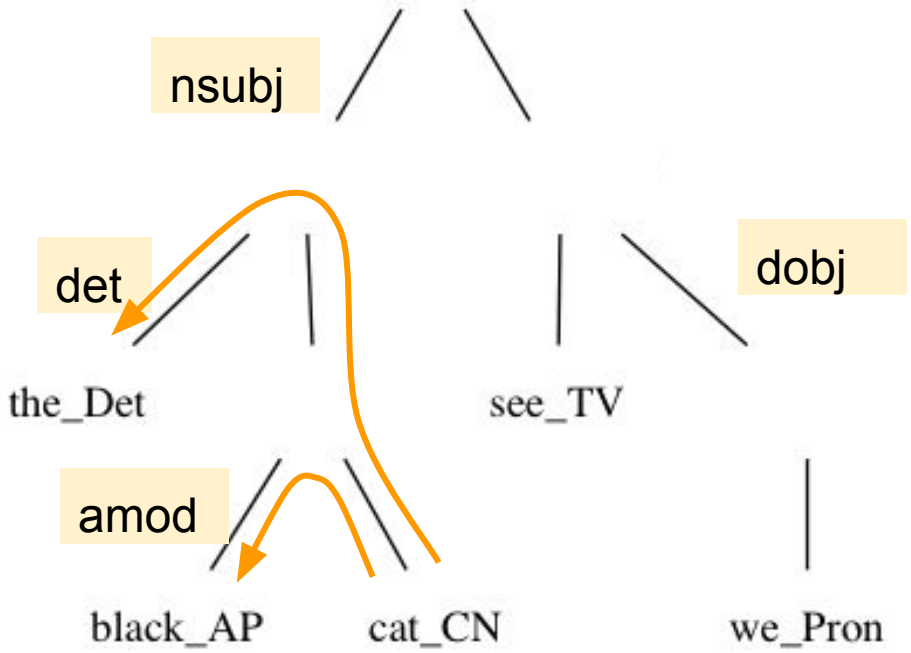


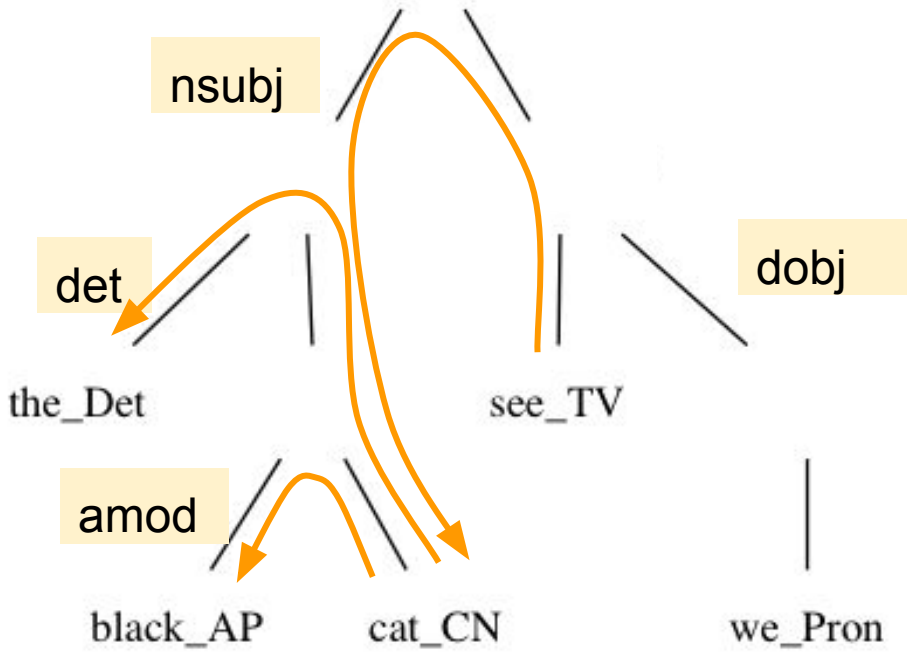
Dependency configuration

