Explainable Machine Translation

Aarne Ranta

Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

and

Digital Grammars AB

Logic and Machine Learning, Gothenburg, 12-13 June 2017

Explainable Machine Translation with Interlingual Trees as Certificates

Aarne Ranta
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

and

Digital Grammars AB

Logic and Machine Learning, Gothenburg, 12-13 June 2017

The Next Big Disruptive Trend in Business. . .

https://disruptionhub.com/next-big-disruptive-trend-business-

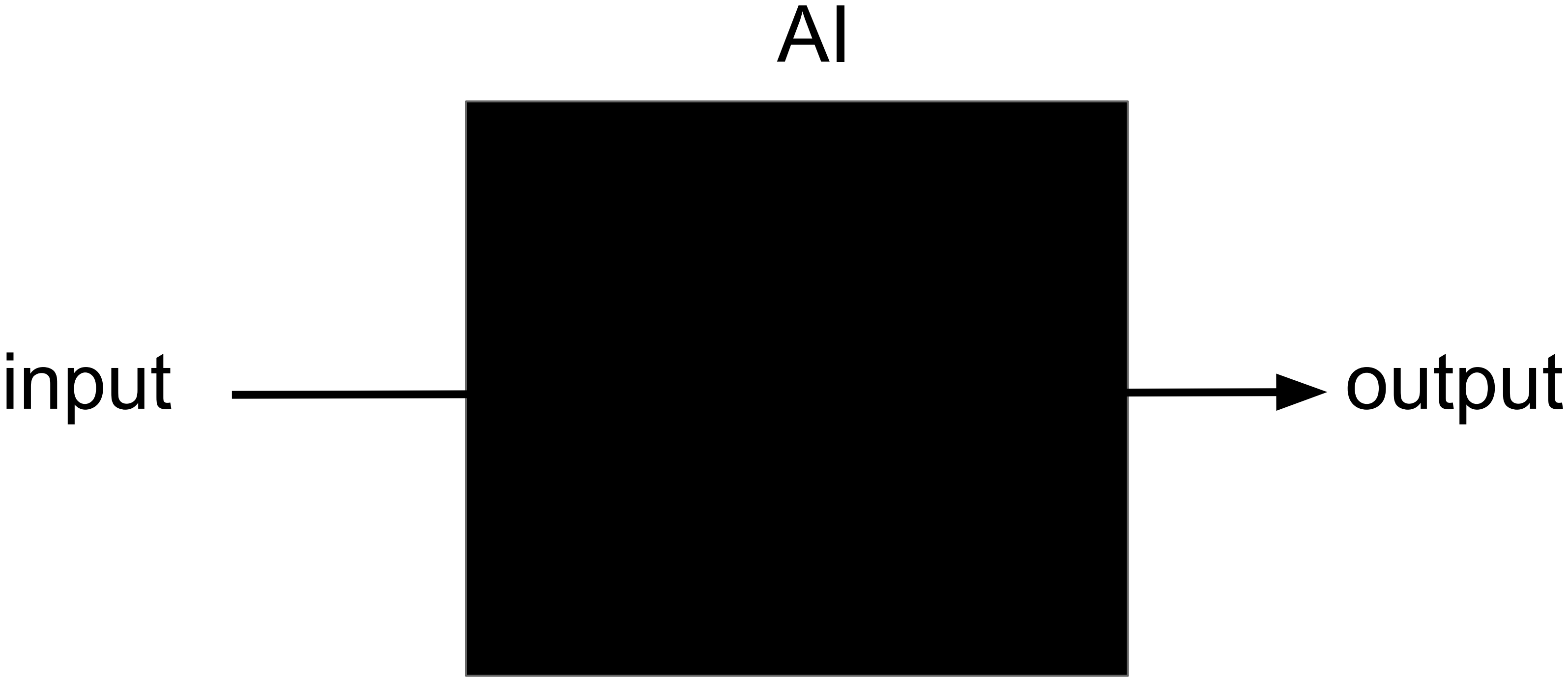
18 January 2017

The Next Big Disruptive Trend in Business. . .

Explainable AI

https://disruptionhub.com/next-big-disruptive-trend-business-explainable-ai/

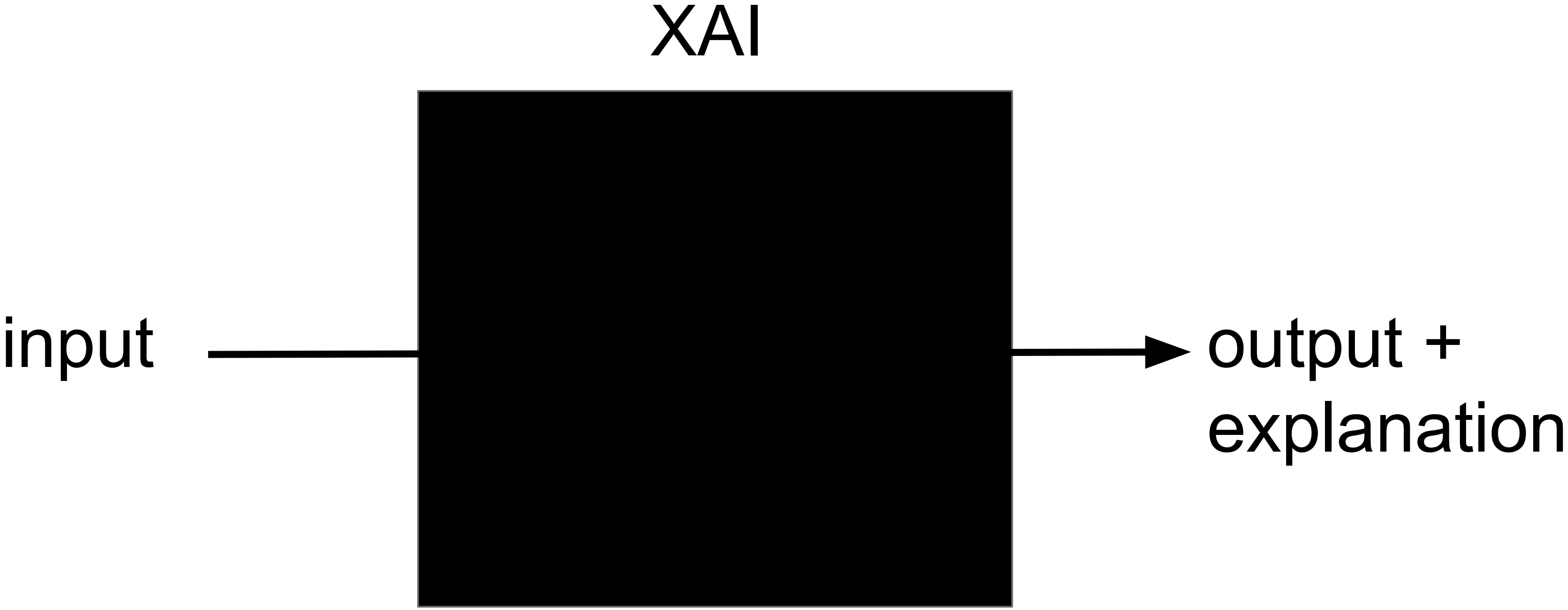
18 January 2017



AI

input

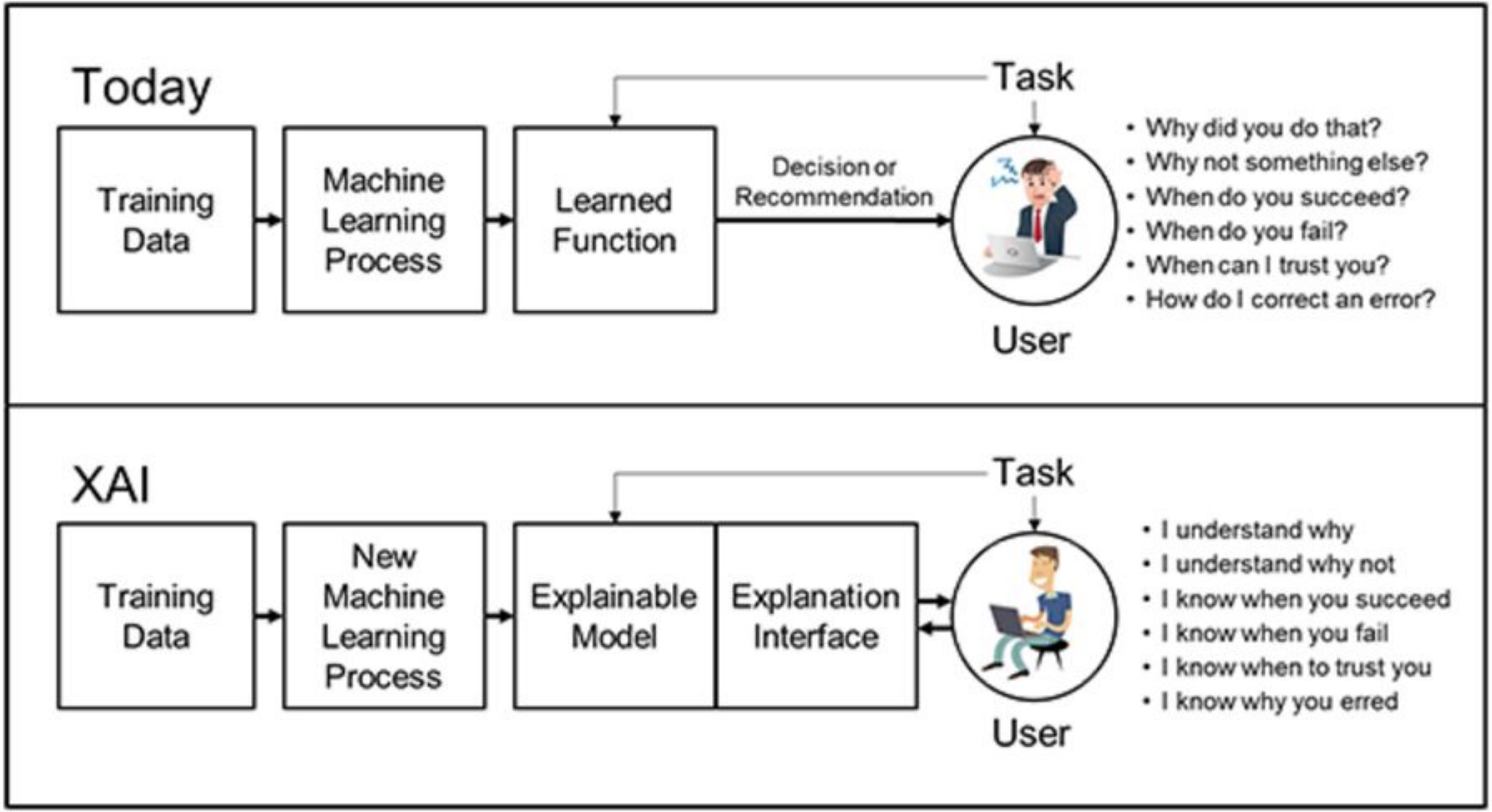
output



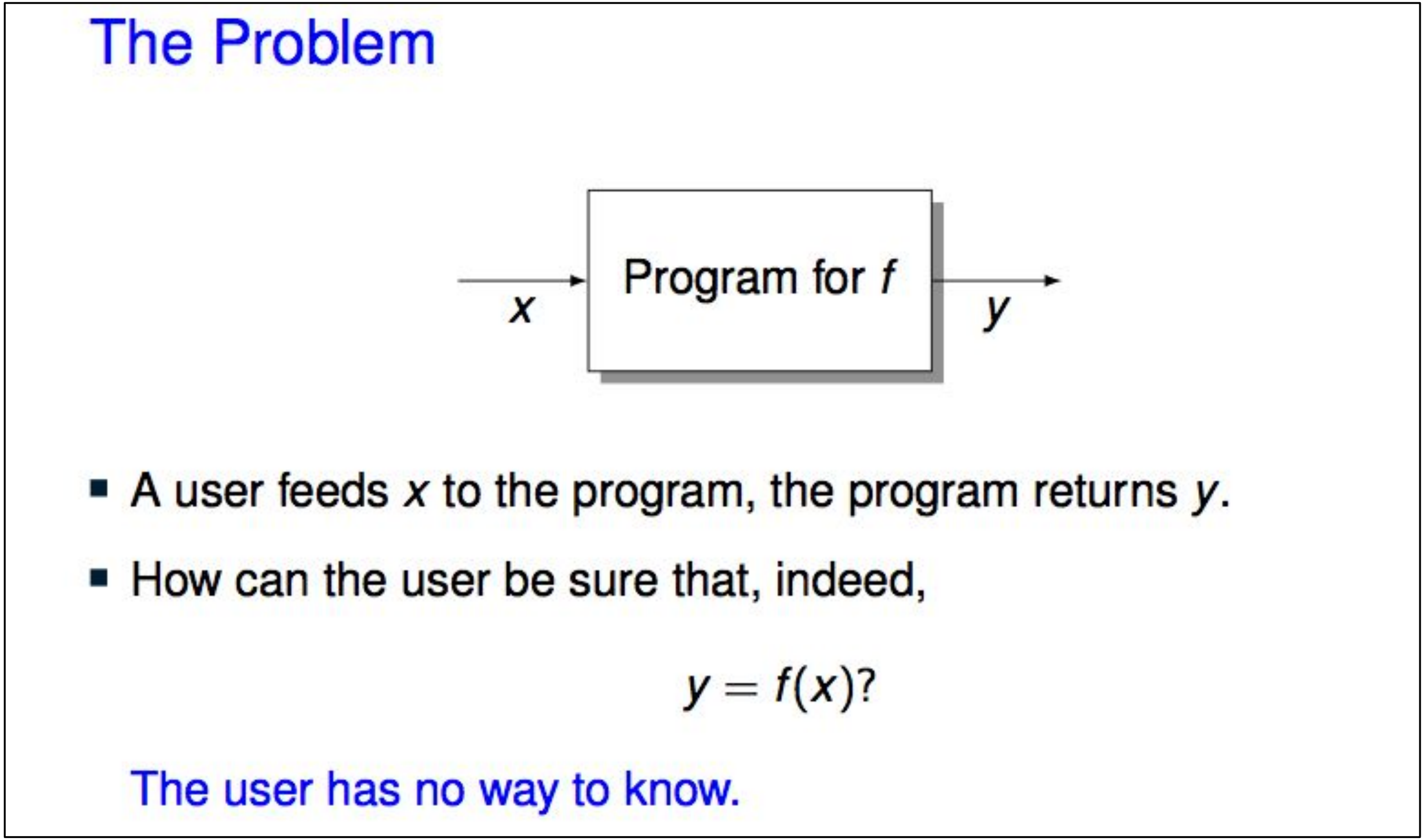
XAI

input

output + explanation



http://www.darpa.mil/program/explainable-artificial-intelligence



The Problem

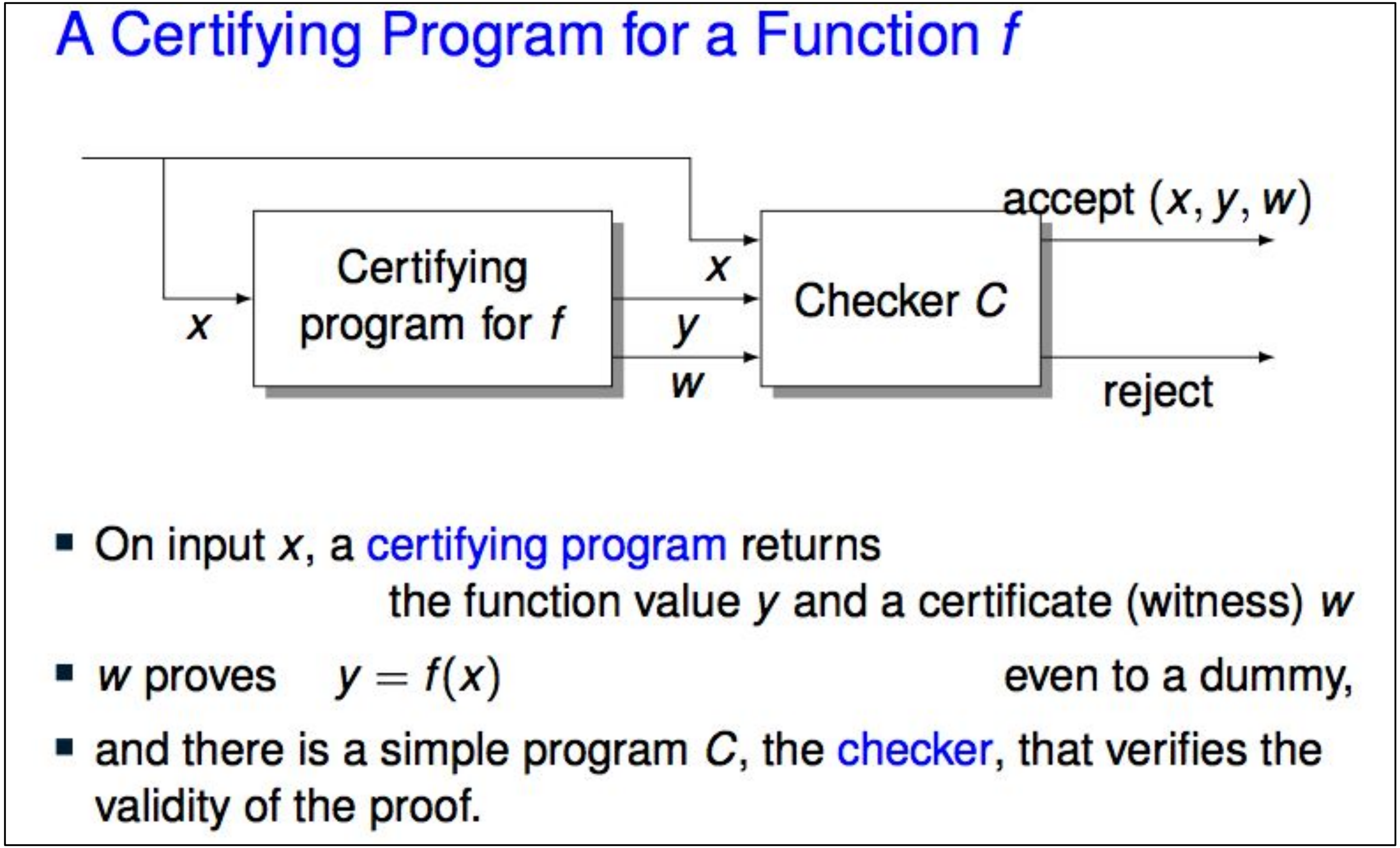
x → Program for f → y

- A user feeds x to the program, the program returns y.
- How can the user be sure that, indeed,

$$y = f(x)?$$

The user has no way to know.

Kurt Mehlhorn, Certifying Algorithms
https://people.mpi-inf.mpg.de/~mehlhorn/ftp/SAPJuly2014.pdf



A Certifying Program for a Function *f*

- On input *x*, a certifying program returns the function value *y* and a certificate (witness) *w*
- *w* proves $y = f(x)$ even to a dummy,
- and there is a simple program *C*, the checker, that verifies the validity of the proof.

Kurt Mehlhorn, Certifying Algorithms
https://people.mpi-inf.mpg.de/~mehlhorn/ftp/SAPJuly2014.pdf

Levels of evidence

	formal	
total	proof of program	

Levels of evidence

	formal	
total	proof of program	
individual	certificate of instance	

Levels of evidence

	formal	**informal**
total	proof of program	correctness by construction
individual	certificate of instance	

Levels of evidence

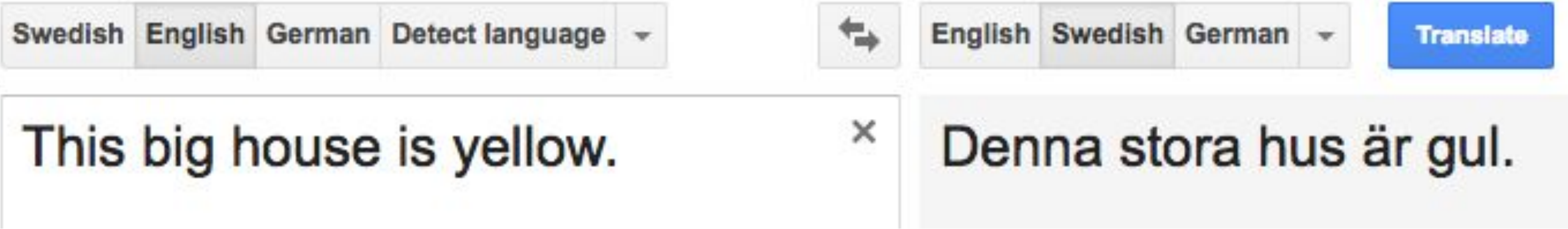
	formal	**informal**
total	proof of program	correctness by construction
individual	certificate of instance	explanation of instance

Evidence for SMT

SMT = Statistical Machine Translation

- glue together segments from aligned texts

Informal evidence: phrase alignments



This big house is yellow.

Denna stora hus är gul.

https://translate.google.com/ 13 mars 2017

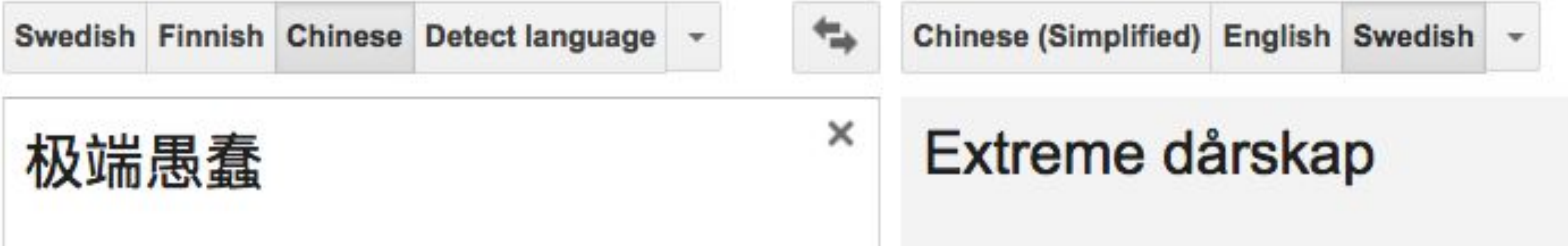
This big car is yellow.	*Denna stora bil är gul.*
This house is clean.	*Detta hus är rent.*
This big house is yellow.	*Denna stora hus är gul.*

Evidence for NMT

NMT = Neural Machine Translation

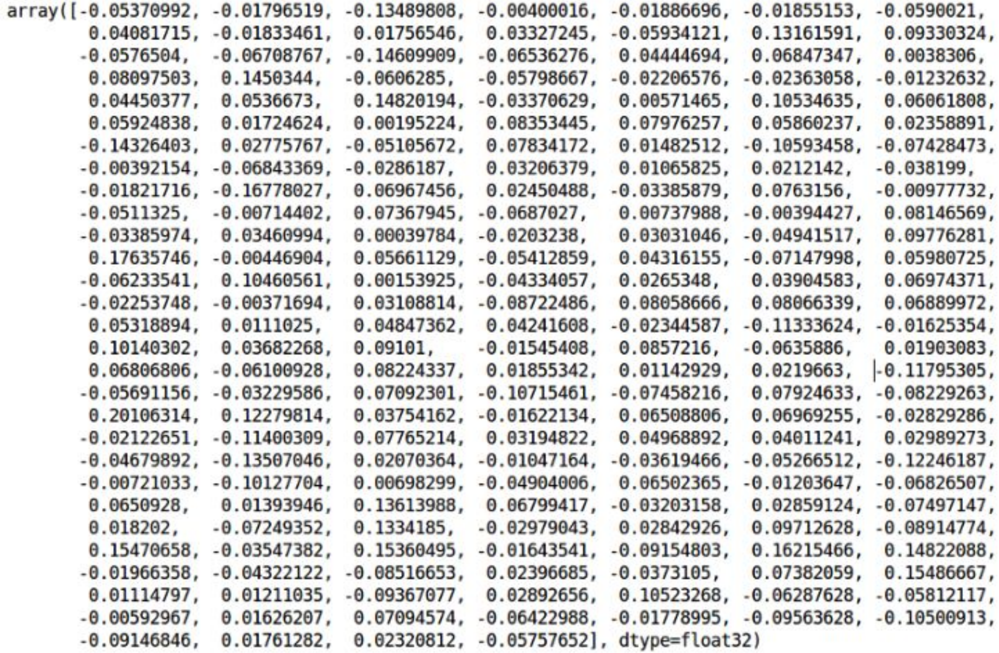
- end-to-end string conversion via a neural network

Individual explanations: word vector “interlingua”



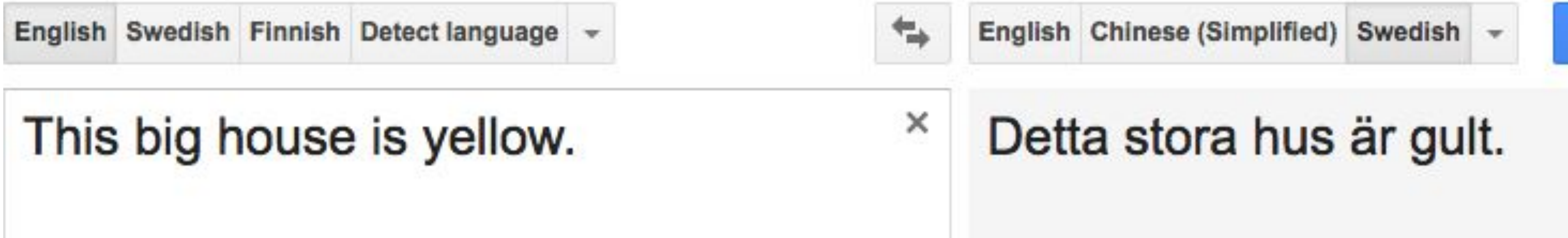
极端愚蠢

Extreme dårskap



```
array([-0.05370992, -0.01796519, -0.13489808, -0.00400016, -0.01886696, -0.01855153, -0.0590021,
        0.04081715, -0.01833461,  0.01756546,  0.03327245, -0.05934121,  0.13161591,  0.09330324,
       -0.0576504,  -0.06708767, -0.14609909, -0.06536276,  0.04444694,  0.06847347,  0.0038306,
        0.08097503,  0.1450344,  -0.0606285,  -0.05798667, -0.02206576, -0.02363058, -0.01232632,
        0.04450377,  0.0536673,   0.14820194, -0.03370629,  0.00571465,  0.10534635,  0.06061808,
        0.05924838,  0.01724624,  0.00195224,  0.08353445,  0.07976257,  0.05860237,  0.02358891,
       -0.14326403,  0.02775767, -0.05105672,  0.07834172,  0.01482512, -0.10593458, -0.07428473,
       -0.00392154, -0.06843369, -0.0286187,   0.03206379,  0.01065825,  0.0212142,  -0.038199,
       -0.01821716, -0.16778027,  0.06967456,  0.02450488, -0.03385879,  0.0763156,  -0.00977732,
       -0.0511325,  -0.00714402,  0.07367945, -0.0687027,   0.00737988, -0.00394427,  0.08146569,
       -0.03385974,  0.03460994,  0.00039784, -0.0203238,   0.03031046, -0.04941517,  0.09776281,
        0.17635746, -0.00446904,  0.05661129, -0.05412859,  0.04316155, -0.07147998,  0.05980725,
       -0.06233541,  0.10460561,  0.00153925, -0.04334057,  0.0265348,   0.03904583,  0.06974371,
       -0.02253748, -0.00371694,  0.03108814, -0.08722486,  0.08058666,  0.08066339,  0.06889972,
        0.05318894,  0.0111025,   0.04847362,  0.04241608, -0.02344587, -0.11333624, -0.01625354,
        0.10140302,  0.03682268,  0.09101,    -0.01545408,  0.0857216,  -0.0635886,   0.01903083,
        0.06806806, -0.06100928,  0.08224337,  0.01855342,  0.01142929,  0.0219663,  -0.11795305,
       -0.05691156, -0.03229586,  0.07092301, -0.10715461, -0.07458216,  0.07924633, -0.08229263,
        0.20106314,  0.12279814,  0.03754162, -0.01622134,  0.06508806,  0.06969255, -0.02829286,
       -0.02122651, -0.11400309,  0.07765214,  0.03194822,  0.04968892,  0.04011241,  0.02989273,
       -0.04679892, -0.13507046,  0.02070364, -0.01047164, -0.03619466, -0.05266512, -0.12246187,
       -0.00721033, -0.10127704,  0.00698299, -0.04904006,  0.06502365, -0.01203647, -0.06826507,
        0.0650928,   0.01393946,  0.13613988,  0.06799417, -0.03203158,  0.02859124, -0.07497147,
        0.018202,   -0.07249352,  0.1334185,  -0.02979043,  0.02842926,  0.09712628, -0.08914774,
        0.15470658, -0.03547382,  0.15360495, -0.01643541, -0.09154803,  0.16215466,  0.14822088,
       -0.01966358, -0.04322122, -0.08516653,  0.02396685, -0.0373105,   0.07382059,  0.15486667,
        0.01114797,  0.01211035, -0.09367077,  0.02892656,  0.10523268, -0.06287628, -0.05812117,
       -0.00592967,  0.01626207,  0.07094574, -0.06422988, -0.01778995, -0.09563628, -0.10500913,
       -0.09146846,  0.01761282,  0.02320812, -0.05757652], dtype=float32)
```

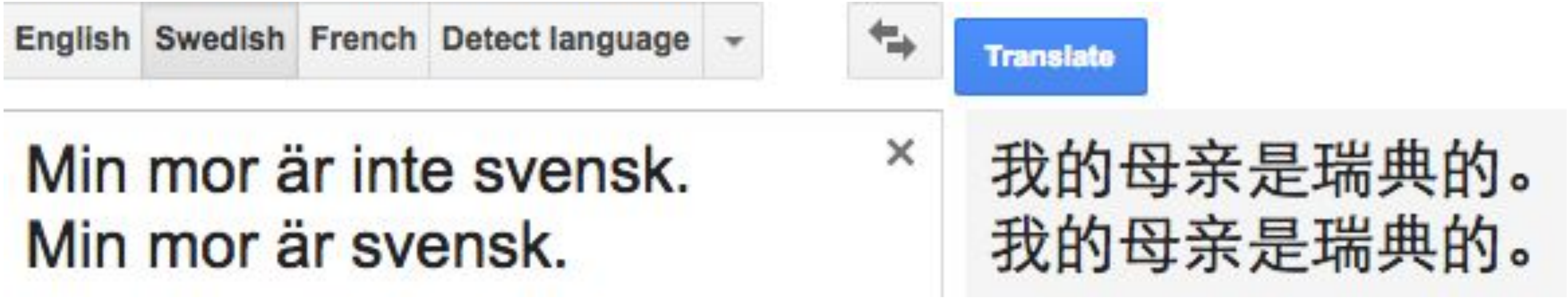
https://gab41.lab41.org/can-word-vectors-help-predict-whether-your-chinese-tweet-gets-censored-711e7682d12f



This big house is yellow.

Detta stora hus är gult.

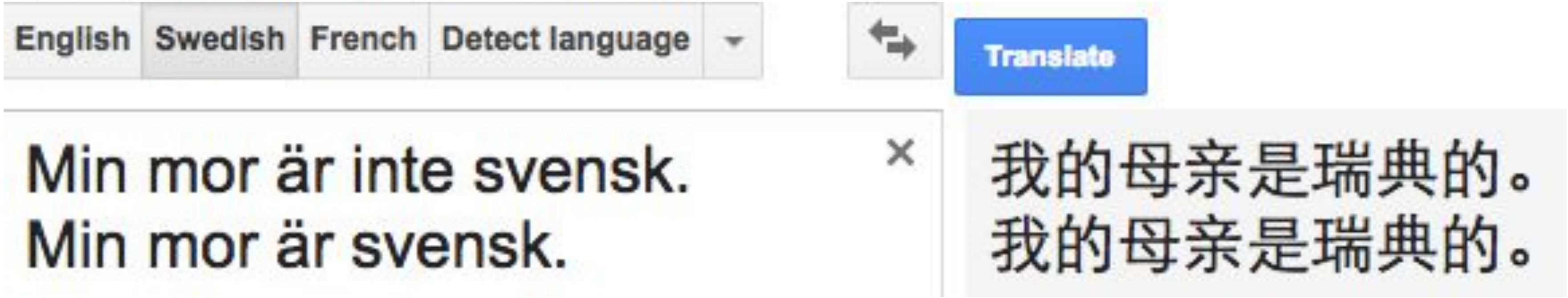
https://translate.google.com/ 5 May 2017



Min mor är inte svensk.
Min mor är svensk.

我的母亲是瑞典的。
我的母亲是瑞典的。

https://translate.google.com/ 5 May 2017

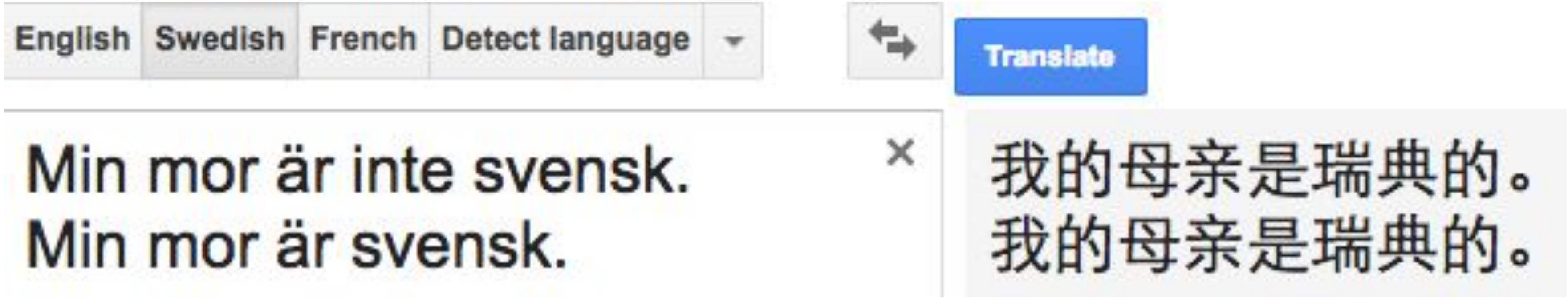


Min mor är inte svensk.
Min mor är svensk.

我的母亲是瑞典的。
我的母亲是瑞典的。

Possible evidence: translation to some language you know

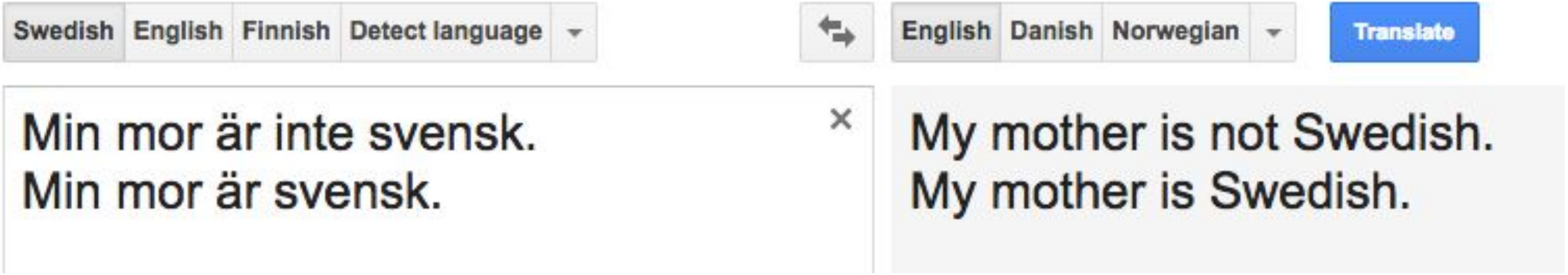
https://translate.google.com/ 5 May 2017



Min mor är inte svensk.
Min mor är svensk.

我的母亲是瑞典的。
我的母亲是瑞典的。

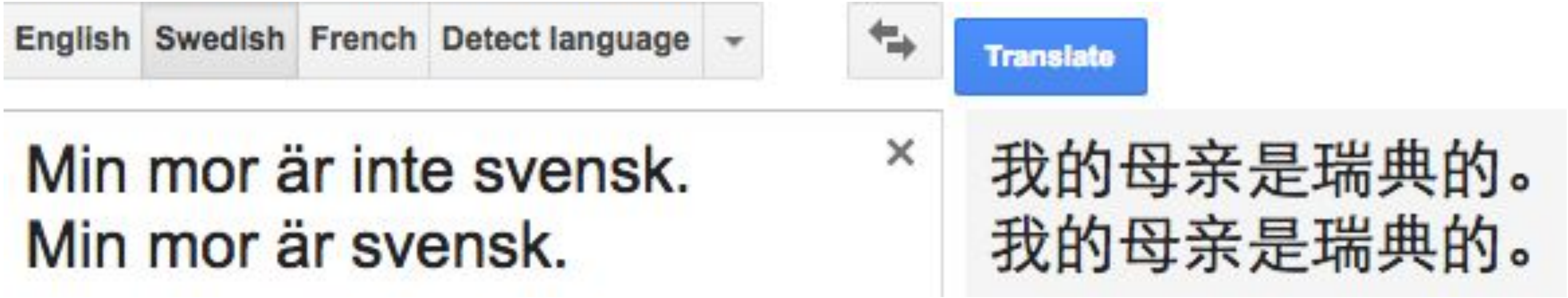
Possible evidence: translation to some language you know



Min mor är inte svensk.
Min mor är svensk.

My mother is not Swedish.
My mother is Swedish.

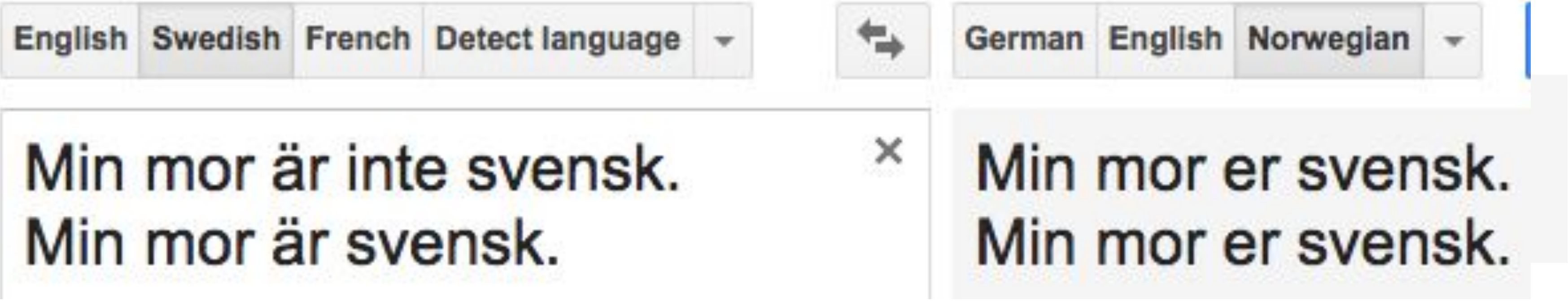
https://translate.google.com/ 5 May 2017



Min mor är inte svensk.
Min mor är svensk.

我的母亲是瑞典的。
我的母亲是瑞典的。

Possible evidence: translation to some language you know



Min mor är inte svensk.
Min mor är svensk.

Min mor er svensk.
Min mor er svensk.

https://translate.google.com/ 5 May 2017

From SMT to NMT

BLEU (max 1.0)

SMT	NMT
0.37	0.41

Fluency (max 6.0)

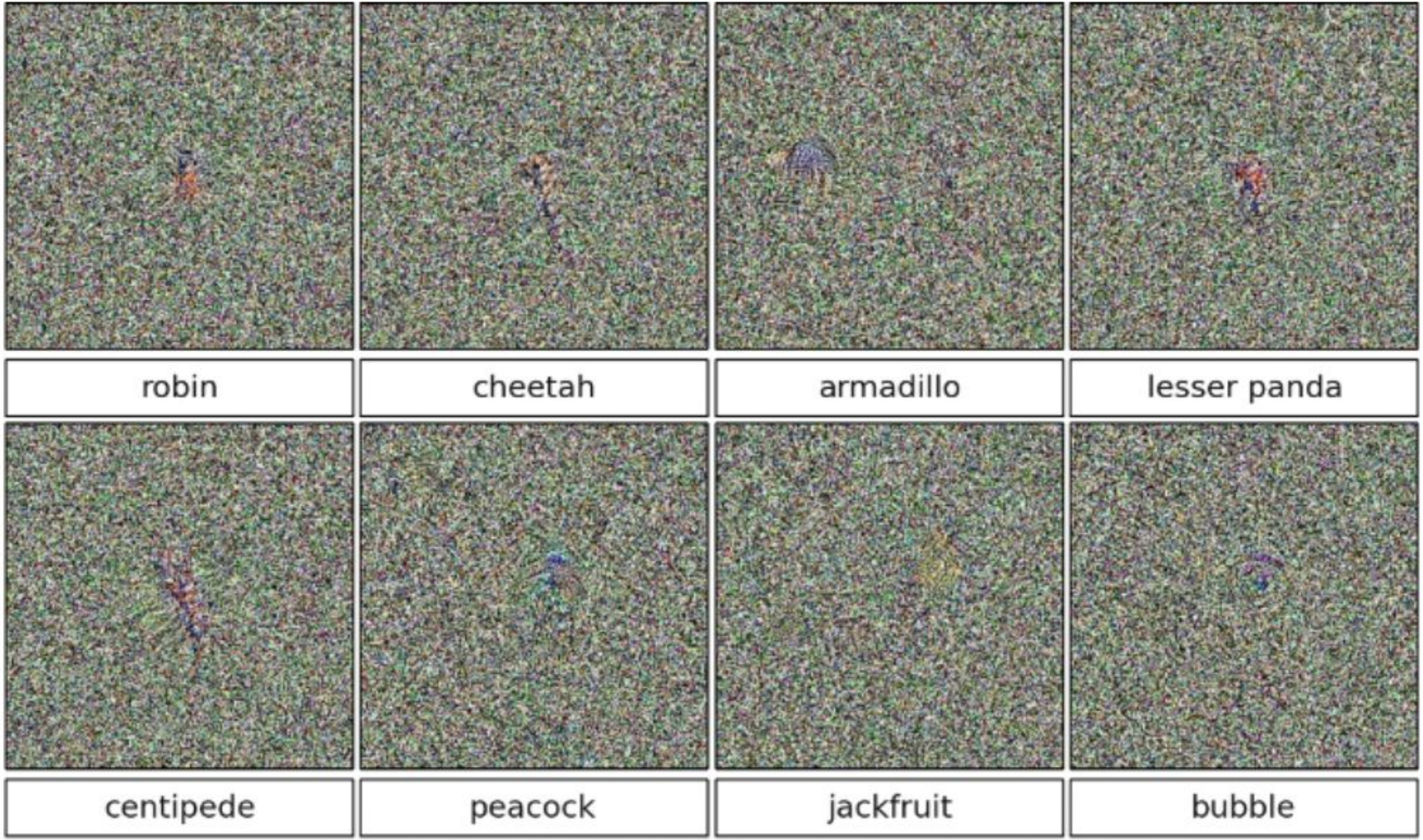
SMT	NMT	human
3.87	4.44	4.82

Wu & al, Bridging the Gap Between Machine and Human Translation, 2016

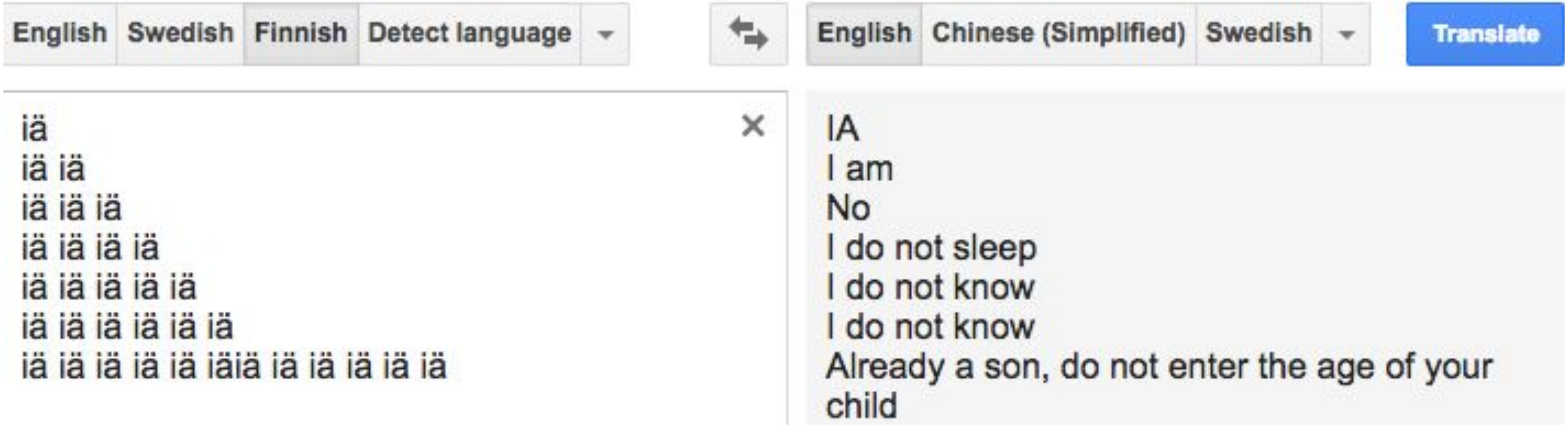
From SMT to NMT

+ improved average scores
+ increased fluency

- harder to predict
- harder to explain



Nguyen & al, Deep Neural Networks are Easily Fooled, CVPR’15, 2015.



iä
iä iä
iä iä iä
iä iä iä iä
iä iä iä iä iä
iä iä iä iä iä iä
iä iä iä iä iä iäiä iä iä iä iä iä