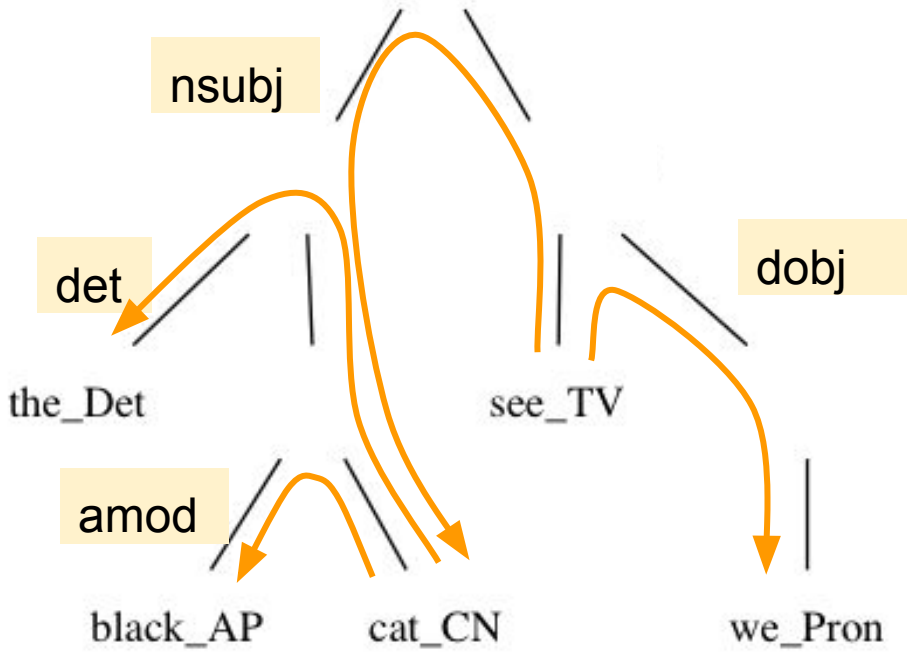
PredVP	nsubj	head
ComplTV	head	dobj
DetCN	det	head
AdjCN	amod	head