IA
I am
No
I do not sleep
I do not know
I do not know
Already a son, do not enter the age of your child

https://translate.google.com/ 5 maj 2017

XMT: our proposal

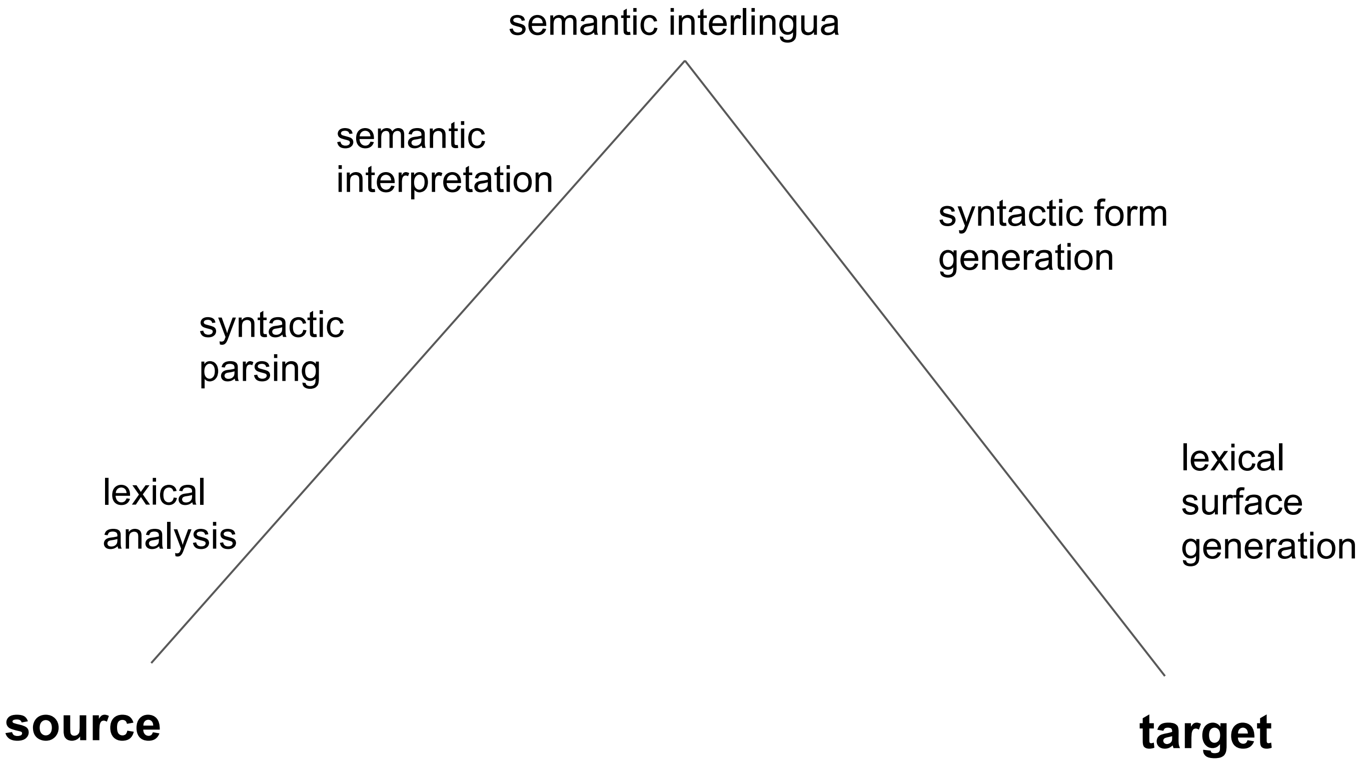
- Explainable Machine Translation

What to verify in translation

1. The output is a valid expression of the target language

2. The output has the same meaning as the input

The Vauquois triangle answer



semantic interlingua

semantic interpretation

syntactic parsing

lexical analysis

syntactic form generation

lexical surface generation

source

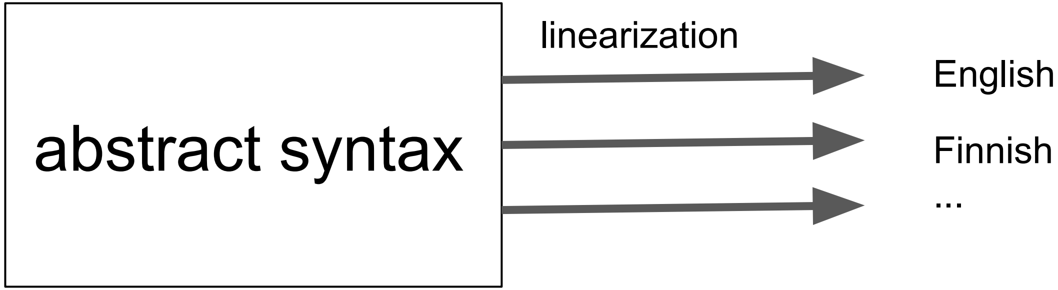
target

GF = Grammatical Framework

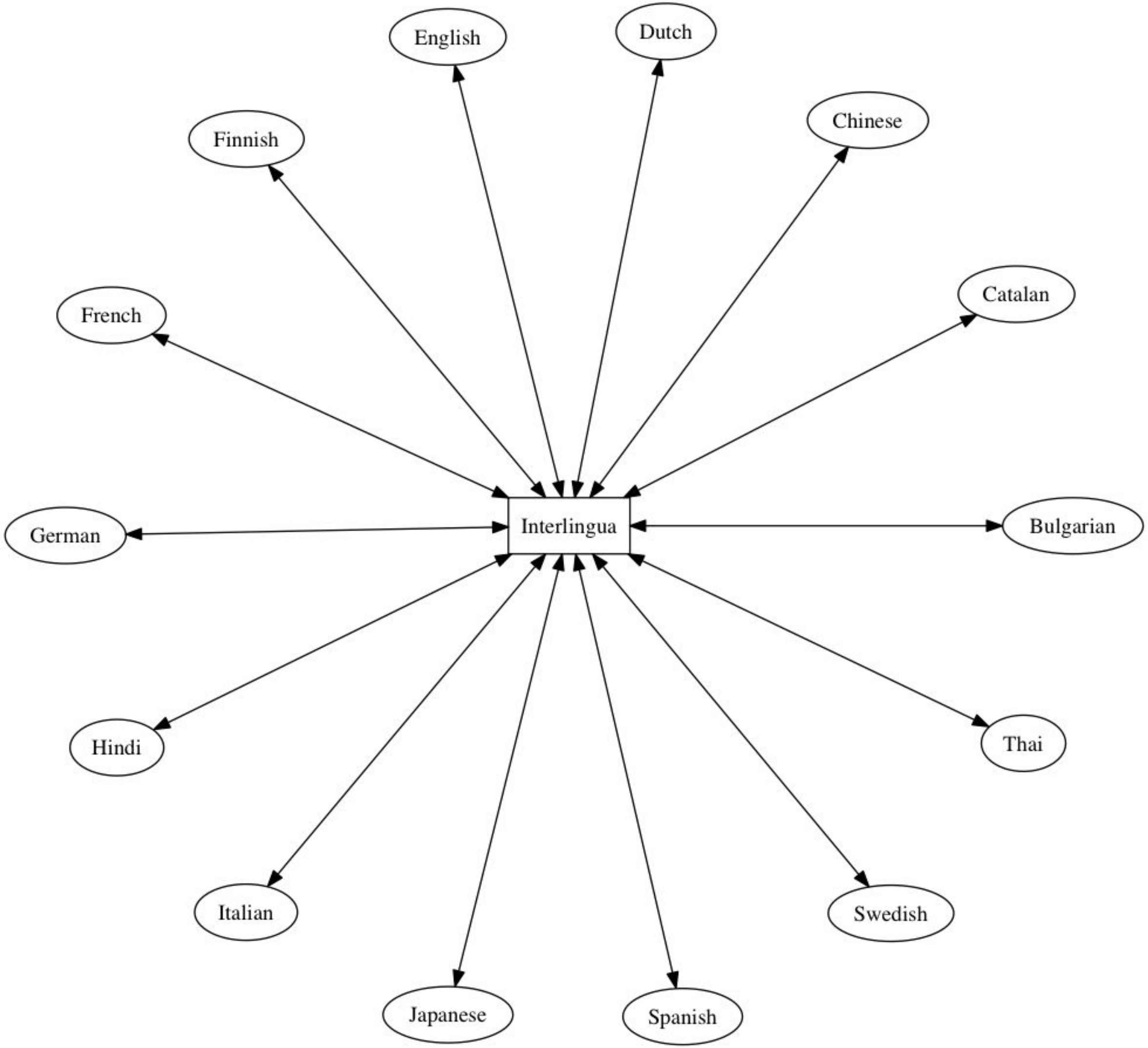
LF = Logical Framework = framework for defining logics

GF = LF + linearization = framework for defining grammars

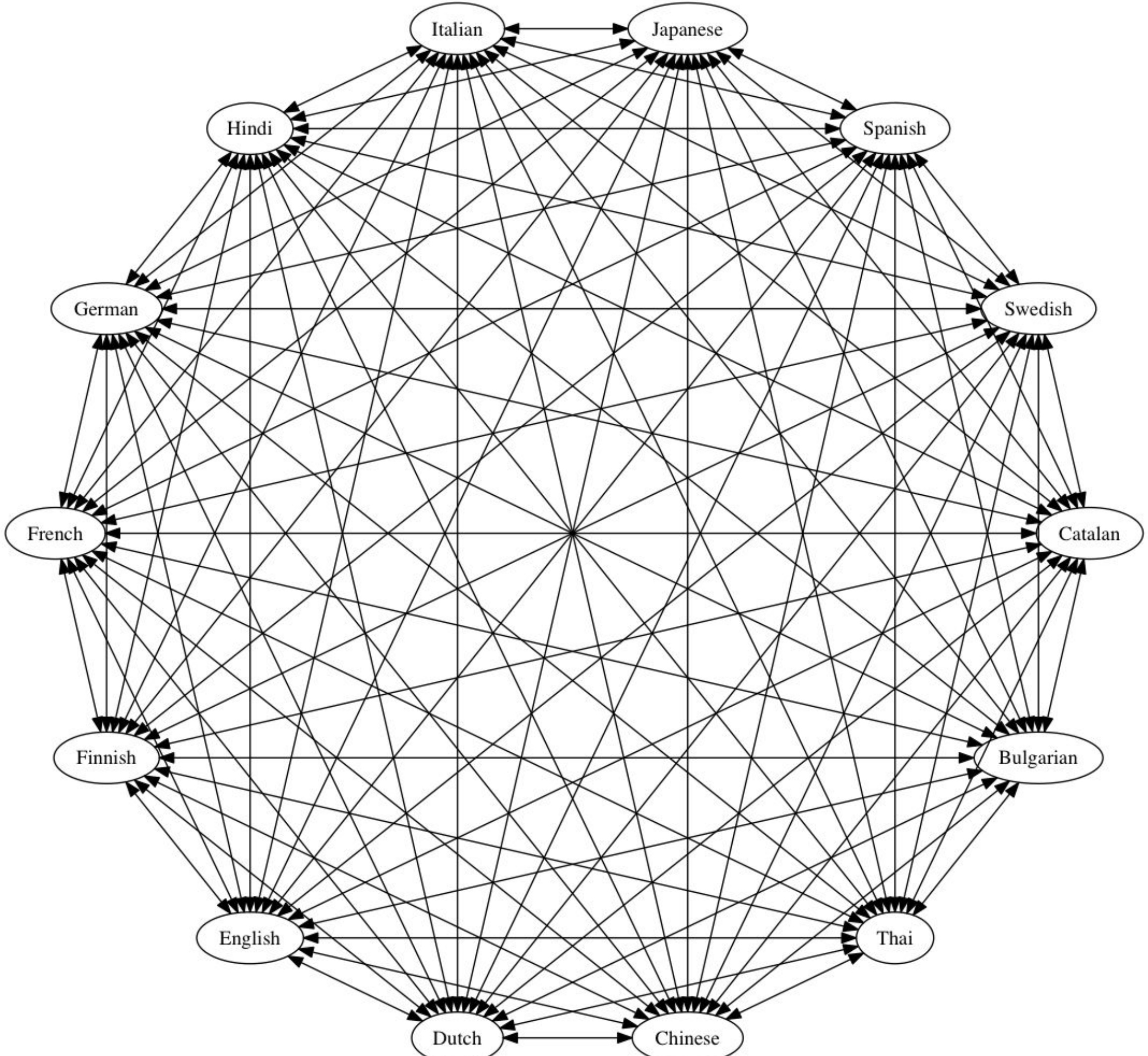
interlingua = abstract syntax = type theoretical logic



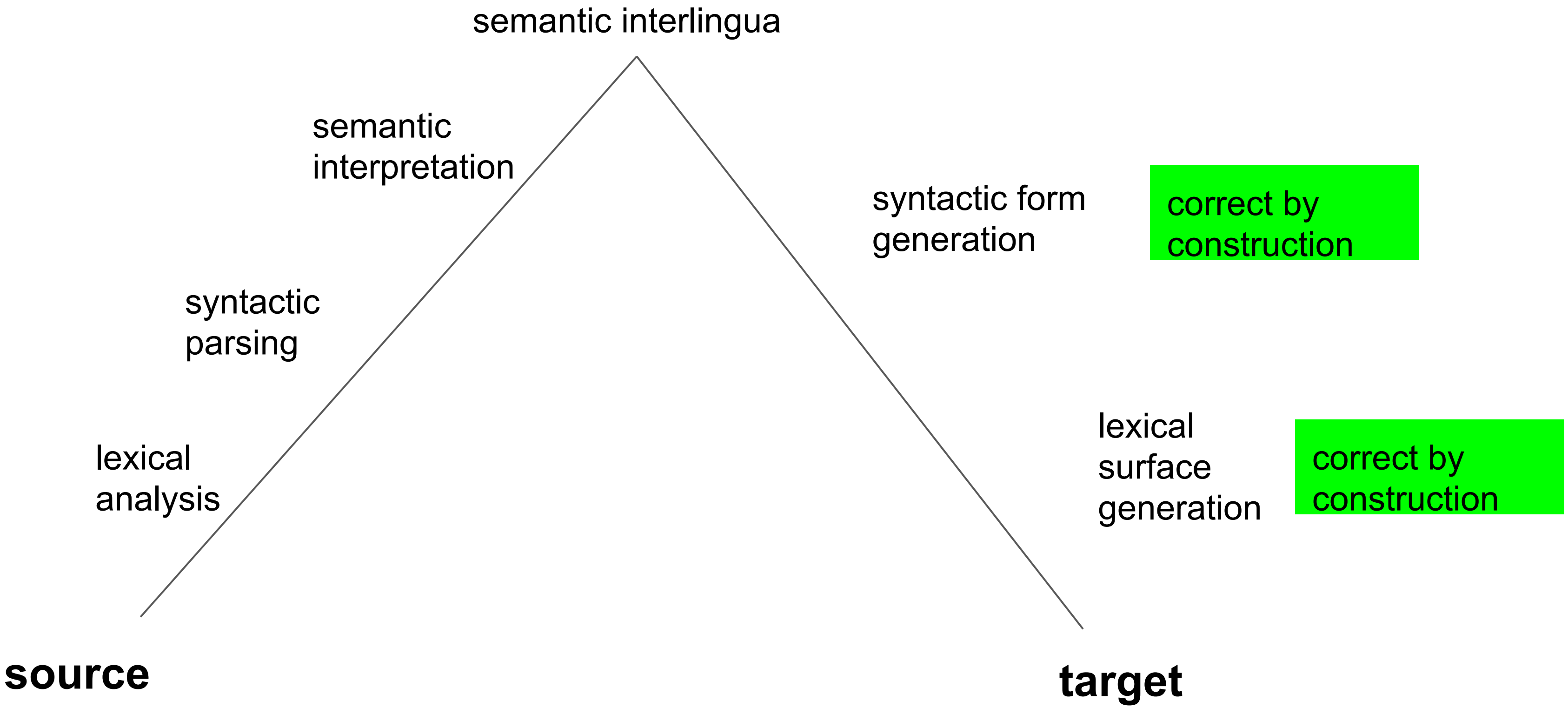
AR, *Grammatical Framework*, CSLI, 2011



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

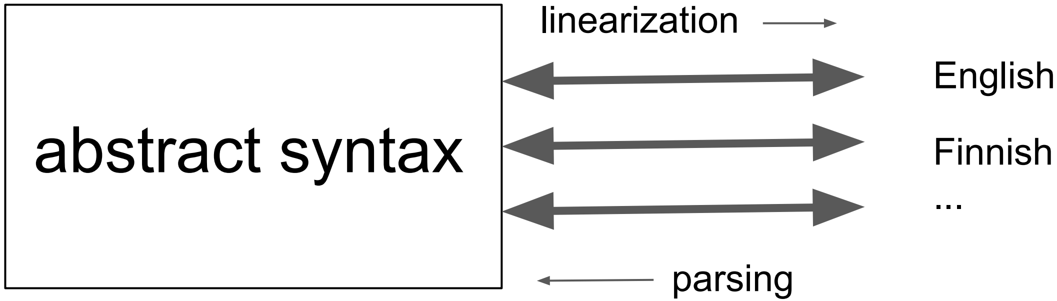

Italian
Japanese
Hindi
Spanish
German
Swedish
French
Catalan
Finnish
Bulgarian
English
Thai
Dutch
Chinese

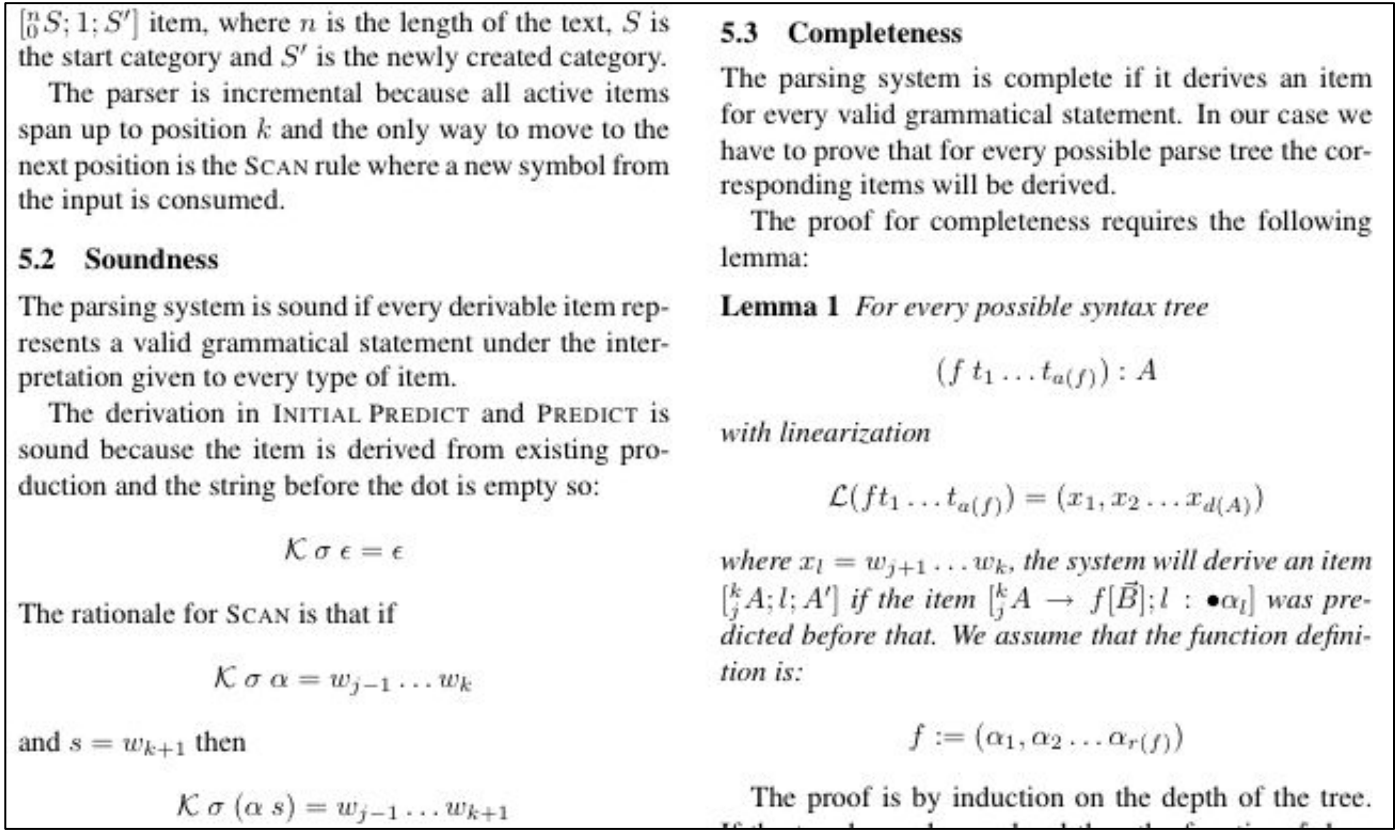
The Vauquois triangle answer



Parsing in GF

Reverse of linearization





$[{}^n_0 S; 1; S']$ item, where n is the length of the text, S is the start category and S' is the newly created category.

The parser is incremental because all active items span up to position k and the only way to move to the next position is the SCAN rule where a new symbol from the input is consumed.

5.2 Soundness

The parsing system is sound if every derivable item represents a valid grammatical statement under the interpretation given to every type of item.

The derivation in INITIAL PREDICT and PREDICT is sound because the item is derived from existing production and the string before the dot is empty so:

$$\mathcal{K}\ \sigma\ \epsilon = \epsilon$$

The rationale for SCAN is that if

$$\mathcal{K}\ \sigma\ \alpha = w_{j-1} \ldots w_k$$

and $s = w_{k+1}$ then

$$\mathcal{K}\ \sigma\ (\alpha\ s) = w_{j-1} \ldots w_{k+1}$$

5.3 Completeness

The parsing system is complete if it derives an item for every valid grammatical statement. In our case we have to prove that for every possible parse tree the corresponding items will be derived.

The proof for completeness requires the following lemma:

Lemma 1 *For every possible syntax tree*

$$(f\ t_1 \ldots t_{a(f)}) : A$$

with linearization

$$\mathcal{L}(f t_1 \ldots t_{a(f)}) = (x_1, x_2 \ldots x_{d(A)})$$

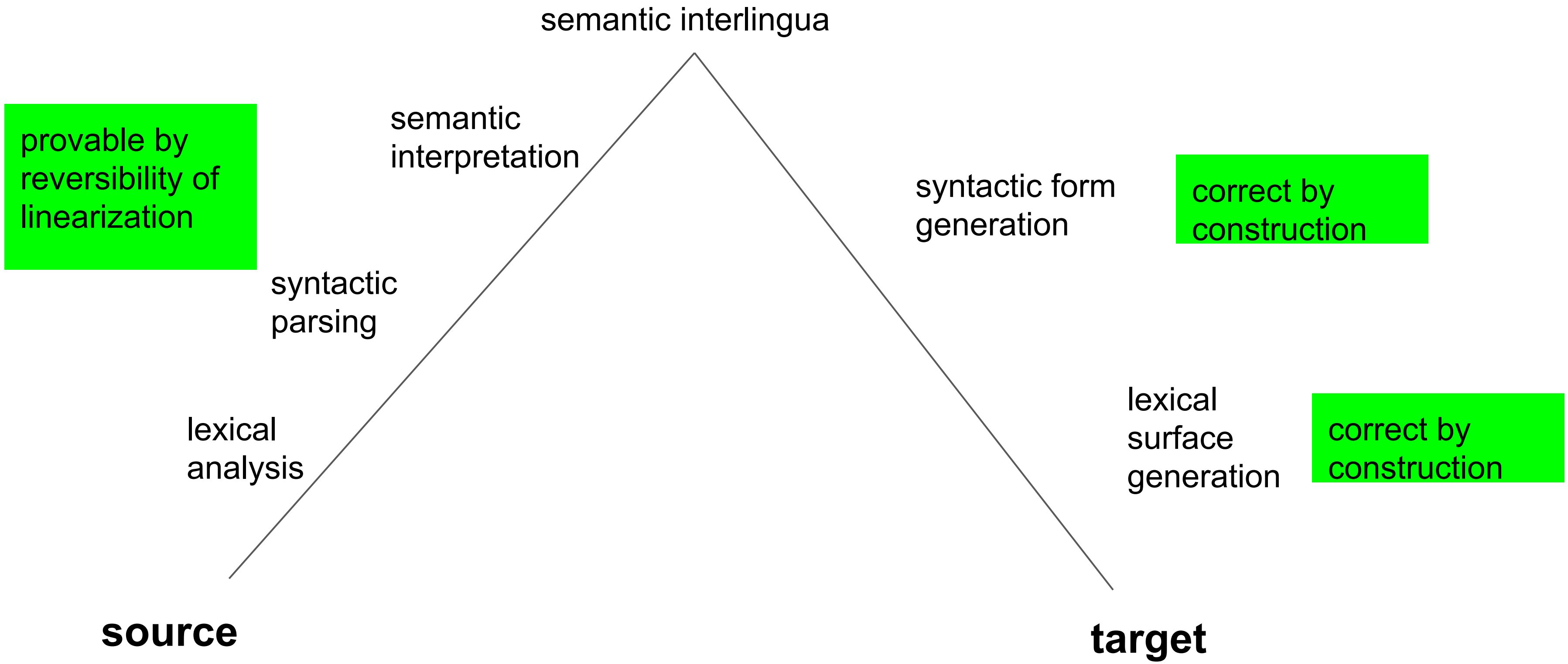
where $x_l = w_{j+1} \ldots w_k$*, the system will derive an item* $[{}^k_j A; l; A']$ *if the item* $[{}^k_j A \to f[\vec{B}]; l : \bullet\alpha_l]$ *was predicted before that. We assume that the function definition is:*

$$f := (\alpha_1, \alpha_2 \ldots \alpha_{r(f)})$$

The proof is by induction on the depth of the tree.