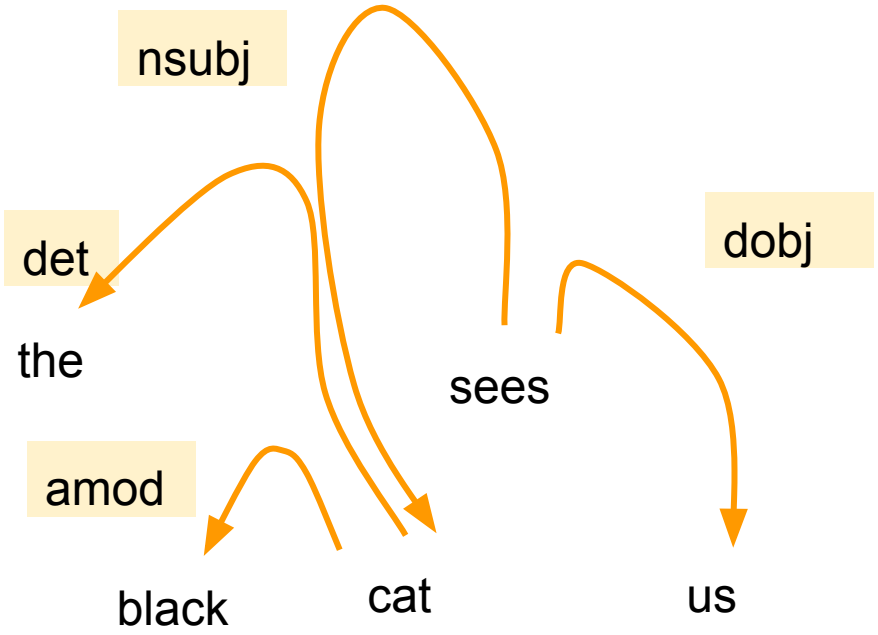

nsubj
det
dobj
the_Det
see_TV
amod
black_AP
cat_CN
we_Pron


nsubj
det
dobj
the_Det
see_TV
amod
black_AP
cat_CN
we_Pron


nsubj
det
dobj
the_Det
see_TV
amod
black_AP
cat_CN
we_Pron

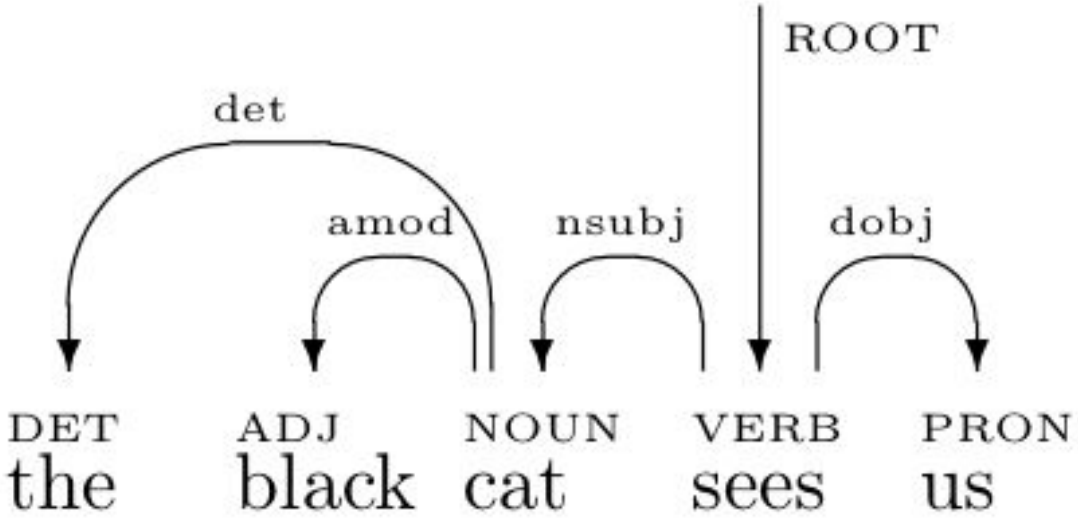

nsubj
det
dobj
the_Det
see_TV
amod
black_AP
cat_CN
we_Pron

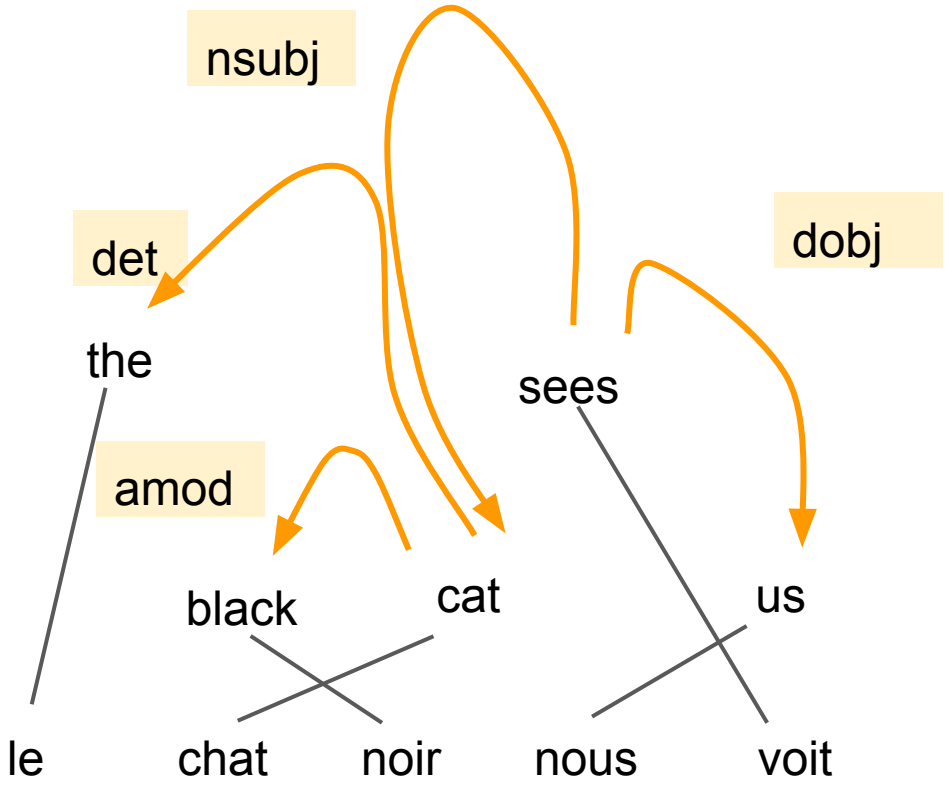

nsubj
det
dobj
amod
the_Det
see_TV
black_AP
cat_CN
we_Pron

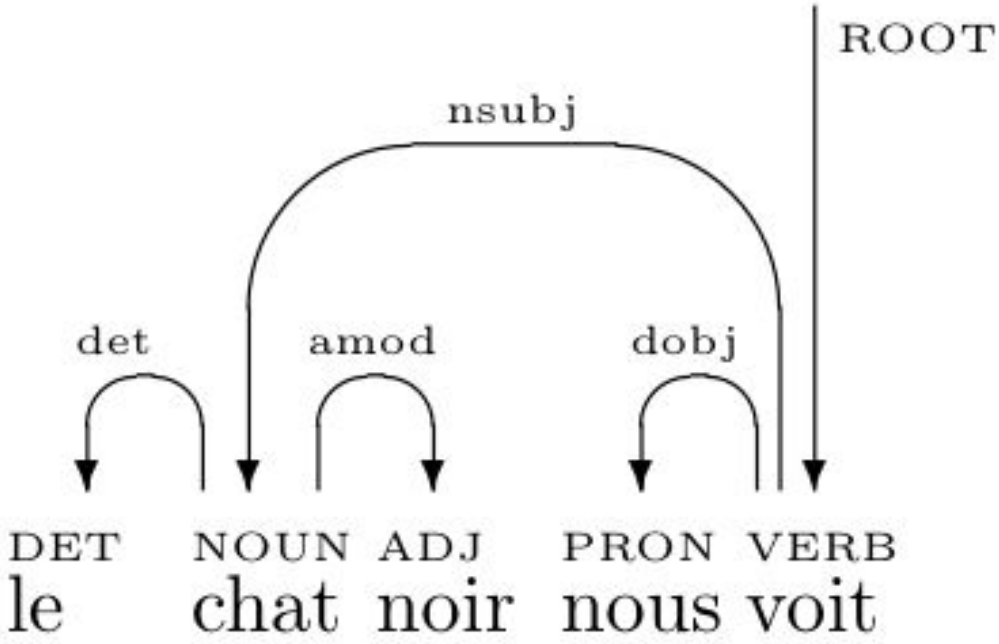

nsubj
det
the
dobj
sees
amod
black
cat
us

POS configuration

Det	DET
AP	ADJ
CN	NOUN
TV	VERB
Pron	PRON




nsubj
det
dobj
the
sees
amod
black
cat
us
le
chat
noir
nous
voit


ROOT
nsubj
det
amod