K. Angelov, Incremental Parsing with Parallel Multiple Context-Free Grammars, EACL 2009

The Vauquois triangle answer, variant 1



semantic interlingua

provable by reversibility of linearization

semantic interpretation

syntactic parsing

lexical analysis

syntactic form generation

correct by construction

lexical surface generation

correct by construction

source

target

Problems with variant 1

Ambiguity

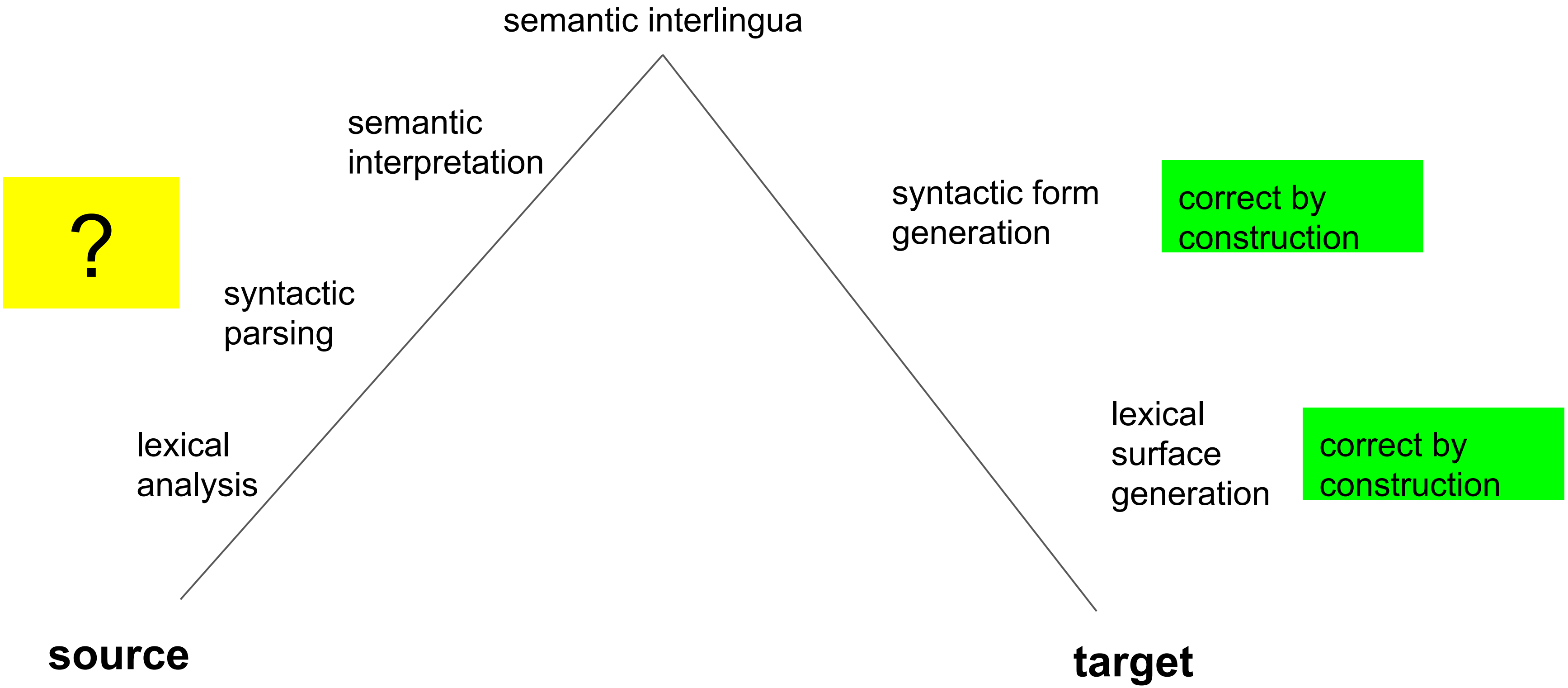
Linearization is many-to-1

→ Parsing is 1-to-many

Incompleteness

The grammar doesn’t cover all input

The Vauquois triangle answer, variant 2



semantic interlingua

semantic interpretation

?

syntactic parsing

lexical analysis

syntactic form generation

correct by construction

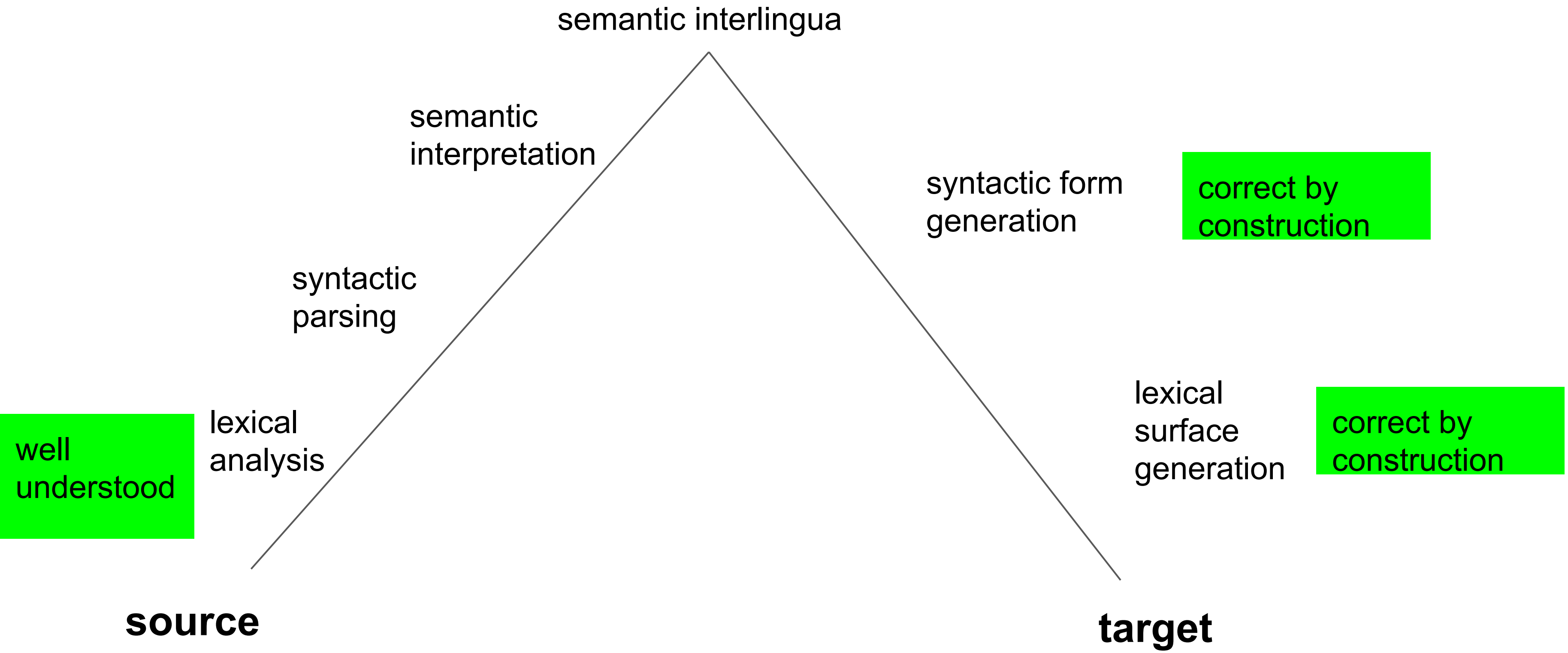
lexical surface generation

correct by construction

source

target

The Vauquois triangle answer, variant 2



semantic interlingua

semantic interpretation

syntactic parsing

lexical analysis

well understood

syntactic form generation

correct by construction

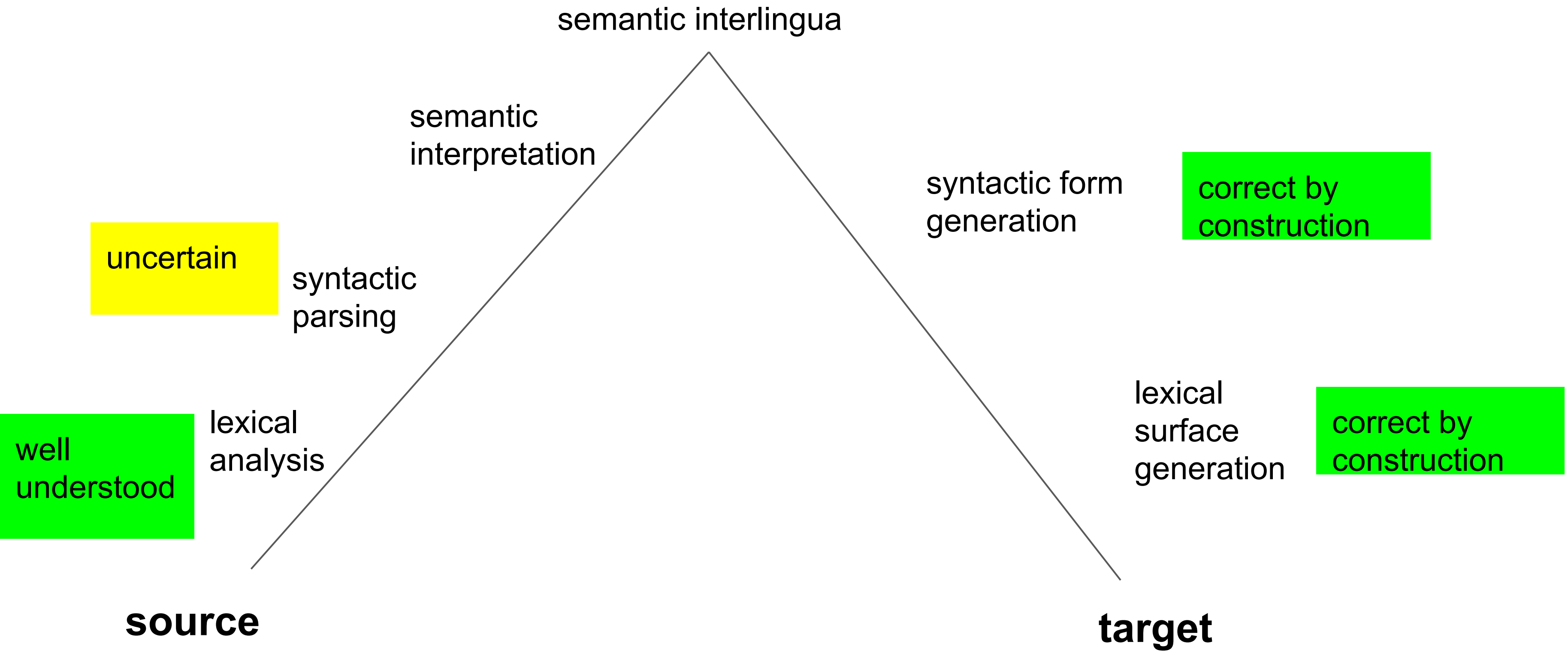
lexical surface generation

correct by construction

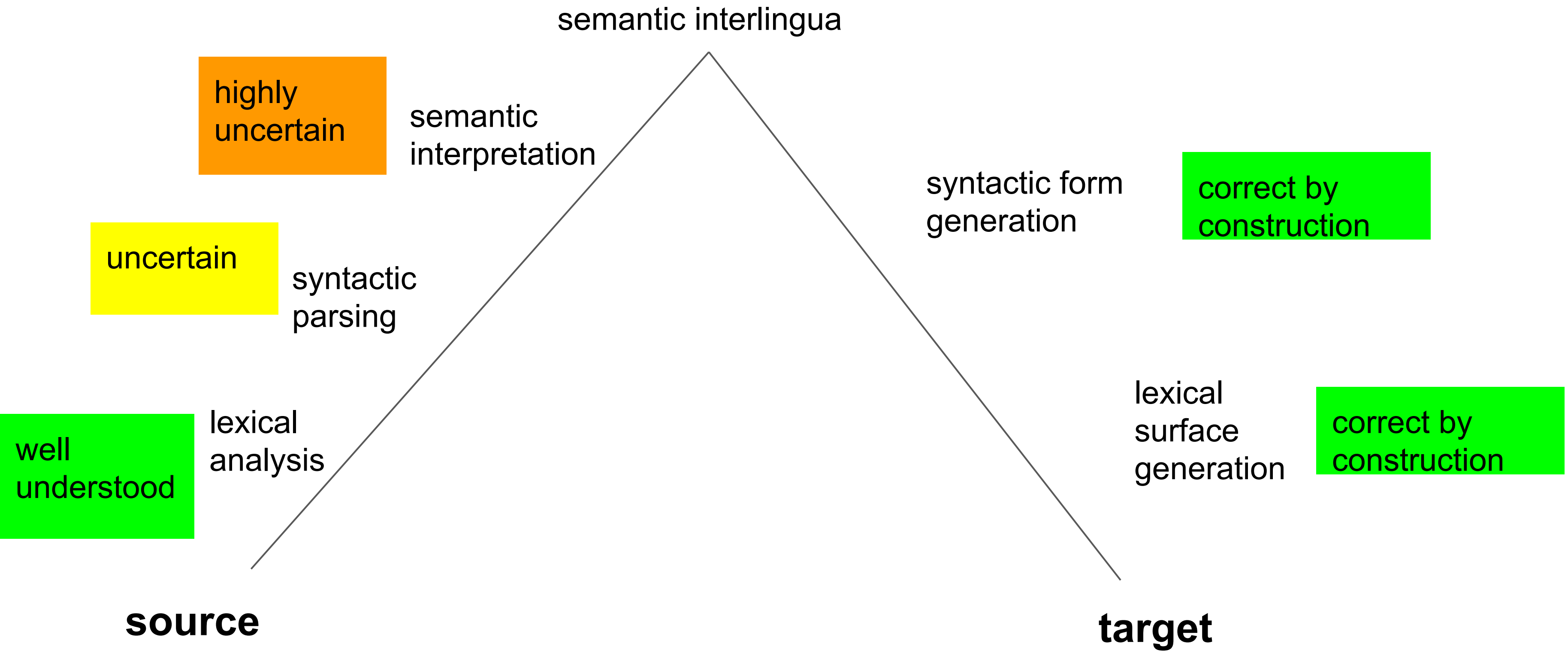
source

target

The Vauquois triangle answer, variant 2



The Vauquois triangle answer, variant 2



semantic interlingua

highly uncertain

semantic interpretation

uncertain

syntactic parsing

well understood

lexical analysis

syntactic form generation

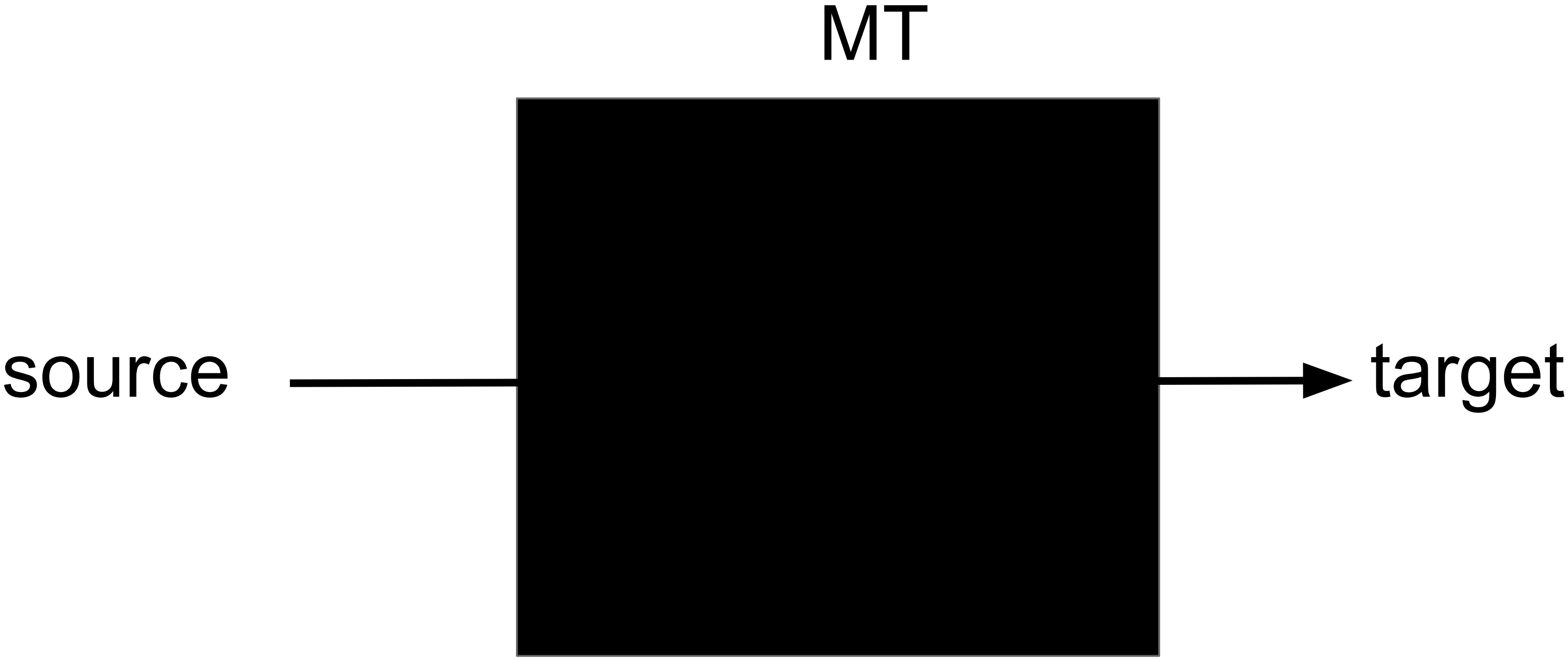
correct by construction

lexical surface generation

correct by construction

source

target

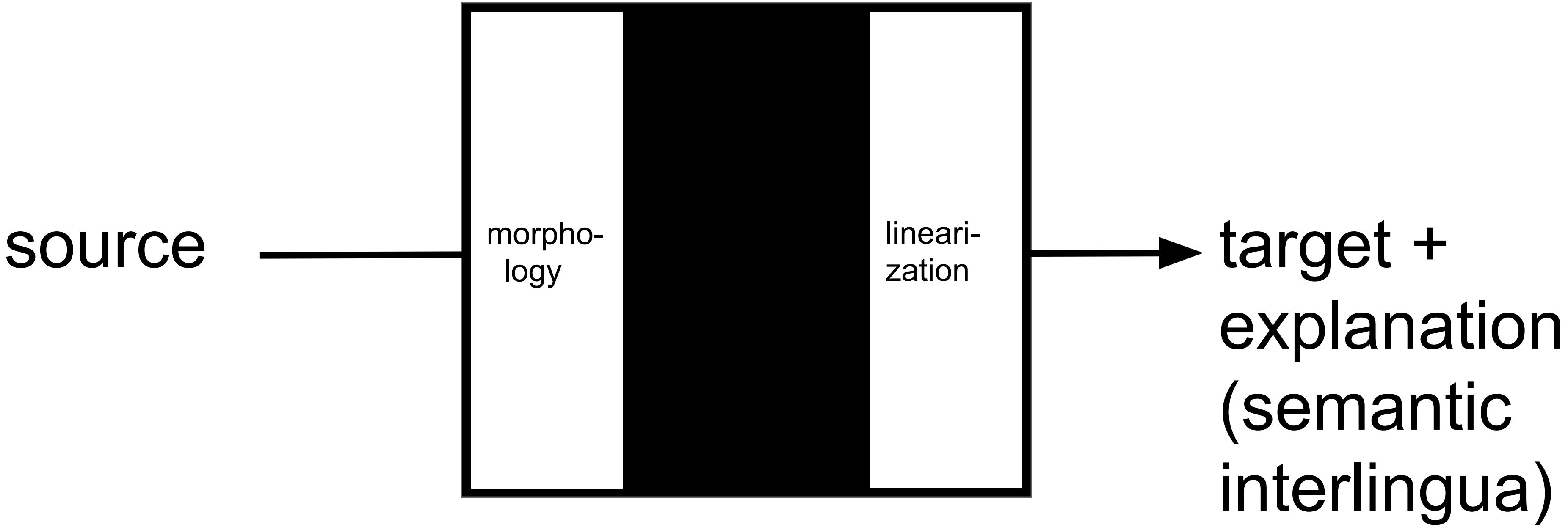


MT

source

target

XMT



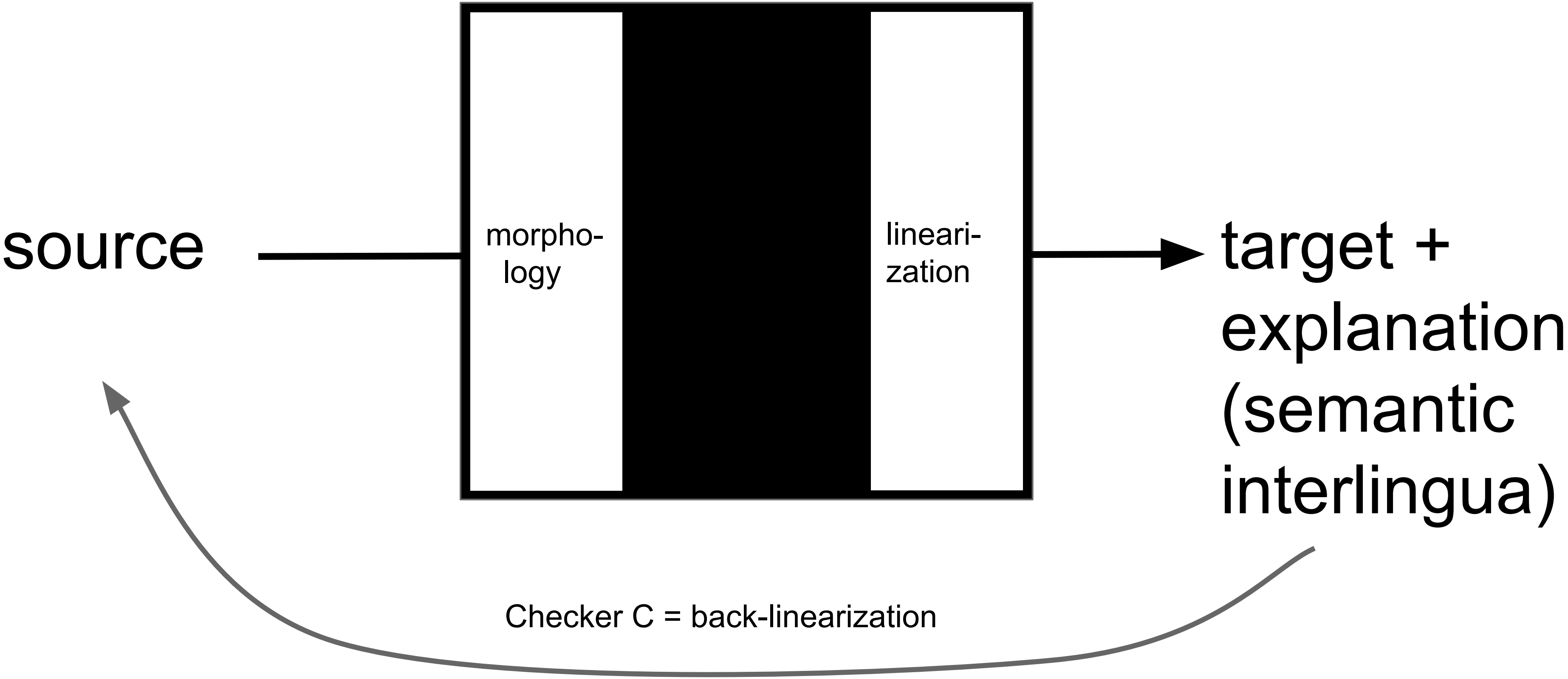
source

morphology

linearization

target + explanation (semantic interlingua)