dobj
DET NOUN ADJ PRON VERB
le chat noir nous voit

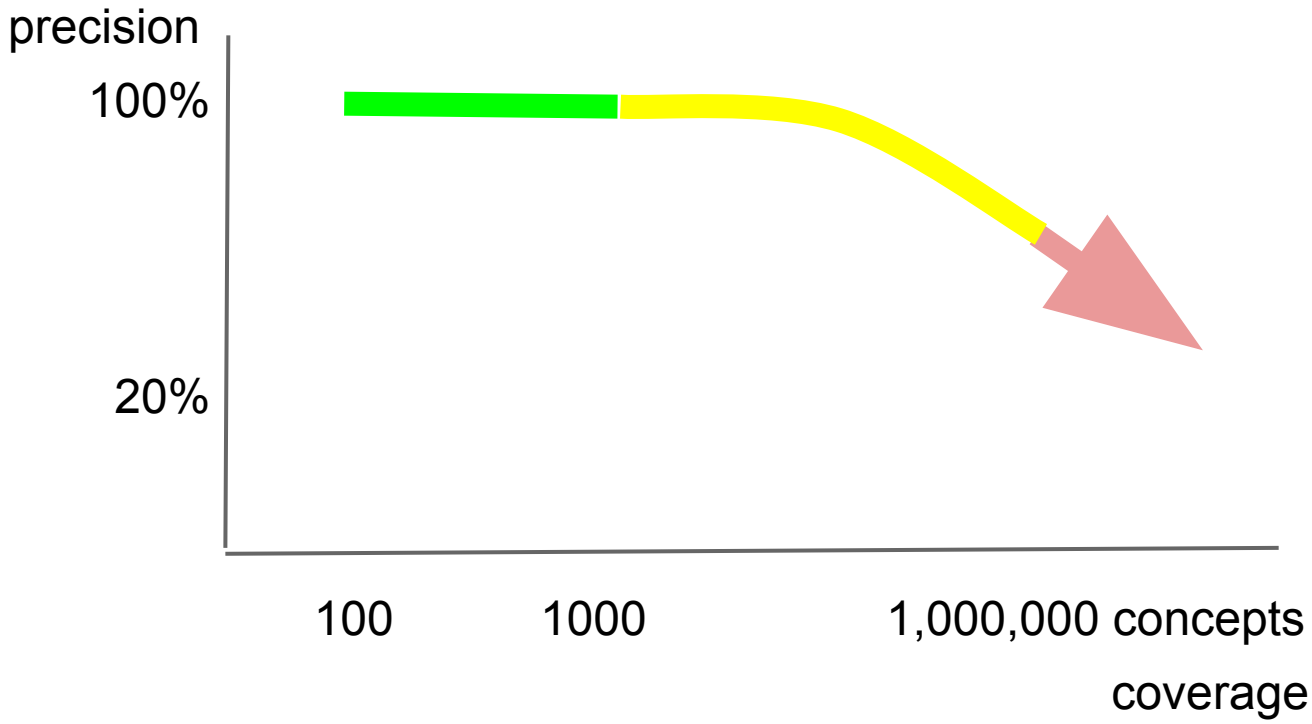
GF	Universal Dependencies
Common abstract syntax	Shared POS tags and labels
30+ languages	30+ languages
Parsing and generation	Parsing only
Mechanic mapping to UD	Partial retrieval of GF
Work on grammar writing	Work on tree annotation

P. Kolachina and A. Ranta, From Abstract Syntax to Universal Dependencies, LiLT 13(3), 2016 (to appear).

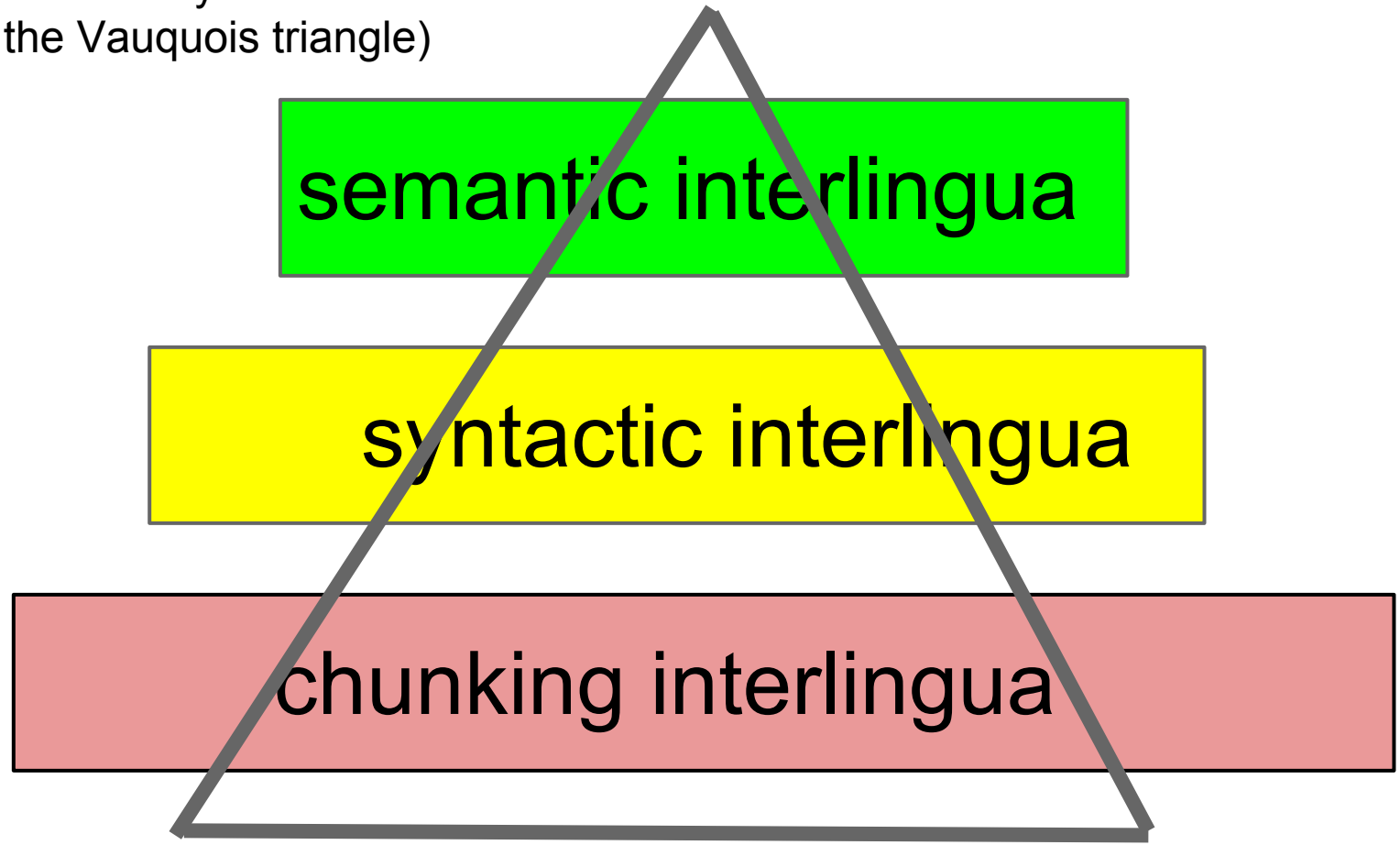
Part 3:

Scaling up

Graceful degradation



Translation layers
(cf. the Vauquois triangle)



Abstract syntax trees

my daughter is hungry

Hungry (Poss I Daughter)

Abstract syntax trees

my daughter is hungry

Hungry (Poss I Daughter)

Pred (Det (Poss i_NP) daughter_N)) (CompAP hungry_A)

Abstract syntax trees

my daughter is hungry

Hungry (Poss I Daughter)

Pred (Det (Poss i_NP) daughter_N)) (CompAP hungry_A)

[DetChunk (Poss i_NP), NChunk daughter_N, copulaChunk, AChunk hungry_A]

Compositional translation

my daughter is hungry

→

Hungry (Poss I Daughter)

ma fille a faim

Pred (Det (Poss i_NP) daughter_N)) (CompAP hungry_A)

ma fille est affamée

[DetChunk (Poss i_NP), NChunk daughter_N, copulaChunk, AChunk hungry_A]

mon fille est affamé

How far is the airport from the hotel? **meaning**

Quelle est la distance de l'hôtel à l'aéroport?

The vice dean kicked the bucket. **syntax**

Le doyen de vice donnait un coup de pied au panier.