XMT



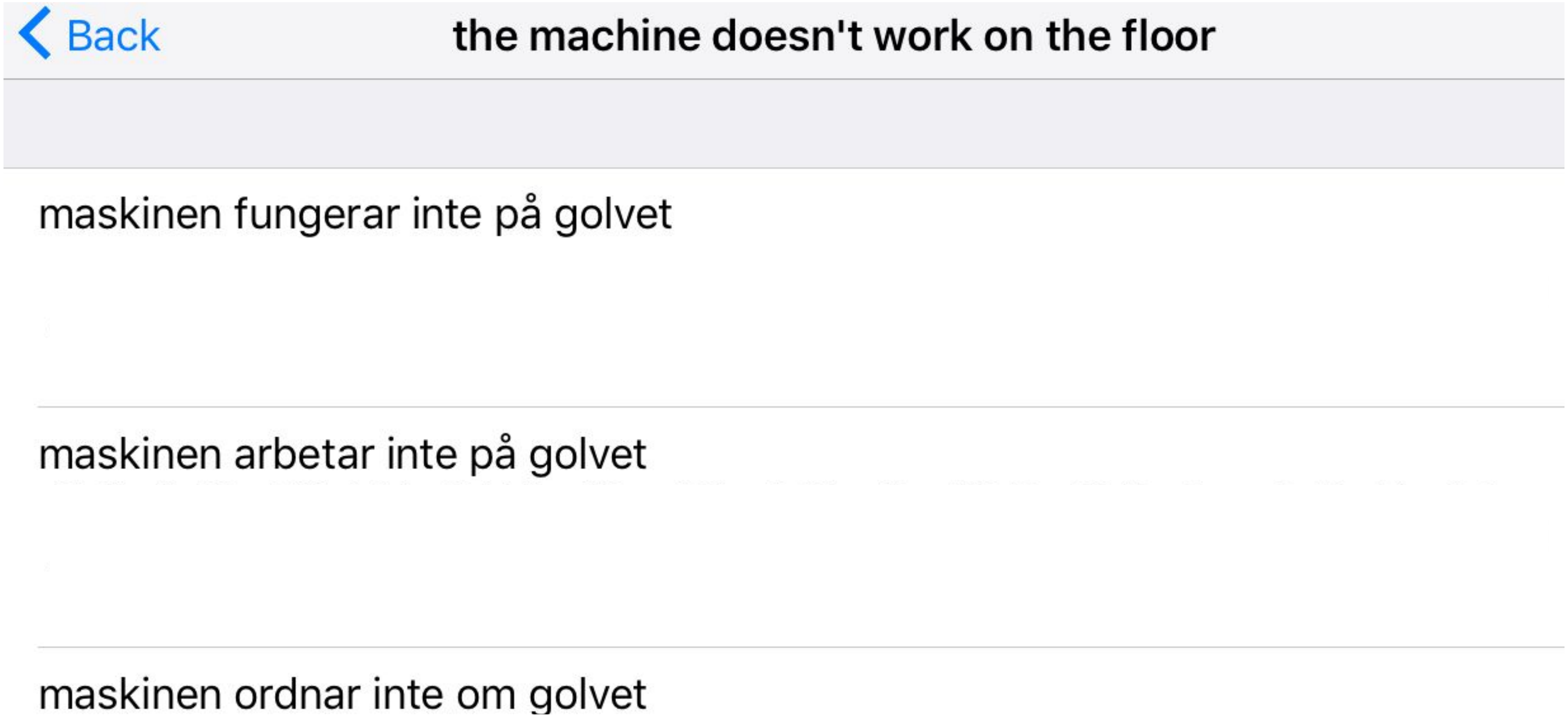
source

morphology

linearization

target + explanation (semantic interlingua)

Checker C = back-linearization



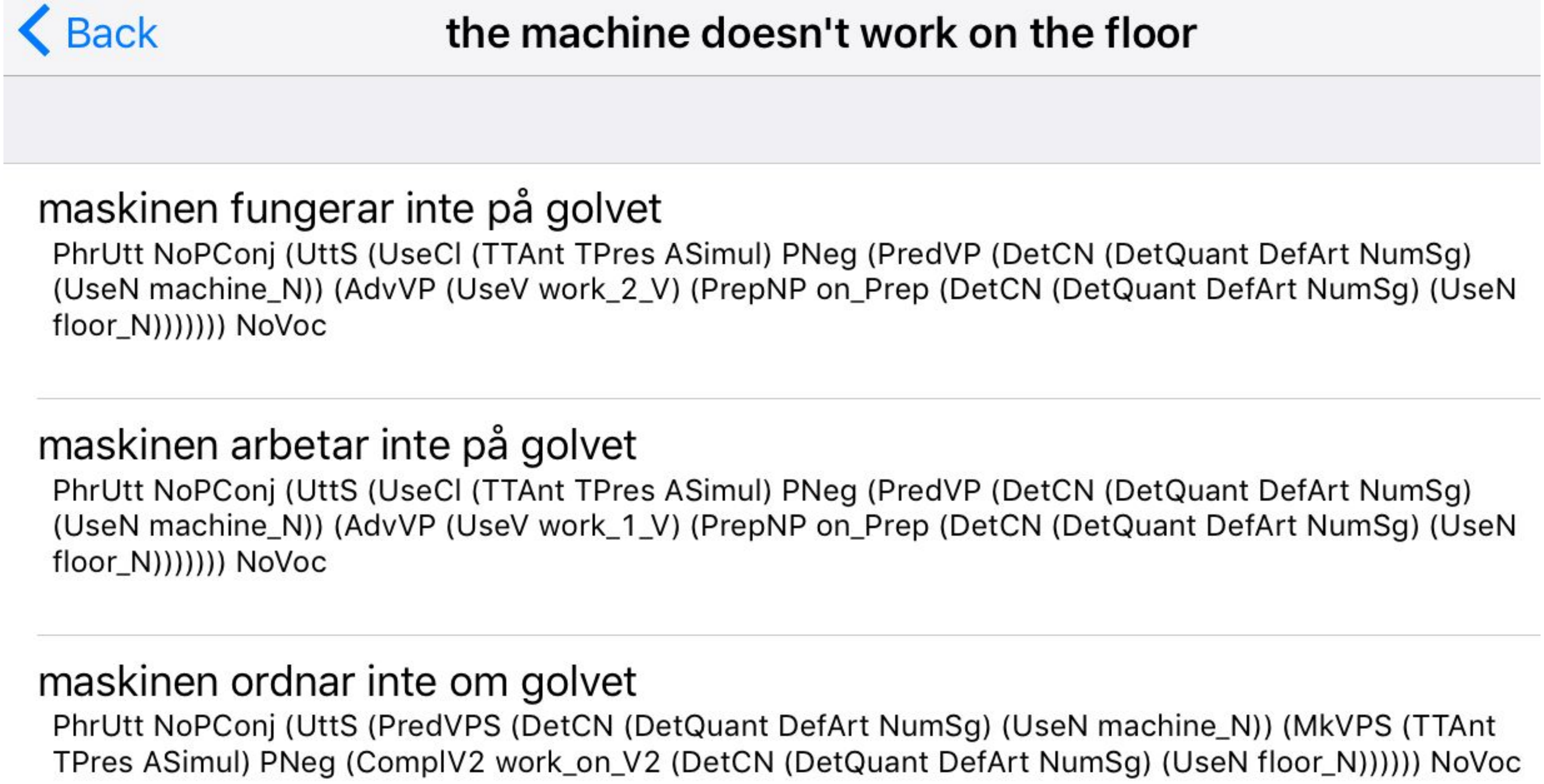
the machine doesn't work on the floor

maskinen fungerar inte på golvet

maskinen arbetar inte på golvet

maskinen ordnar inte om golvet

t
'oc



the machine doesn't work on the floor

maskinen fungerar inte på golvet

PhrUtt NoPConj (UttS (UseCl (TTAnt TPres ASimul) PNeg (PredVP (DetCN (DetQuant DefArt NumSg) (UseN machine_N)) (AdvVP (UseV work_2_V) (PrepNP on_Prep (DetCN (DetQuant DefArt NumSg) (UseN floor_N))))))) NoVoc

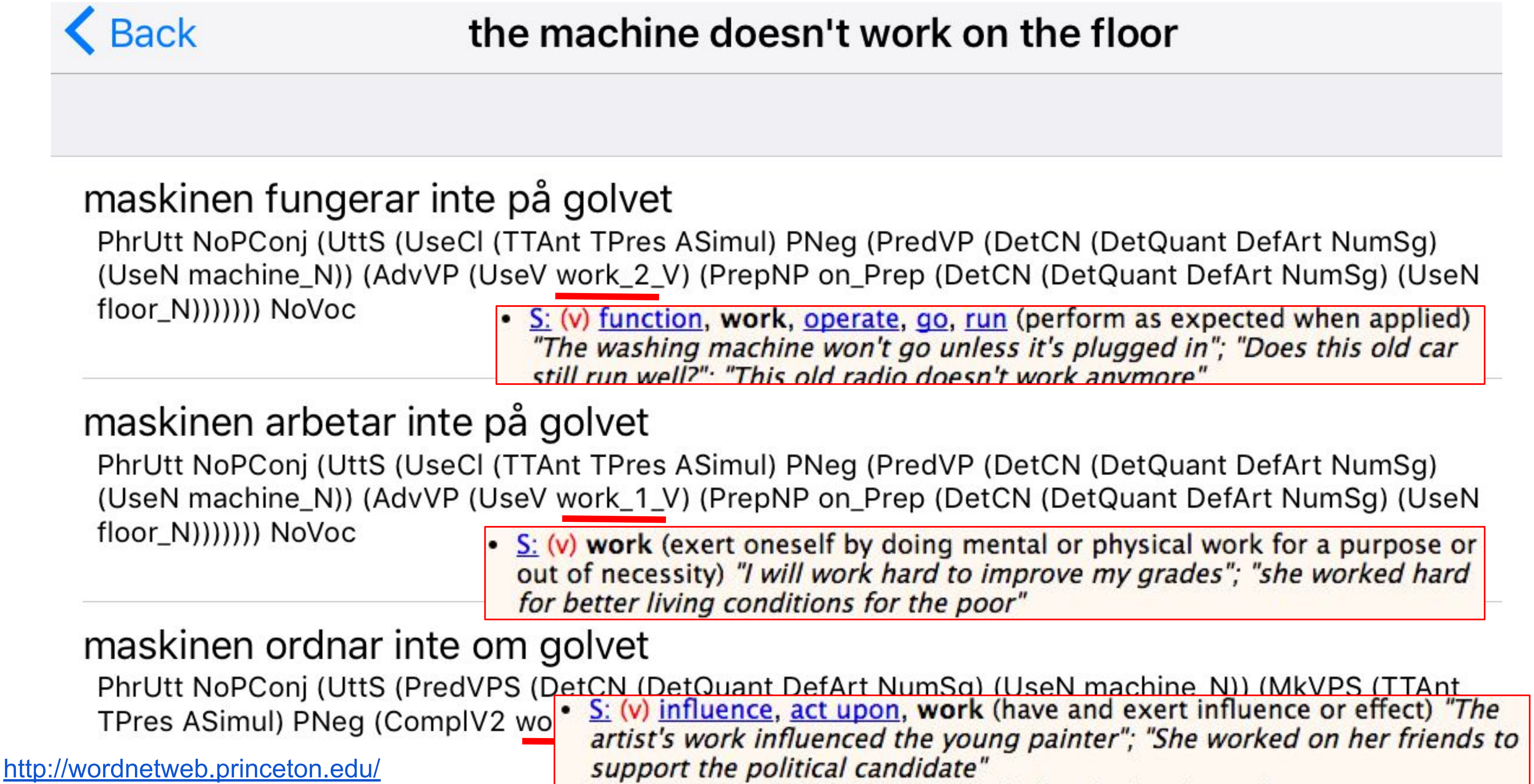
maskinen arbetar inte på golvet

PhrUtt NoPConj (UttS (UseCl (TTAnt TPres ASimul) PNeg (PredVP (DetCN (DetQuant DefArt NumSg) (UseN machine_N)) (AdvVP (UseV work_1_V) (PrepNP on_Prep (DetCN (DetQuant DefArt NumSg) (UseN floor_N))))))) NoVoc

maskinen ordnar inte om golvet

PhrUtt NoPConj (UttS (PredVPS (DetCN (DetQuant DefArt NumSg) (UseN machine_N)) (MkVPS (TTAnt TPres ASimul) PNeg (ComplV2 work_on_V2 (DetCN (DetQuant DefArt NumSg) (UseN floor_N)))))) NoVoc

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



the machine doesn't work on the floor

maskinen fungerar inte på golvet

PhrUtt NoPConj (UttS (UseCl (TTAnt TPres ASimul) PNeg (PredVP (DetCN (DetQuant DefArt NumSg) (UseN machine_N)) (AdvVP (UseV work_2_V) (PrepNP on_Prep (DetCN (DetQuant DefArt NumSg) (UseN floor_N))))))) NoVoc

- S: (v) function, **work**, operate, go, run (perform as expected when applied) *"The washing machine won't go unless it's plugged in"; "Does this old car still run well?"; "This old radio doesn't work anymore"*

maskinen arbetar inte på golvet

PhrUtt NoPConj (UttS (UseCl (TTAnt TPres ASimul) PNeg (PredVP (DetCN (DetQuant DefArt NumSg) (UseN machine_N)) (AdvVP (UseV work_1_V) (PrepNP on_Prep (DetCN (DetQuant DefArt NumSg) (UseN floor_N))))))) NoVoc

- S: (v) **work** (exert oneself by doing mental or physical work for a purpose or out of necessity) *"I will work hard to improve my grades"; "she worked hard for better living conditions for the poor"*

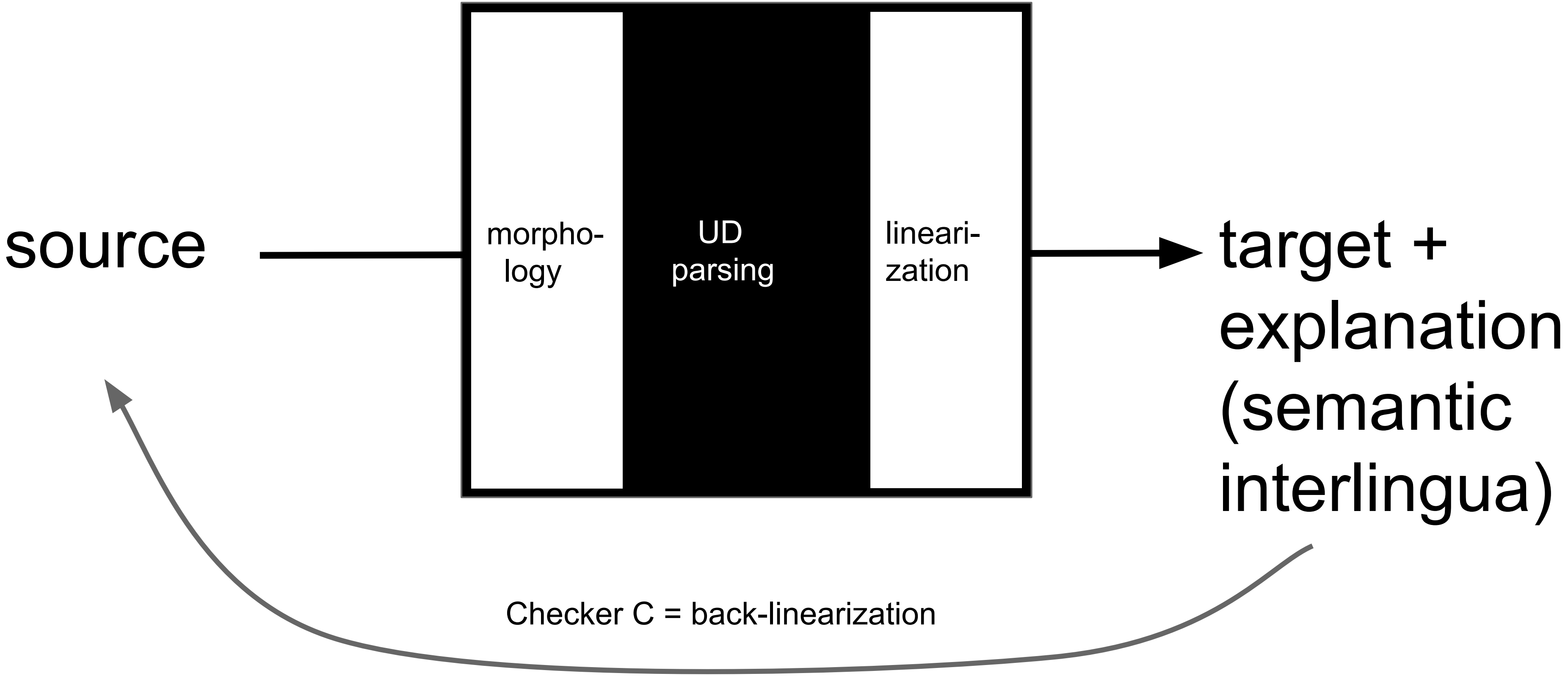
maskinen ordnar inte om golvet

PhrUtt NoPConj (UttS (PredVPS (DetCN (DetQuant DefArt NumSg) (UseN machine_N)) (MkVPS (TTAnt TPres ASimul) PNeg (ComplV2 wo

- S: (v) influence, act upon, **work** (have and exert influence or effect) *"The artist's work influenced the young painter"; "She worked on her friends to support the political candidate"*

http://wordnetweb.princeton.edu/

XMT



source

morphology

UD parsing

linearization

target + explanation (semantic interlingua)

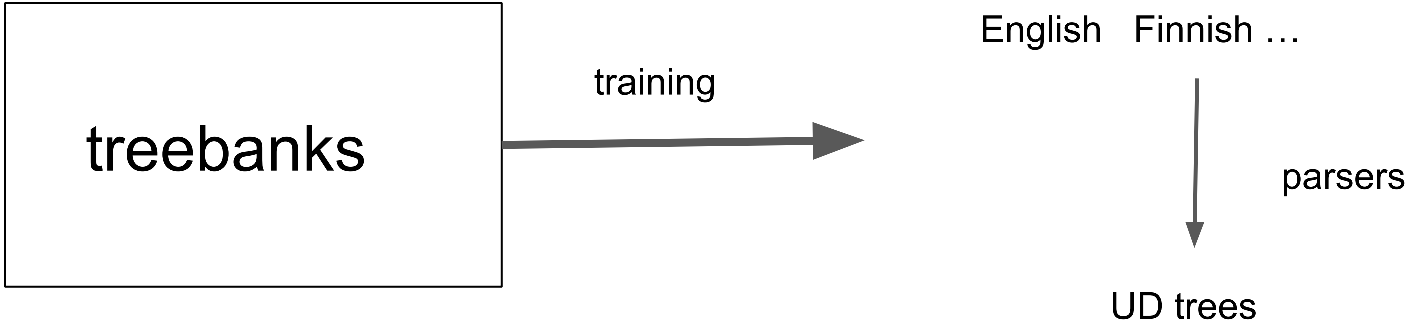
Checker C = back-linearization

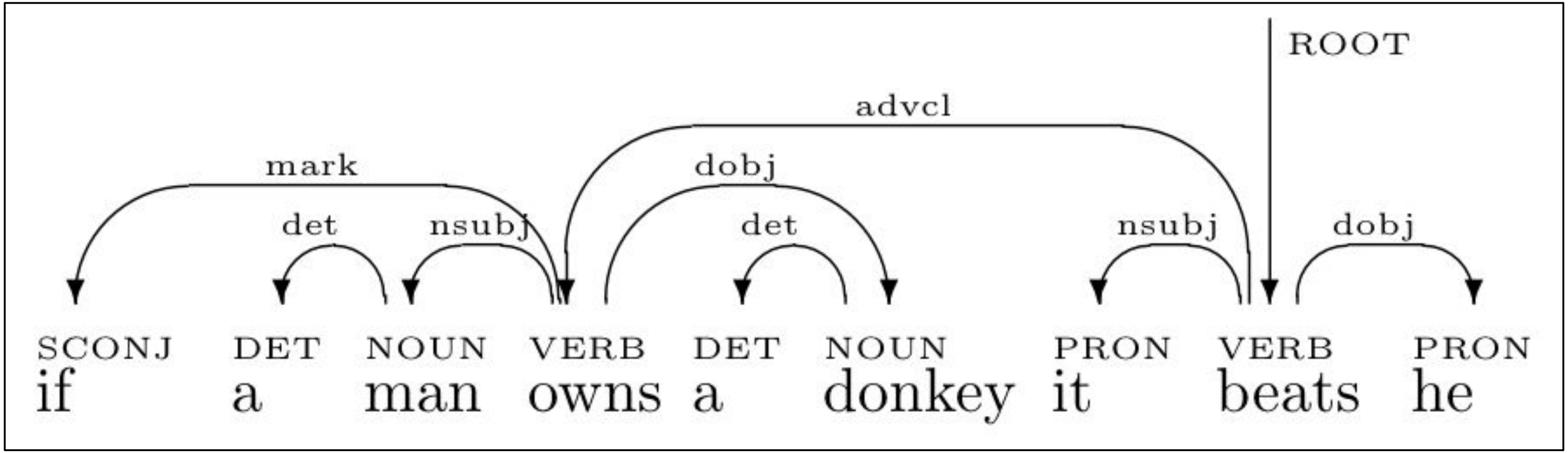
UD = Universal Dependencies

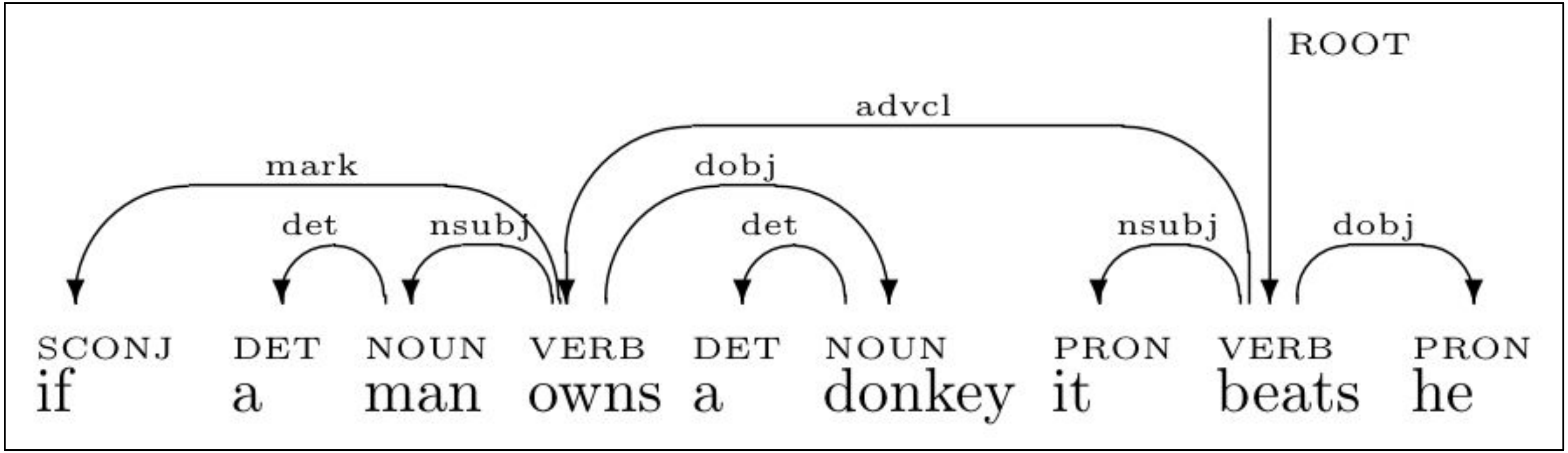
Dependency tree: labelled arcs between words