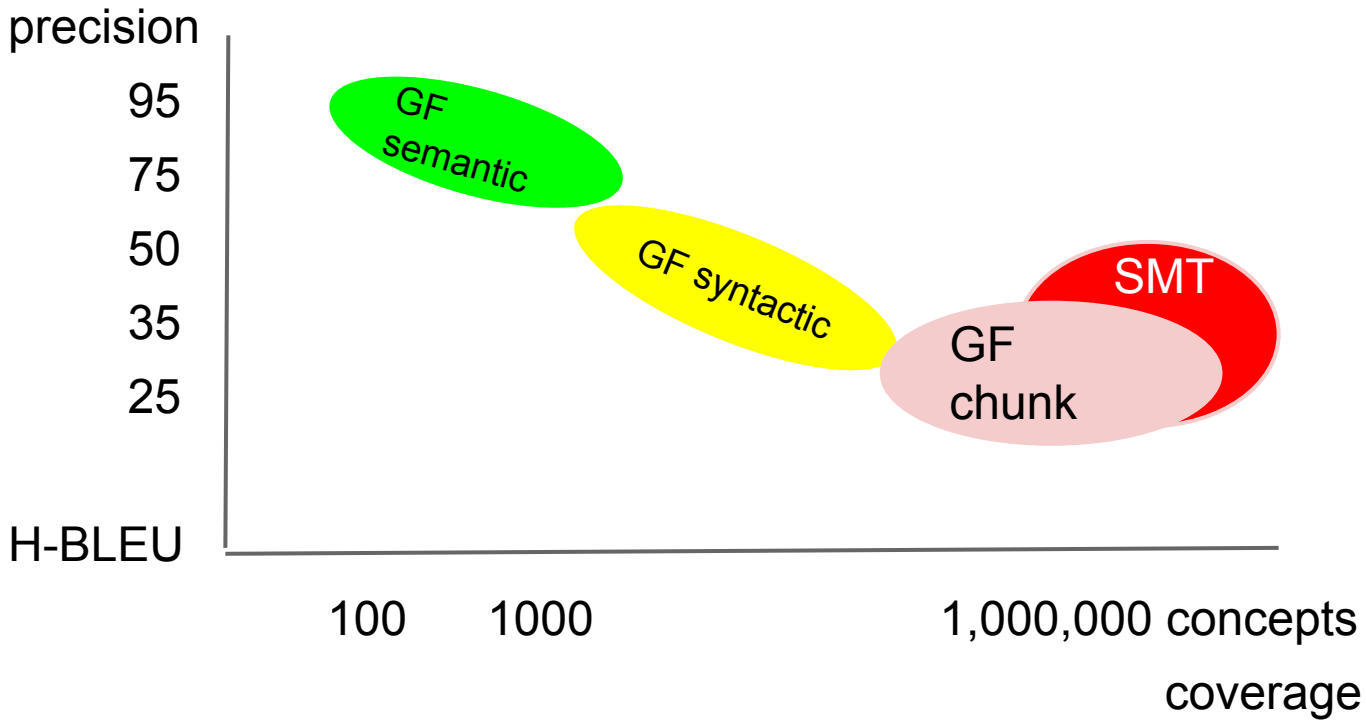
We copes from grammar errors by this way. **chunks**

Nous se débrouille par cette route d'erreurs de grammaire.