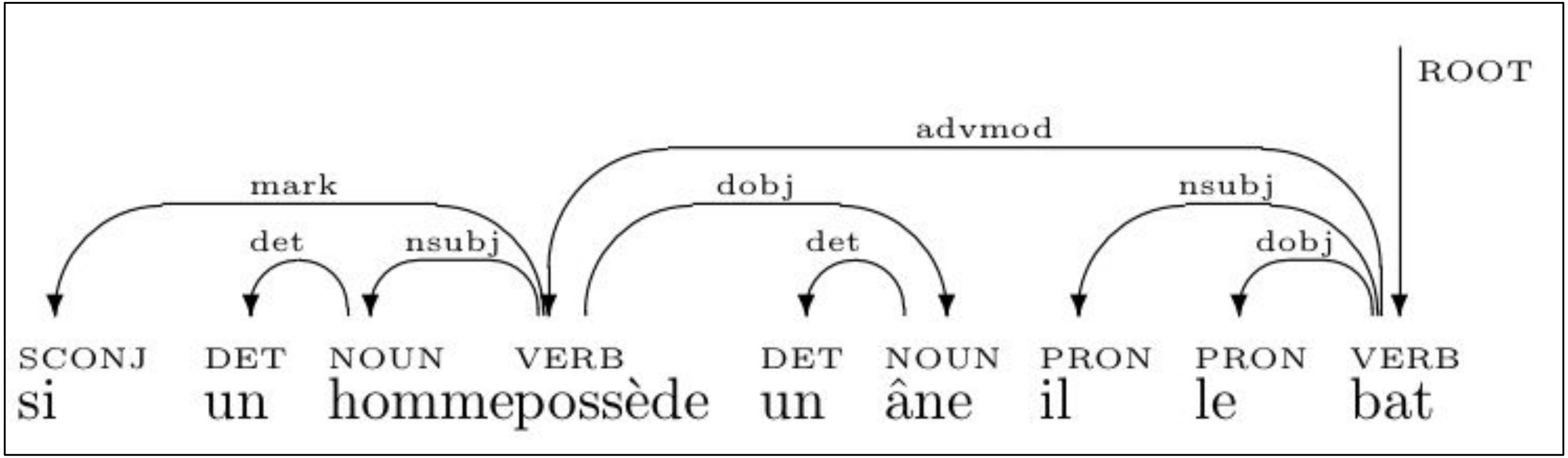
Universal: same labels in different languages

Parsing: machine-learned from treebanks




ROOT
advcl
mark
dobj
det
nsubj
det
nsubj
dobj
SCONJ
DET
NOUN
VERB
DET
NOUN
PRON
VERB
PRON
if
a
man
owns
a
donkey
it
beats
he


ROOT
advcl
mark
dobj
det
nsubj
det
nsubj
dobj
SCONJ DET NOUN VERB DET NOUN PRON VERB PRON
if a man owns a donkey it beats he


ROOT
advmod
mark
dobj
nsubj
det
nsubj
det
dobj
SCONJ DET NOUN VERB DET NOUN PRON PRON VERB
si un hommepossède un âne il le bat

Languages in UD and GF

Basque
Belarusian Buryat
Coptic Croatian
Czech Galician
Hungarian
Indonesian Irish
Kazakh Korean
Kurmanji Lithuanian
NorthSami
OldChurchSlavonic
Portuguese Sanskrit
Slovak Tamil
Ukranian
UpperSorbian Uyghur
Vietnamese

Arabic
Bulgarian
Catalan Chinese
Danish Dutch English
Estonian Finnish
French German Gothic
Greek(Ancient,Modern)
Hebrew Hindi Italian
Japanese
Latin Latvian Maltese
Norwegian(bokmål,nynorsk)
Persian Polish
Romanian Russian
Slovenian Spanish Swedish
Thai Turkish Urdu

Afrikaans
Amharic
Icelandic
Mongolian
Nepali
Punjabi
Sindhi
Swahili

http://universaldependencies.org/

http://www.grammaticalframework.org/lib/doc/status.html

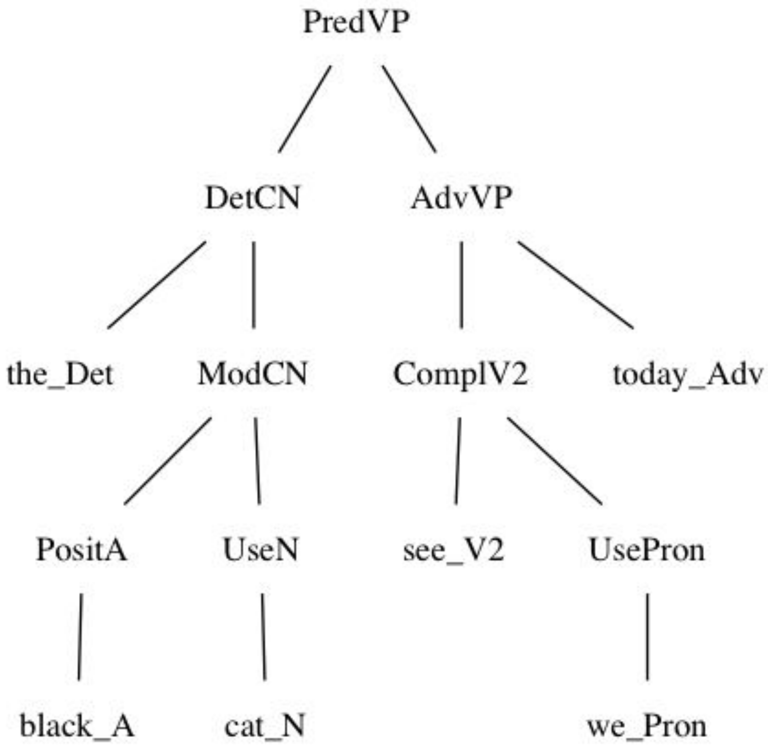
abstract syntax

```
PredVP  : NP  -> VP  -> Cl

ComplV2 : V2  -> NP  -> VP

AdvVP   : VP  -> Adv -> VP

DetCN   : Det -> CN  -> NP

ModCN   : AP  -> CN  -> CN

UseN    : N    -> CN

UsePron : Pron -> NP

PositA  : A    -> AP
```

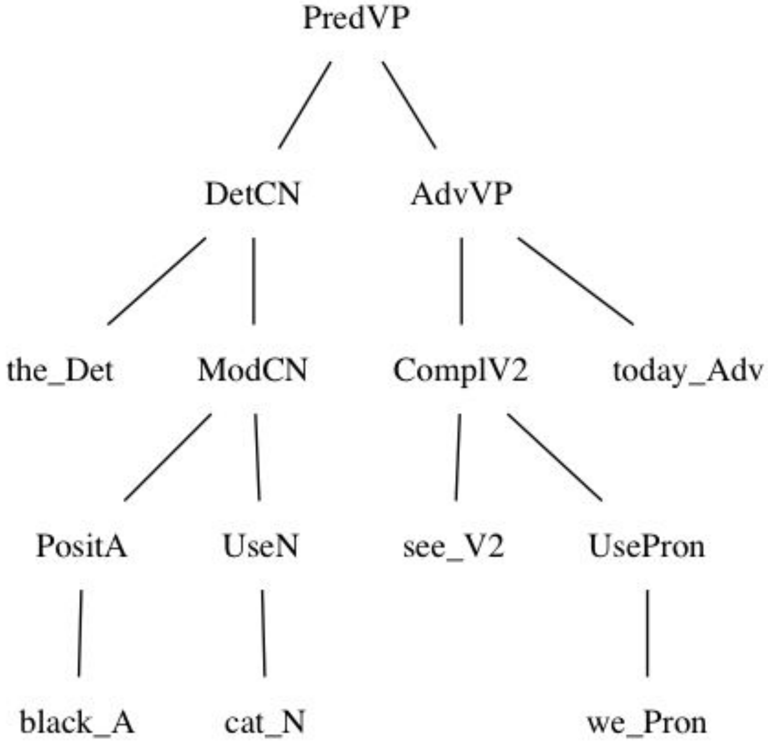
abstract syntax

```
PredVP  : NP  -> VP  -> Cl

ComplV2 : V2  -> NP  -> VP

AdvVP   : VP  -> Adv -> VP

DetCN   : Det -> CN  -> NP

ModCN   : AP  -> CN  -> CN

UseN    : N    -> CN

UsePron : Pron -> NP

PositA  : A    -> AP
```

the black cat sees us today

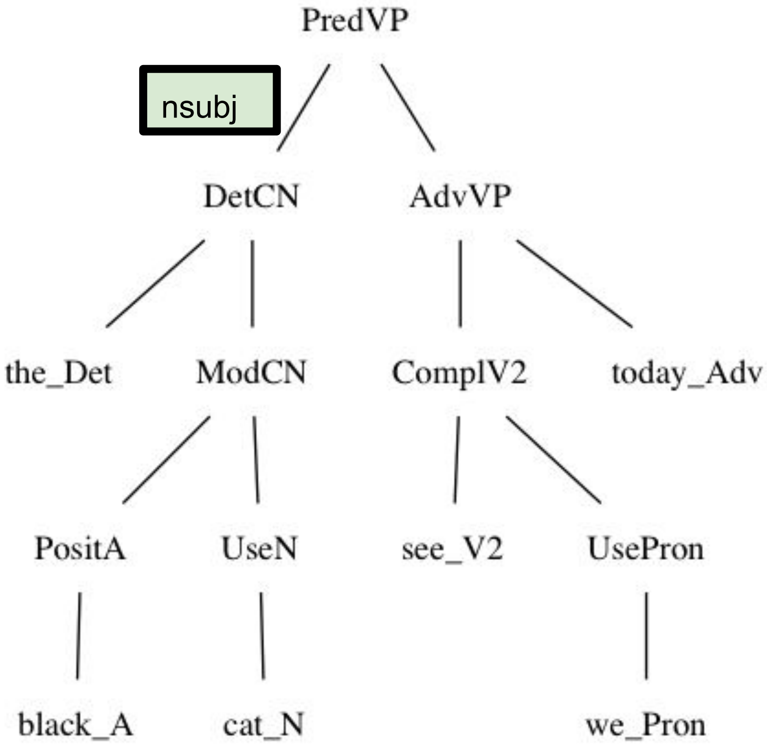


abstract syntax	**dependency configuration**
`PredVP  : NP   -> VP   -> Cl`	nsubj head
`ComplV2 : V2   -> NP   -> VP`	head dobj
`AdvVP   : VP   -> Adv  -> VP`	head advmod
`DetCN   : Det  -> CN   -> NP`	det head
`ModCN   : AP   -> CN   -> CN`	amod head
`UseN    : N     -> CN`	head
`UsePron : Pron  -> NP`	head
`PositA  : A     -> AP`	head

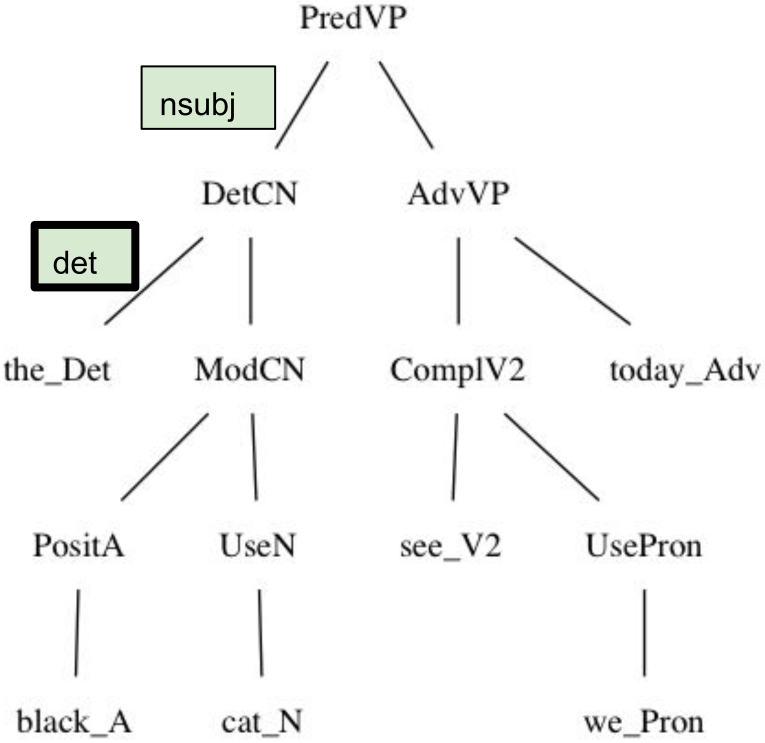


Kolachina & Ranta, From Abstract Syntax to Universal Dependencies, LiLT 2016.

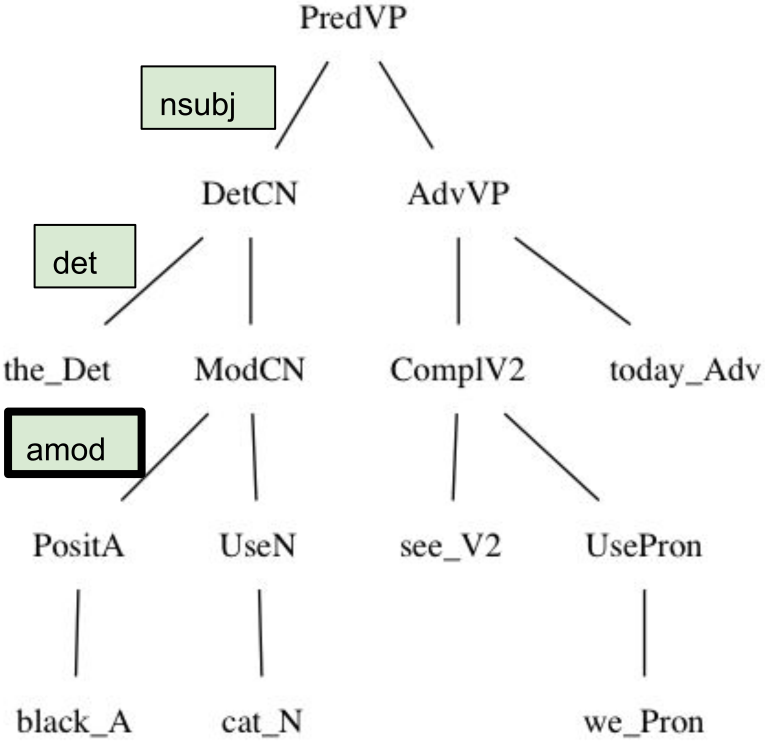
abstract syntax	dependency configuration
`PredVP  : NP   -> VP   -> Cl`	nsubj head
`ComplV2 : V2   -> NP   -> VP`	head dobj
`AdvVP   : VP   -> Adv  -> VP`	head advmod
`DetCN   : Det  -> CN   -> NP`	det head
`ModCN   : AP   -> CN   -> CN`	amod head
`UseN    : N     -> CN`	head
`UsePron : Pron  -> NP`	head
`PositA  : A     -> AP`	head



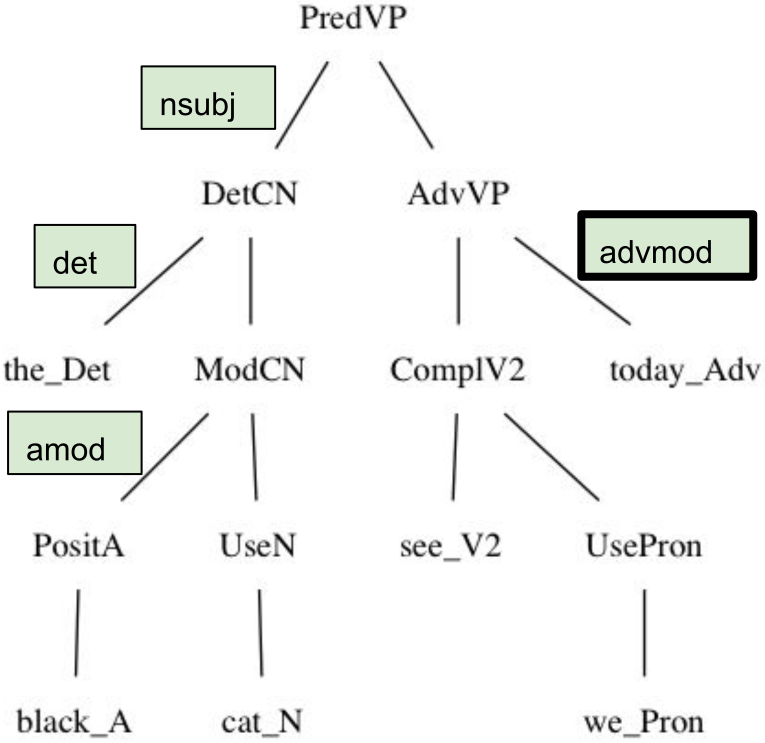
abstract syntax	dependency configuration
`PredVP  : NP   -> VP   -> Cl`	`nsubj  head`
`ComplV2 : V2   -> NP   -> VP`	`head   dobj`
`AdvVP   : VP   -> Adv  -> VP`	`head   advmod`
`DetCN   : Det  -> CN   -> NP`	`det    head`
`ModCN   : AP   -> CN   -> CN`	`amod   head`
`UseN    : N     -> CN`	`head`
`UsePron : Pron  -> NP`	`head`
`PositA  : A     -> AP`	`head`



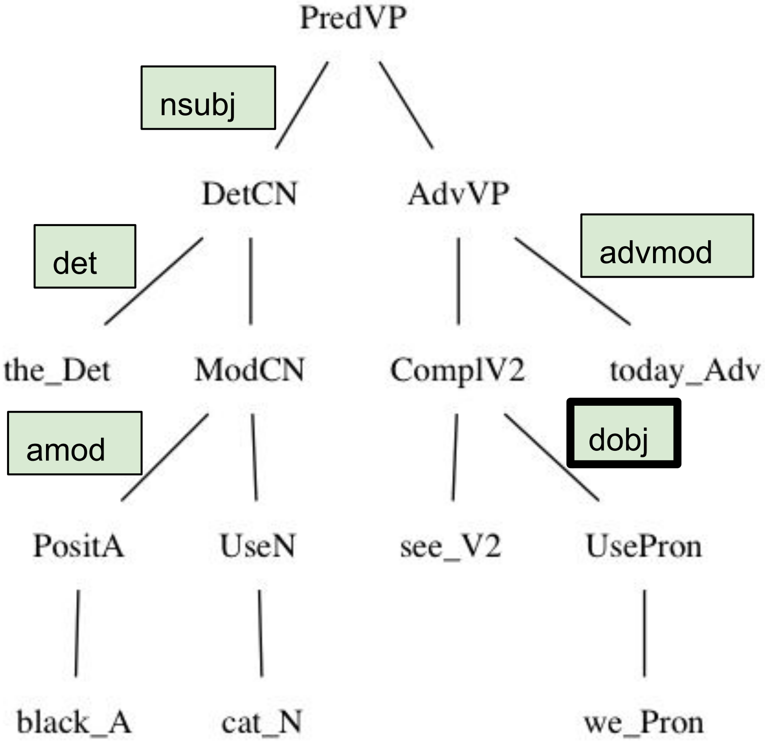
abstract syntax	dependency configuration
`PredVP  : NP   -> VP   -> Cl`	`nsubj  head`
`ComplV2 : V2   -> NP   -> VP`	`head   dobj`
`AdvVP   : VP   -> Adv  -> VP`	`head   advmod`
`DetCN   : Det  -> CN   -> NP`	`det    head`
`ModCN   : AP   -> CN   -> CN`	`amod   head`
`UseN    : N     -> CN`	`head`
`UsePron : Pron  -> NP`	`head`
`PositA  : A     -> AP`	`head`

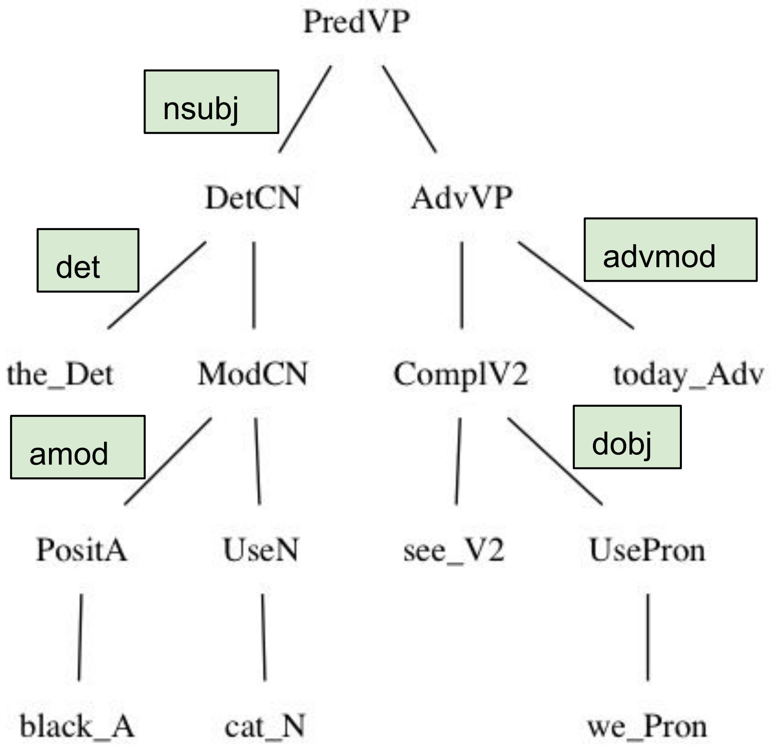


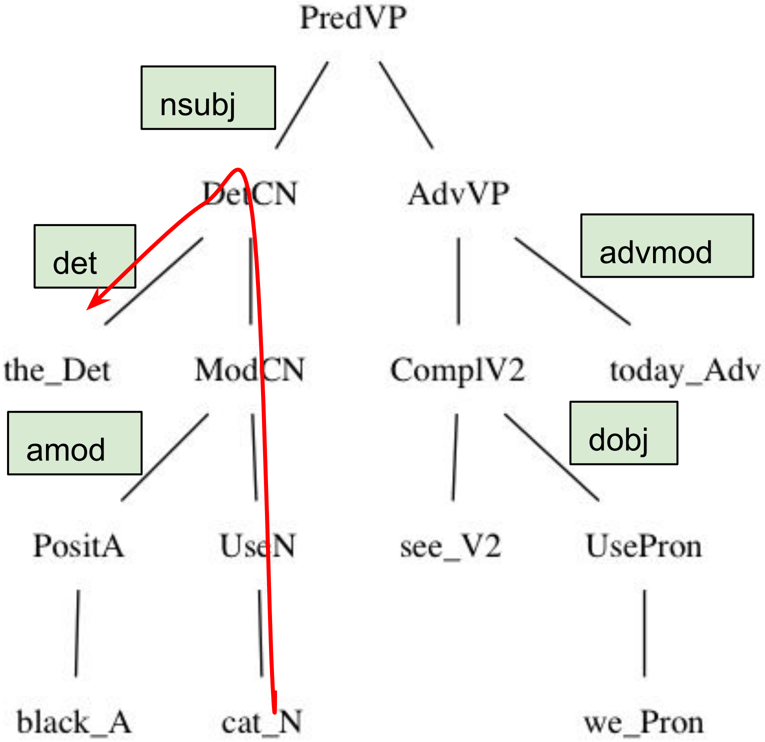
abstract syntax	dependency configuration
PredVP : NP -> VP -> Cl	nsubj head
ComplV2 : V2 -> NP -> VP	head dobj
AdvVP : VP -> Adv -> VP	head advmod
DetCN : Det -> CN -> NP	det head
ModCN : AP -> CN -> CN	amod head
UseN : N -> CN	head
UsePron : Pron -> NP	head
PositA : A -> AP	head

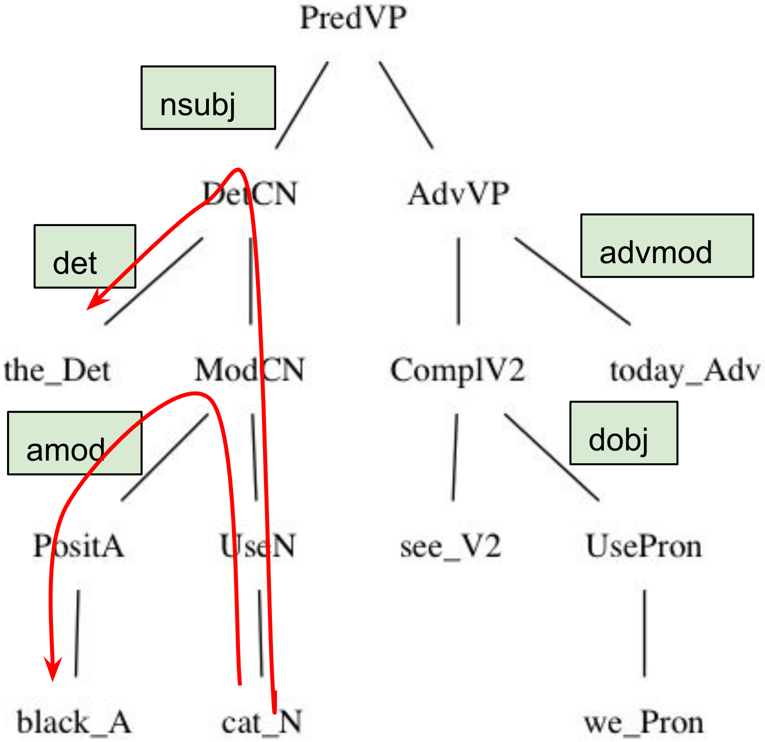


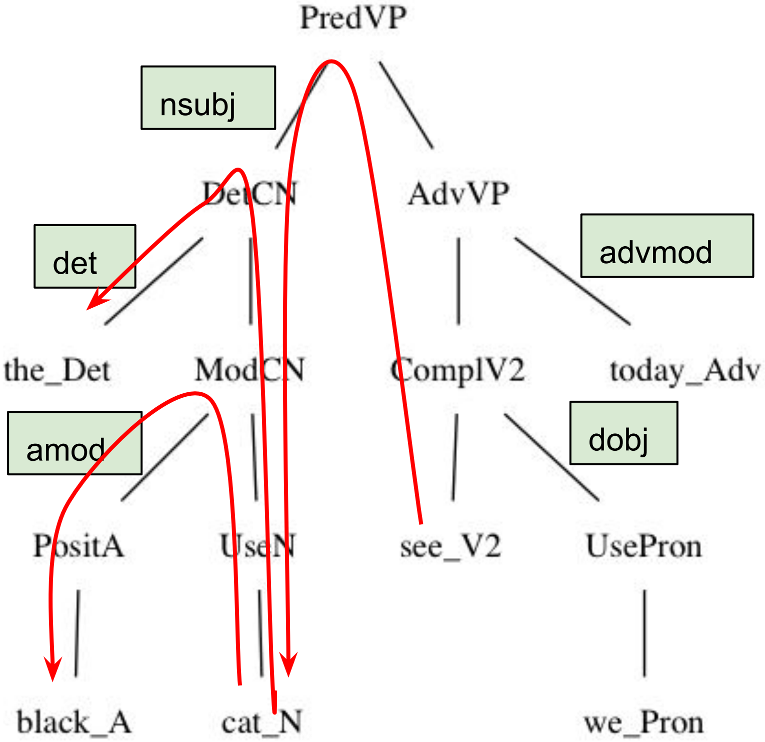
abstract syntax	dependency configuration
`PredVP  : NP   -> VP   -> Cl`	`nsubj  head`
`ComplV2 : V2   -> NP   -> VP`	`head   dobj`
`AdvVP   : VP   -> Adv  -> VP`	`head   advmod`
`DetCN   : Det  -> CN   -> NP`	`det    head`
`ModCN   : AP   -> CN   -> CN`	`amod   head`
`UseN    : N     -> CN`	`head`
`UsePron : Pron  -> NP`	`head`
`PositA  : A     -> AP`	`head`

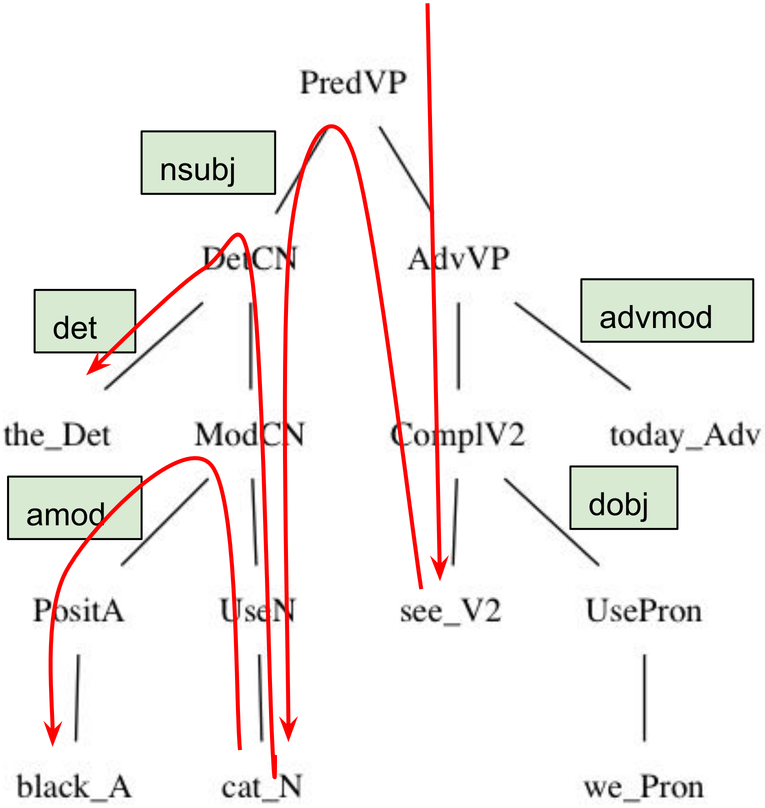


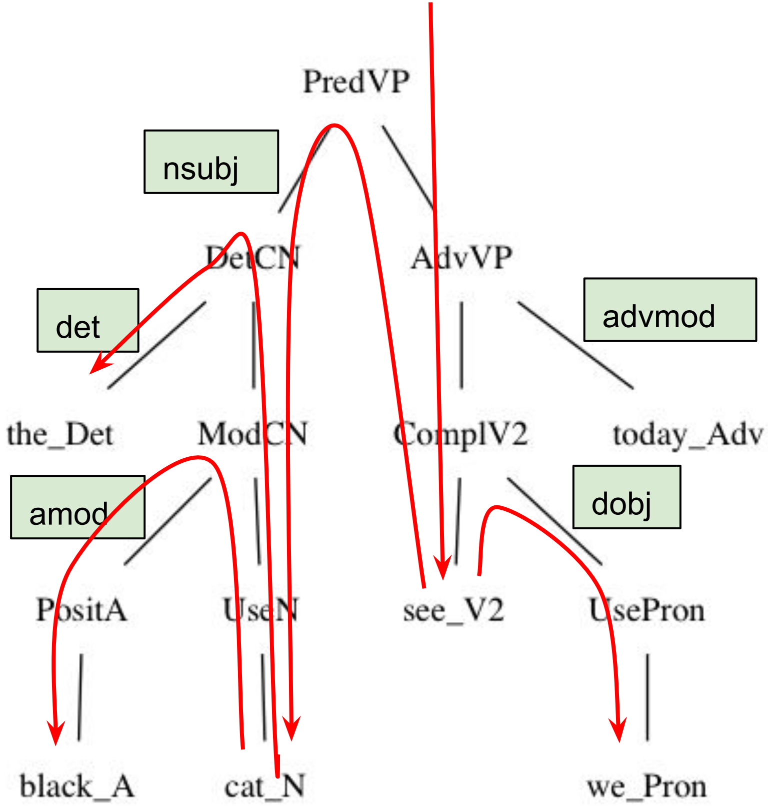

PredVP
nsubj
DetCN
AdvVP
det
advmod
the_Det
ModCN
ComplV2
today_Adv
amod
dobj
PositA
UseN
see_V2
UsePron
black_A
cat_N
we_Pron

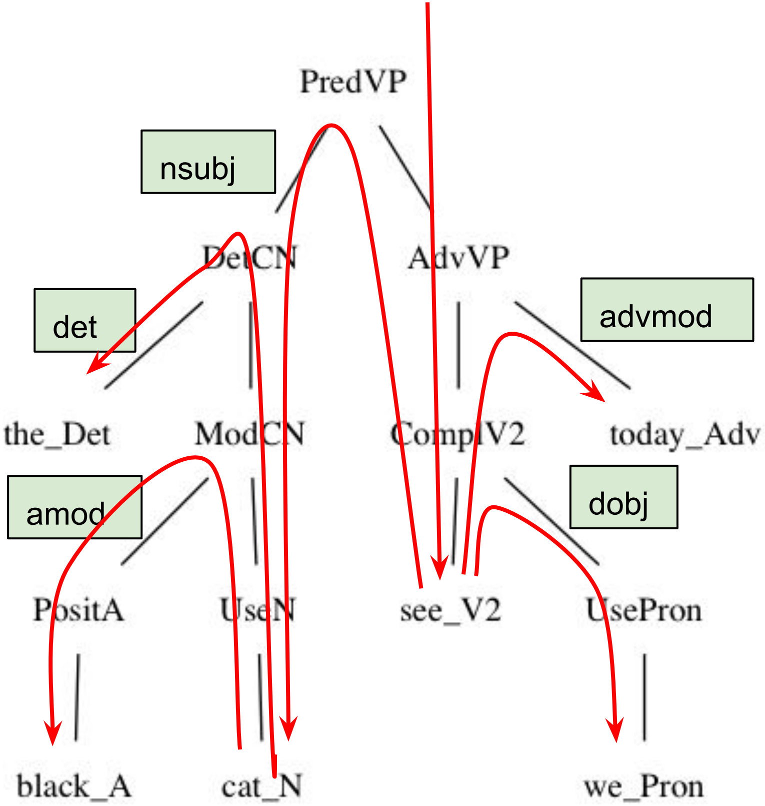

PredVP
nsubj
DetCN
AdvVP
det
advmod
the_Det
ModCN
ComplV2
today_Adv
amod
dobj
PositA
UseN
see_V2
UsePron
black_A
cat_N
we_Pron

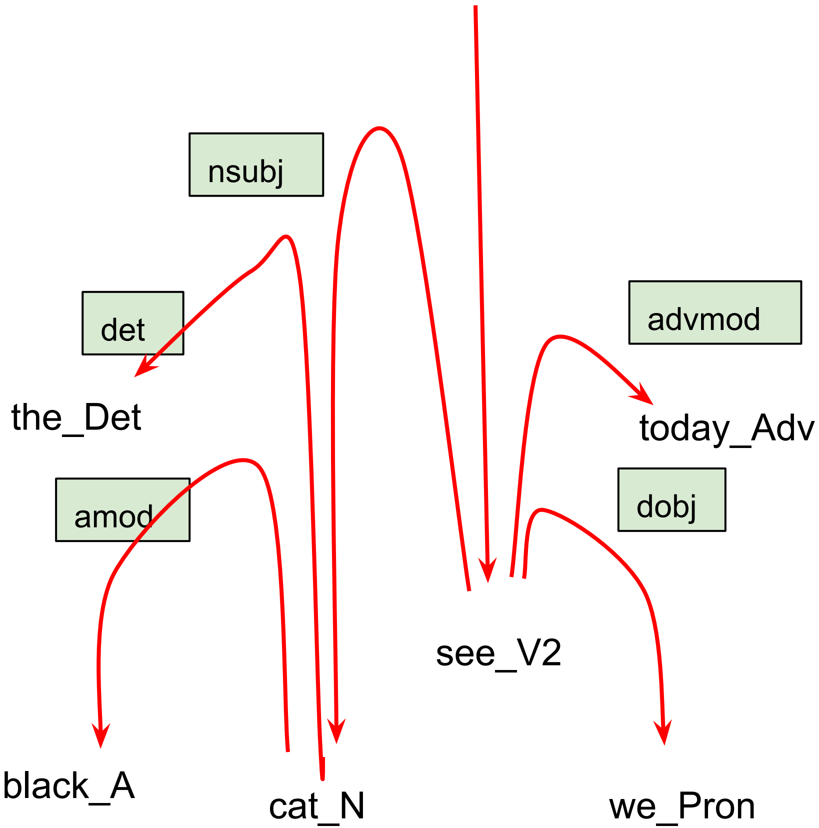

PredVP
nsubj
DetCN
AdvVP
det
advmod
the_Det
ModCN
ComplV2
today_Adv
amod
dobj
PositA
UseN
see_V2
UsePron
black_A
cat_N
we_Pron

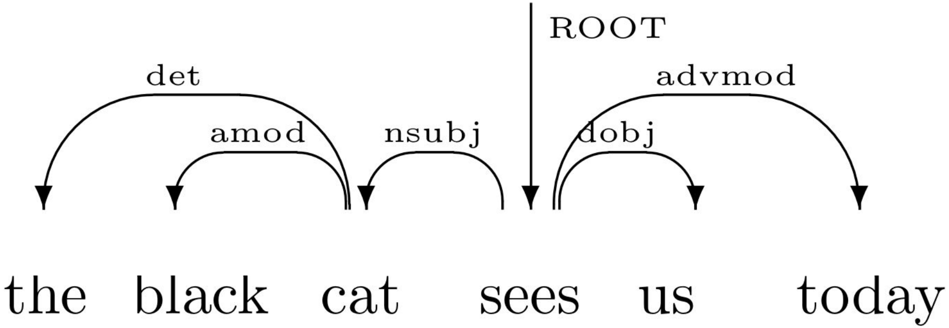

PredVP
nsubj
DetCN
AdvVP
det
advmod
the_Det
ModCN
ComplV2
today_Adv
amod
dobj
PositA
UseN
see_V2
UsePron
black_A
cat_N
we_Pron

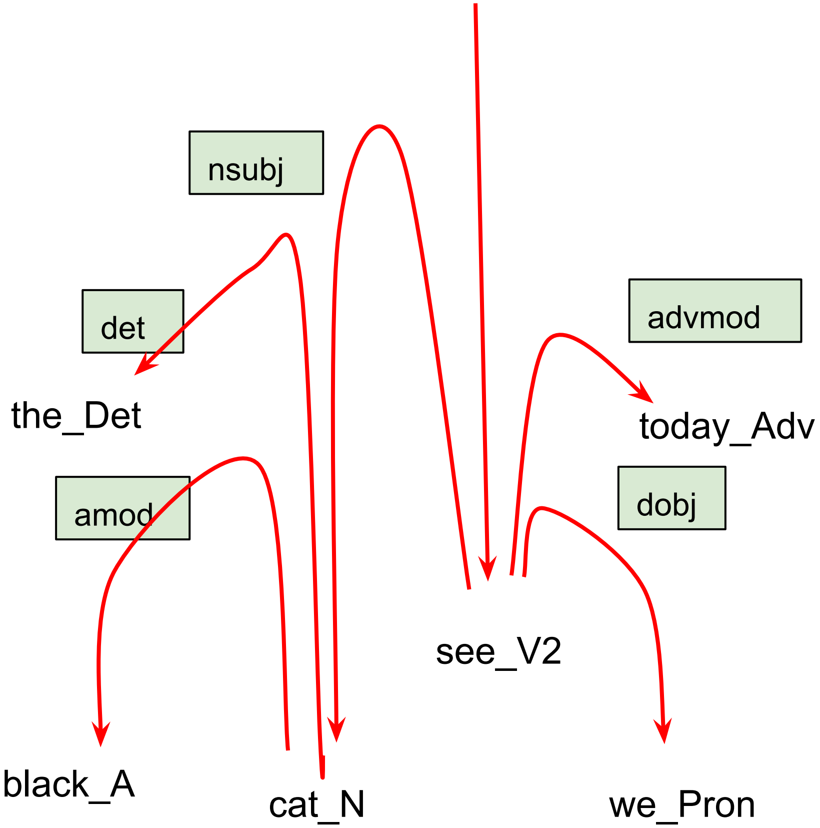

PredVP
nsubj
DetCN
AdvVP
det
advmod
the_Det
ModCN
ComplV2
today_Adv
amod
dobj
PositA
UseN
see_V2
UsePron
black_A
cat_N
we_Pron


PredVP
nsubj
DetCN
AdvVP
det
advmod
the_Det
ModCN
ComplV2
today_Adv
amod
dobj
PositA
UseN
see_V2
UsePron
black_A
cat_N
we_Pron

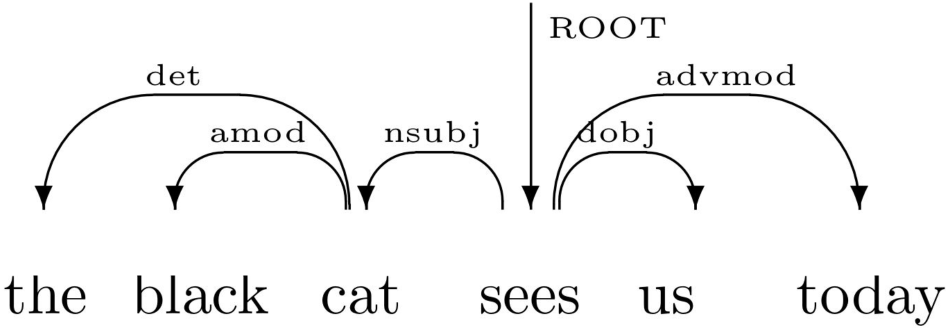

PredVP
nsubj
DetCN
AdvVP
det
advmod
the_Det
ModCN
ComplV2
today_Adv
amod
dobj
PositA
UseN
see_V2
UsePron
black_A
cat_N
we_Pron

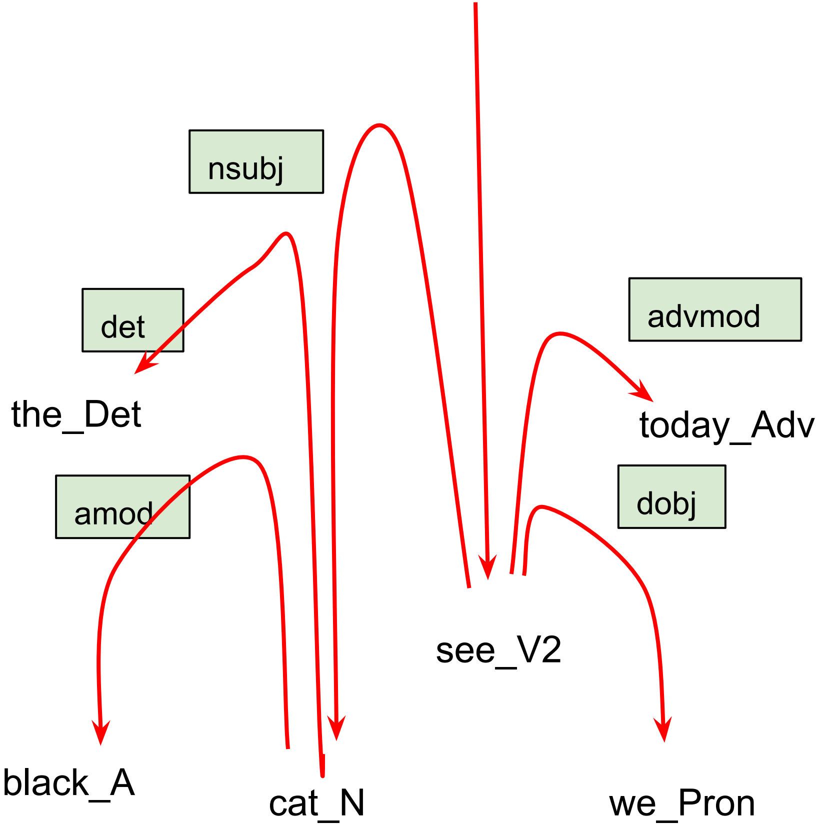

nsubj
det
the_Det
advmod
today_Adv
amod
dobj
see_V2
black_A
cat_N
we_Pron


ROOT
det
amod
nsubj
dobj
advmod
the
black
cat
sees
us
today

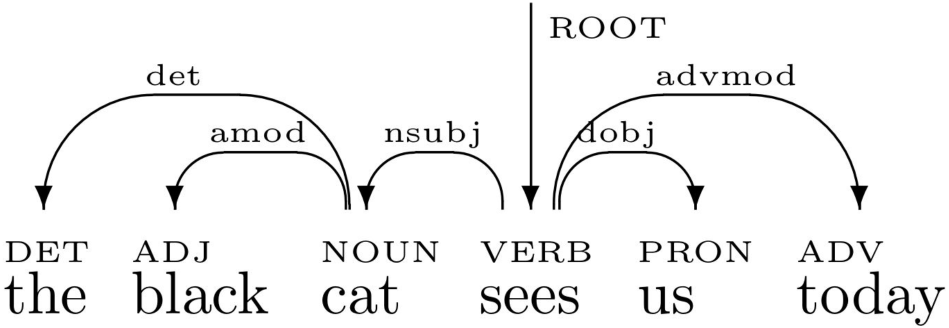

nsubj
det
the_Det
amod
black_A
cat_N
see_V2
advmod
today_Adv
dobj
we_Pron

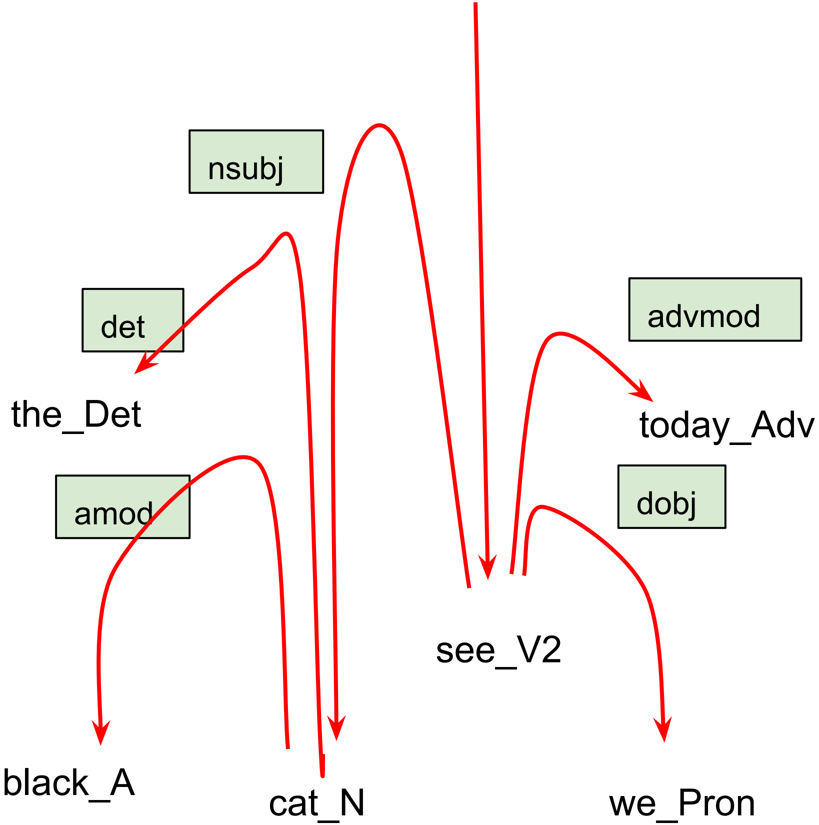
```
abstract syntax category configuration

Det         DET
A           ADJ
N           NOUN
V2          VERB
Pron        PRON
Adv         ADV
```