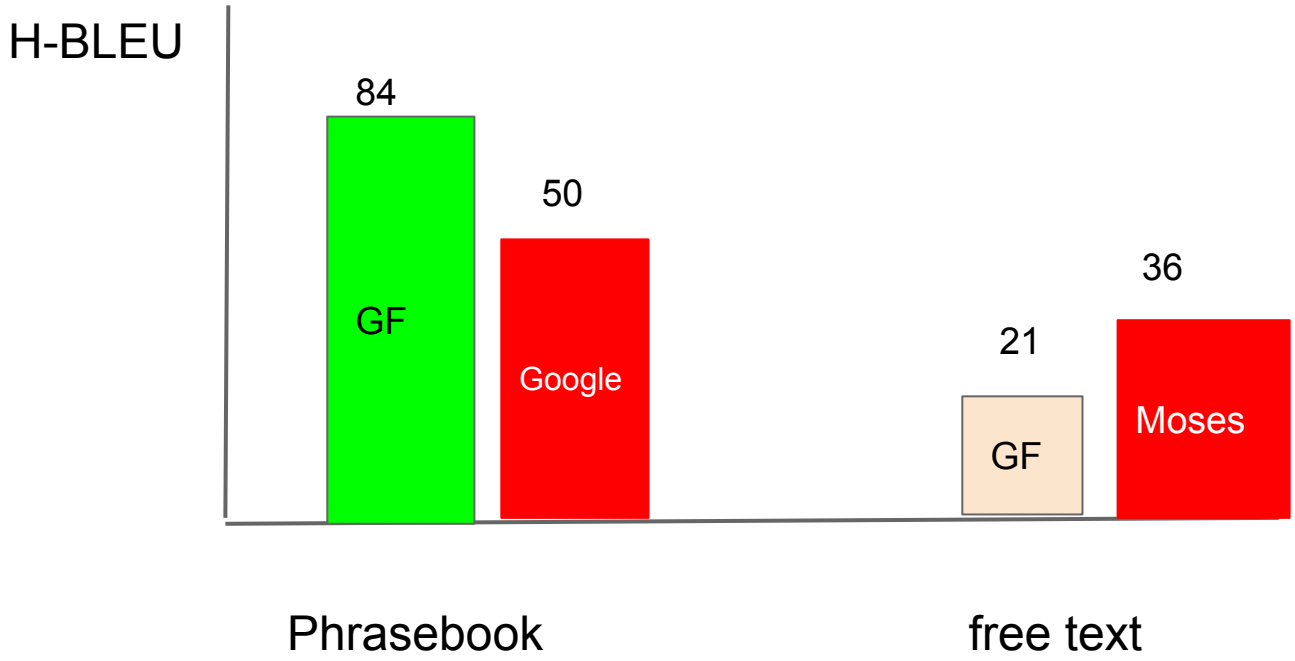
Demo:
Wide coverage translator on the cloud

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

Where we are now



Some scores (English-Chinese)



A.Ranta, Yan Tian and Qiao Haiyan, Chinese in the Grammatical Framework: Grammar, Translation, and Other Applications, ACL-SIGHAN 2015

Swe TD to Fin, Ger, Spa

	GF, BLEU	Google, BLEU
Fin, CNL	**77**	31
Fin, robust	**31**	20
Fin, all	**73**	28
Ger, CNL	**75**	37
Ger, robust	33	**34**
Ger, all	**73**	37
Spa, CNL	**76**	28
Spa, robust	**39**	25
Spa, all	**74**	38

Ranta, Unger and Vidal, ACL-GEAF 2015.

GF vs. Statistical Translation

+ grammatical correctness
+ feedback: trees, colours
+ less dependent on language data
+ compact size of multilingual systems

GF vs. Statistical Translation

+ grammatical correctness
+ feedback: trees, colours
+ less dependent on language data
+ compact size of multilingual systems
- non-compositional idioms
- contextual disambiguation

- cases for hybrid methods

Part 4:

Future trends with GF

Controlled language tasks

Localization

Text robots

Data-driven documentation

Precision translation

Business: Digital Grammars AB

Hybridization

GF-driven

Statistical disambiguation

Grammar automation by machine learning

Statistics-driven, exploiting GF

Factored models on abstract syntax

UD treebank bootstrapping

John likes Paris

Parigi piace a John

John likes Paris	*Parigi piace a John*
mkCl john_NP like_V2 paris_NP	mkCl paris_NP piacere_V2 john_NP

John likes Paris	*Parigi piace a John*
mkCl john_NP like_V2 paris_NP	mkCl paris_NP piacere_V2 john_NP
mkCl x like_V2 y	mkCl y piacere_V2 x

John likes Paris	*Parigi piace a John*
mkCl john_NP like_V2 paris_NP	mkCl paris_NP piacere_V2 john_NP
mkCl x like_V2 y	mkCl y piacere_V2 x
we like Paris	*Parigi ci piace*
who doesn't like Naples	*a chi non piacerebbe Napoli*
…	*…*

Mission still to accomplish: formalizing the grammars of the world!

Constant work on

- more languages
- wider coverage
- better quality
- usability in other contexts

Take home points

GF

- Grammar engineering without tears
- Multilinguality via abstract syntax
- “Compiling natural language”
- Interlingual translation
- Domain adaptation
- Libraries for over 30 languages, growing

http://www.grammaticalframework.org