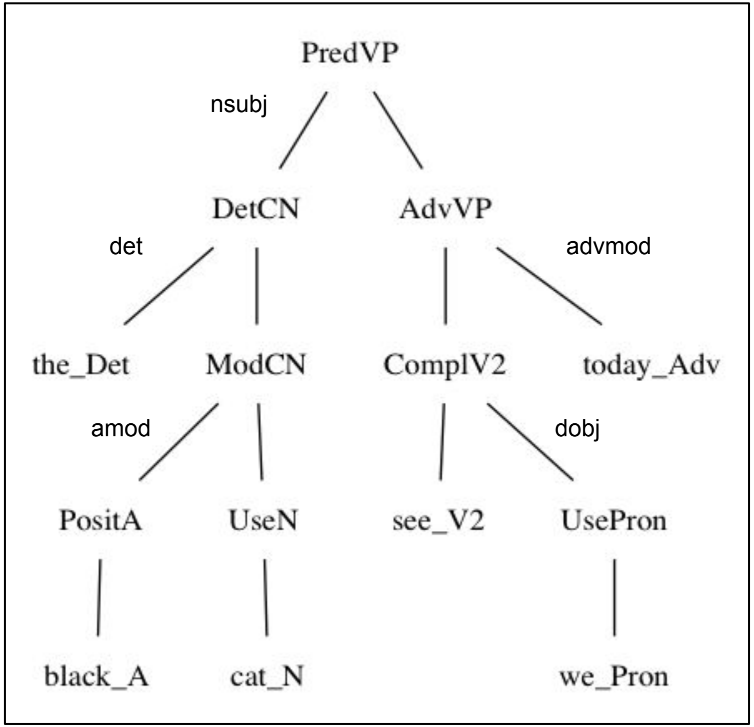


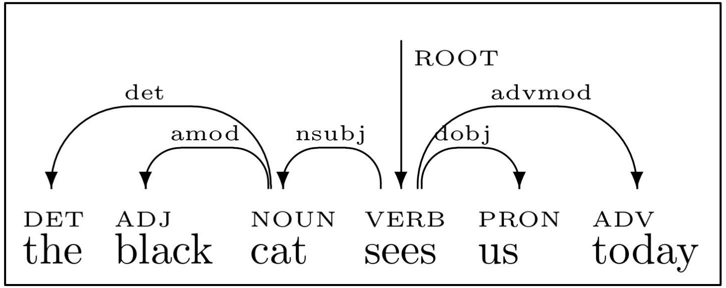


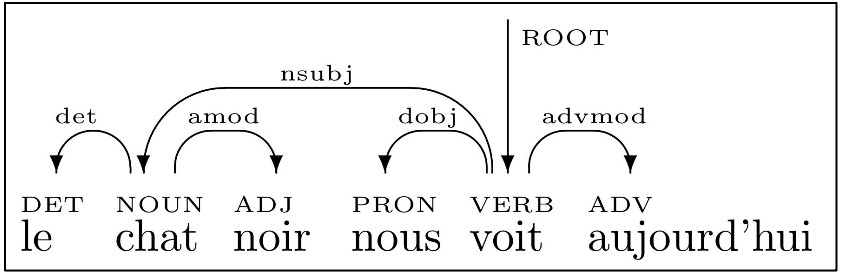
```
abstract syntax category configuration

Det         DET
A           ADJ
N           NOUN
V2          VERB
Pron        PRON
Adv         ADV
```



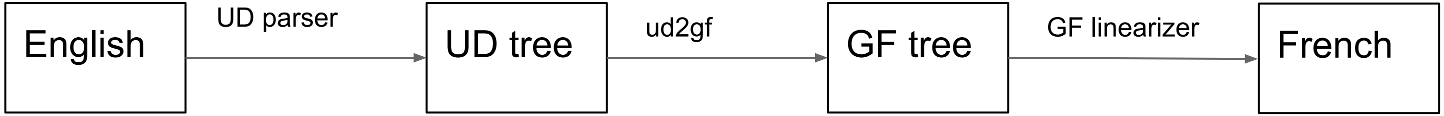



PredVP
nsubj
DetCN
AdvVP
det
advmod
the_Det
ModCN
ComplV2
today_Adv
amod
dobj
PositA
UseN
see_V2
UsePron
black_A
cat_N
we_Pron


ROOT
det
advmod
amod
nsubj
dobj
DET
ADJ
NOUN
VERB
PRON
ADV
the
black
cat
sees
us
today


ROOT
nsubj
det
amod
dobj
advmod
DET
NOUN
ADJ
PRON
VERB
ADV
le
chat
noir
nous
voit
aujourd'hui

Example pipeline 1



English

UD parser

UD tree

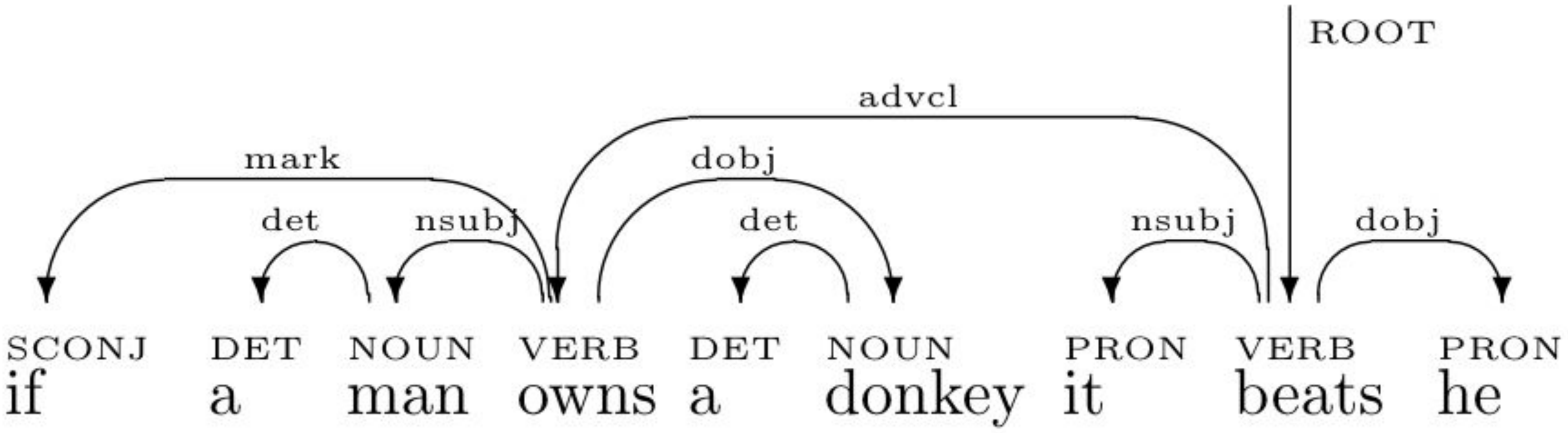
ud2gf

GF tree

GF linearizer

French

if a man owns a donkey it beats he


ROOT
advcl
mark
dobj
det
nsubj
det
nsubj
dobj
SCONJ
if
DET
a
NOUN
man
VERB
owns
DET
a
NOUN
donkey
PRON
it
VERB
beats
PRON
he

PARSER OUTPUT IN CONLL FORMAT:

```
1     if    if    SCONJ SCONJ _     4     mark  _     _
2     a     a     DET   DET   Definite=Ind|PronType=Art   3     det   _     _
3     man   man   NOUN  NOUN  Number=Sing     4     nsubj _     _
4     owns  own   VERB  VERB  Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin  8     advcl _     _
5     a     a     DET   DET   Definite=Ind|PronType=Art   6     det   _     _
6     donkey      donkey      NOUN  NOUN  Number=Sing     4     dobj  _     _
7     it    it    PRON  PRON  Case=Nom|Gender=Neut|Number=Sing|Person=3|PronType=Prs 8     nsubj _     _
8     beats beat  VERB  VERB  Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin  0     root  _     _
9     he    he    PRON  PRON  Case=Nom|Gender=Masc|Number=Sing|Person=3|PronType=Prs 8     dobj  _     _
```

```
STRUCTURED TREE:

root beat VERB Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin 8
    advcl own VERB Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin 4
        mark if SCONJ _ 1
        nsubj man NOUN Number=Sing 3
            det a DET Definite=Ind|PronType=Art 2
        dobj donkey NOUN Number=Sing 6
            det a DET Definite=Ind|PronType=Art 5
    nsubj it PRON Case=Nom|Gender=Neut|Number=Sing|Person=3|PronType=Prs 7
    dobj he PRON Case=Nom|Gender=Masc|Number=Sing|Person=3|PronType=Prs 9
```

Ranta & Kolachina, From Universal Dependencies
to Abstract Syntax, UD Workshop, 2017.

```
LEXICALLY ANNOTATED TREE:

root VERB beat_V2 : V2 [beat_V : V] {} (8) 8
    advcl VERB own_V2 : V2 [] {} (4) 4
        mark SCONJ if_Subj : Subj [] {} (1) 1
        nsubj NOUN man_N : N [] {} (3) 3
            det DET IndefArt : Quant [] {} (2) 2
        dobj NOUN donkey_N : N [] {} (6) 6
            det DET IndefArt : Quant [] {} (5) 5
    nsubj PRON "it" : Cleft_ [it_Pron : Pron] {} (7) 7
    dobj PRON he_Pron : Pron [] {} (9) 9
```

```
GF lexicon:

fun beat_V2 : V2
lin beat_V2 =
  mkV2 IrregEng.beat_V

fun own_V2 : V2
lin own_V2 = mkV2 “own”

fun man_N : N
lin man_N = mkN “man” “men”

fun donkey_N : N
lin donkey_N = mkN “donkey”

fun he_Pron : Pron
fun it_Pron : Pron
fun Cleft_ : NP -> RS -> Cl
fun IndefArt : Quant
fun if_Subj : Subj
```

A part of GF Resource Grammar Abstract Syntax:

```
fun
  PredVP   : NP -> VP -> Cl
  ComplV2  : V2 -> NP -> VP
  DetCN    : Det -> CN -> NP
  DetQuant : Quant -> Num -> Det
  AdvS     : Adv -> S -> S
  SubjS    : Subj -> S -> Adv
  UseCl    : Temp -> Pol -> Cl -> S
  UsePron  : Pron -> NP
  UseN     : N -> CN
```

```
Dependency configurations for abstract syntax:

fun
  PredVP   : NP -> VP -> Cl            -- nsubj head
  ComplV2  : V2 -> NP -> VP            -- head dobj
  DetCN    : Det -> CN -> NP           -- det head
  DetQuant : Quant -> Num -> Det       -- head [nummod]
  AdvS     : Adv -> S -> S             -- advcl head
  SubjS    : Subj -> S -> Adv          -- mark head
  UseCl    : Temp -> Pol -> Cl -> S    -- [aux] [neg] head
  UsePron  : Pron -> NP                -- head
  UseN     : N -> CN                   -- head
```

```
Dependency configurations for abstract syntax:

fun
  PredVP   : NP -> VP -> Cl             -- nsubj head
  ComplV2  : V2 -> NP -> VP             -- head dobj
  DetCN    : Det -> CN -> NP            -- det head
  DetQuant : Quant -> Num -> Det        -- head [nummod]
  AdvS     : Adv -> S -> S              -- advcl head
  SubjS    : Subj -> S -> Adv           -- mark head
  UseCl    : Temp -> Pol -> Cl -> S     -- [aux] [neg] head
  UsePron  : Pron -> NP                 -- head
  UseN     : N -> CN                    -- head

Helper functions:

  DetQuantSg_   : Quant -> Det  = \q  -> DetQuant q NumSg
  UseClPresPos_ : Cl -> S       = \cl -> UseCl Pres Pos cl
```

```
TRAVERSING THE TREE:

root VERB beat_V2 : V2 [beat_V : V] 8
    advcl VERB own_V2 : V2 4  (ComplV2 4 6)
        mark SCONJ if_Subj : Subj 1
        nsubj NOUN man_N : N 3
            det DET IndefArt : Quant 2 (DetQuantSg_ 2)
        dobj NOUN donkey_N : N 6
            det DET IndefArt : Quant 5
    nsubj PRON "it" : Cleft_ [it_Pron : Pron] 7 (UsePron 7)
    dobj PRON he_Pron : Pron 9 (UsePron 9)
```

```
PredVP      : NP -> VP -> Cl   -- nsubj head
ComplV2     : V2 -> NP -> VP   -- head dobj
DetCN       : Det -> CN -> NP  -- det head
AdvS        : Adv -> S -> S    -- advcl head
SubjS       : Subj -> S -> Adv -- mark head
UsePron     : Pron -> NP
UseN        : N -> CN
DetQuantSg_ : Quant -> Det
UseClPrPos_ : Cl -> S
```

```
TRAVERSING THE TREE:

root VERB beat_V2 : V2 [beat_V : V] 8
    advcl VERB own_V2 : V2 4  (ComplV2 4 6)
        mark SCONJ if_Subj : Subj 1
        nsubj NOUN man_N : N 3
            det DET IndefArt : Quant 2 (DetQuantSg_ 2)
        dobj NOUN donkey_N : N 6
            det DET IndefArt : Quant 5
    nsubj PRON "it" : Cleft_ [it_Pron : Pron] 7 (UsePron 7)
    dobj PRON he_Pron : Pron 9 (UsePron 9)
```

```
PredVP       : NP -> VP -> Cl   -- nsubj head
ComplV2      : V2 -> NP -> VP   -- head dobj
DetCN        : Det -> CN -> NP  -- det head
AdvS         : Adv -> S -> S    -- advcl head
SubjS        : Subj -> S -> Adv -- mark head
UsePron      : Pron -> NP
UseN         : N -> CN
DetQuantSg_  : Quant -> Det
UseClPrPos_  : Cl -> S
```

```
TRAVERSING THE TREE:

root VERB beat_V2 : V2 [beat_V : V] 8
    advcl VERB own_V2 : V2 4  (ComplV2 4 6)
        mark SCONJ if_Subj : Subj 1
        nsubj NOUN man_N : N 3         (UseN 3)
            det DET IndefArt : Quant 2 (DetQuantSg_ 2)
        dobj NOUN donkey_N : N 6
            det DET IndefArt : Quant 5
    nsubj PRON "it" : Cleft_ [it_Pron : Pron] 7 (UsePron 7)
    dobj PRON he_Pron : Pron 9 (UsePron 9)
```

```
PredVP      : NP -> VP -> Cl   -- nsubj head
ComplV2     : V2 -> NP -> VP   -- head dobj
DetCN       : Det -> CN -> NP  -- det head
AdvS        : Adv -> S -> S    -- advcl head
SubjS       : Subj -> S -> Adv -- mark head
UsePron     : Pron -> NP
UseN        : N -> CN
DetQuantSg_ : Quant -> Det
UseClPrPos_ : Cl -> S
```

```
TRAVERSING THE TREE:

root VERB beat_V2 : V2 [beat_V : V] 8
    advcl VERB own_V2 : V2 4  (ComplV2 4 6)
        mark SCONJ if_Subj : Subj 1
        nsubj NOUN man_N : N 3         (UseN 3) (DetCN 2 3)
            det DET IndefArt : Quant 2 (DetQuantSg_ 2)
        dobj NOUN donkey_N : N 6
            det DET IndefArt : Quant 5
    nsubj PRON "it" : Cleft_ [it_Pron : Pron] 7 (UsePron 7)
    dobj PRON he_Pron : Pron 9 (UsePron 9)
```

```
PredVP       : NP -> VP -> Cl   -- nsubj head
ComplV2      : V2 -> NP -> VP   -- head dobj
DetCN        : Det -> CN -> NP  -- det head
AdvS         : Adv -> S -> S    -- advcl head
SubjS        : Subj -> S -> Adv -- mark head
UsePron      : Pron -> NP
UseN         : N -> CN
DetQuantSg_  : Quant -> Det
UseClPrPos_  : Cl -> S
```

```
TRAVERSING THE TREE:

root VERB beat_V2 : V2 [beat_V : V] 8
    advcl VERB own_V2 : V2 4  (ComplV2 4 6)
        mark SCONJ if_Subj : Subj 1
        nsubj NOUN man_N : N 3         (UseN 3) (DetCN 2 3)
            det DET IndefArt : Quant 2 (DetQuantSg_ 2)
        dobj NOUN donkey_N : N 6       (UseN 6) (DetCN 5 6)
            det DET IndefArt : Quant 5 (DetQuantSg_ 5)
    nsubj PRON "it" : Cleft_ [it_Pron : Pron] 7 (UsePron 7)
    dobj PRON he_Pron : Pron 9 (UsePron 9)
```

```
PredVP      : NP -> VP -> Cl   -- nsubj head
ComplV2     : V2 -> NP -> VP   -- head dobj
DetCN       : Det -> CN -> NP  -- det head
AdvS        : Adv -> S -> S    -- advcl head
SubjS       : Subj -> S -> Adv -- mark head
UsePron     : Pron -> NP
UseN        : N -> CN
DetQuantSg_ : Quant -> Det
UseClPrPos_ : Cl -> S
```

```
TRAVERSING THE TREE:

root VERB beat_V2 : V2 [beat_V : V] 8
    advcl VERB own_V2 : V2 4  (ComplV2 4 6)
        mark SCONJ if_Subj : Subj 1
        nsubj NOUN man_N : N 3         (UseN 3) (DetCN 2 3)
            det DET IndefArt : Quant 2 (DetQuantSg_ 2)
        dobj NOUN donkey_N : N 6       (UseN 6) (DetCN 5 6)
            det DET IndefArt : Quant 5 (DetQuantSg_ 5)
    nsubj PRON "it" : Cleft_ [it_Pron : Pron] 7 (UsePron 7)
    dobj PRON he_Pron : Pron 9 (UsePron 9)
```

```
PredVP       : NP -> VP -> Cl   -- nsubj head
ComplV2      : V2 -> NP -> VP   -- head dobj
DetCN        : Det -> CN -> NP  -- det head
AdvS         : Adv -> S -> S    -- advcl head
SubjS        : Subj -> S -> Adv -- mark head
UsePron      : Pron -> NP
UseN         : N -> CN
DetQuantSg_  : Quant -> Det
UseClPrPos_  : Cl -> S
```

```
TRAVERSING THE TREE:

root VERB beat_V2 : V2 [beat_V : V] 8
    advcl VERB own_V2 : V2 4  (ComplV2 4 6)
        mark SCONJ if_Subj : Subj 1
        nsubj NOUN man_N : N 3         (UseN 3) (DetCN 2 3)
            det DET IndefArt : Quant 2 (DetQuantSg_ 2)
        dobj NOUN donkey_N : N 6       (UseN 6) (DetCN 5 6)
            det DET IndefArt : Quant 5 (DetQuantSg_ 5)
    nsubj PRON "it" : Cleft_ [it_Pron : Pron] 7 (UsePron 7)
    dobj PRON he_Pron : Pron 9 (UsePron 9)
```

```
PredVP      : NP -> VP -> Cl   -- nsubj head
ComplV2     : V2 -> NP -> VP   -- head dobj
DetCN       : Det -> CN -> NP  -- det head
AdvS        : Adv -> S -> S    -- advcl head
SubjS       : Subj -> S -> Adv -- mark head
UsePron     : Pron -> NP
UseN        : N -> CN
DetQuantSg_ : Quant -> Det
UseClPrPos_ : Cl -> S
```

```
TRAVERSING THE TREE:

root VERB beat_V2 : V2 [beat_V : V] 8
    advcl VERB own_V2 : V2 4  (ComplV2 4 6) (PredVP 3 4)
        mark SCONJ if_Subj : Subj 1
        nsubj NOUN man_N : N 3          (UseN 3) (DetCN 2 3)
            det DET IndefArt : Quant 2 (DetQuantSg_ 2)
        dobj NOUN donkey_N : N 6        (UseN 6) (DetCN 5 6)
            det DET IndefArt : Quant 5 (DetQuantSg_ 5)
    nsubj PRON "it" : Cleft_ [it_Pron : Pron] 7 (UsePron 7)
    dobj PRON he_Pron : Pron 9 (UsePron 9)
```

```
PredVP      : NP -> VP -> Cl    -- nsubj head
ComplV2     : V2 -> NP -> VP    -- head dobj
DetCN       : Det -> CN -> NP   -- det head
AdvS        : Adv -> S -> S     -- advcl head
SubjS       : Subj -> S -> Adv -- mark head
UsePron     : Pron -> NP
UseN        : N -> CN
DetQuantSg_ : Quant -> Det
UseClPrPos_ : Cl -> S
```

```
TRAVERSING THE TREE:

root VERB beat_V2 : V2 [beat_V : V] 8
    advcl VERB own_V2 : V2 4  (ComplV2 4 6) (PredVP 3 4) (UseClPrPos_ 4)
        mark SCONJ if_Subj : Subj 1
        nsubj NOUN man_N : N 3         (UseN 3) (DetCN 2 3)
            det DET IndefArt : Quant 2 (DetQuantSg_ 2)
        dobj NOUN donkey_N : N 6       (UseN 6) (DetCN 5 6)
            det DET IndefArt : Quant 5 (DetQuantSg_ 5)
    nsubj PRON "it" : Cleft_ [it_Pron : Pron] 7 (UsePron 7)
    dobj PRON he_Pron : Pron 9 (UsePron 9)
```

```
PredVP       : NP -> VP -> Cl   -- nsubj head
ComplV2      : V2 -> NP -> VP   -- head dobj
DetCN        : Det -> CN -> NP  -- det head
AdvS         : Adv -> S -> S    -- advcl head
SubjS        : Subj -> S -> Adv -- mark head
UsePron      : Pron -> NP
UseN         : N -> CN
DetQuantSg_  : Quant -> Det
UseClPrPos_  : Cl -> S
```

```
TRAVERSING THE TREE:

root VERB beat_V2 : V2 [beat_V : V] 8
    advcl VERB own_V2 : V2 4  (ComplV2 4 6) (PredVP 3 4) (UseClPrPos_ 4) (AdvS 1 4)
        mark SCONJ if_Subj : Subj 1
        nsubj NOUN man_N : N 3         (UseN 3) (DetCN 2 3)
            det DET IndefArt : Quant 2 (DetQuantSg_ 2)
        dobj NOUN donkey_N : N 6       (UseN 6) (DetCN 5 6)
            det DET IndefArt : Quant 5 (DetQuantSg_ 5)
    nsubj PRON "it" : Cleft_ [it_Pron : Pron] 7 (UsePron 7)
    dobj PRON he_Pron : Pron 9 (UsePron 9)
```

```
PredVP       : NP -> VP -> Cl   -- nsubj head
ComplV2      : V2 -> NP -> VP   -- head dobj
DetCN        : Det -> CN -> NP  -- det head
AdvS         : Adv -> S -> S    -- advcl head
SubjS        : Subj -> S -> Adv -- mark head
UsePron      : Pron -> NP
UseN         : N -> CN
DetQuantSg_  : Quant -> Det
UseClPrPos_  : Cl -> S
```

```
TRAVERSING THE TREE:

root VERB beat_V2 : V2 [beat_V : V] 8 (ComplV2 8 9)
    advcl VERB own_V2 : V2 4  (ComplV2 4 6) (PredVP 3 4) (UseClPrPos_ 4) (AdvS 1 4)
        mark SCONJ if_Subj : Subj 1
        nsubj NOUN man_N : N 3         (UseN 3) (DetCN 2 3)
            det DET IndefArt : Quant 2 (DetQuantSg_ 2)
        dobj NOUN donkey_N : N 6       (UseN 6) (DetCN 5 6)
            det DET IndefArt : Quant 5 (DetQuantSg_ 5)
    nsubj PRON "it" : Cleft_ [it_Pron : Pron] 7 (UsePron 7)
    dobj PRON he_Pron : Pron 9 (UsePron 9)
```

```
PredVP       : NP -> VP -> Cl   -- nsubj head
ComplV2      : V2 -> NP -> VP   -- head dobj
DetCN        : Det -> CN -> NP  -- det head
AdvS         : Adv -> S -> S    -- advcl head
SubjS        : Subj -> S -> Adv -- mark head
UsePron      : Pron -> NP
UseN         : N -> CN
DetQuantSg_  : Quant -> Det
UseClPrPos_  : Cl -> S
```

```
TRAVERSING THE TREE:

root VERB beat_V2 : V2 [beat_V : V] 8 (ComplV2 8 9) (PredVP 7 8)
    advcl VERB own_V2 : V2 4  (ComplV2 4 6) (PredVP 3 4) (UseClPrPos_ 4) (AdvS 1 4)
        mark SCONJ if_Subj : Subj 1
        nsubj NOUN man_N : N 3          (UseN 3) (DetCN 2 3)
            det DET IndefArt : Quant 2 (DetQuantSg_ 2)
        dobj NOUN donkey_N : N 6        (UseN 6) (DetCN 5 6)
            det DET IndefArt : Quant 5 (DetQuantSg_ 5)
    nsubj PRON "it" : Cleft_ [it_Pron : Pron] 7 (UsePron 7)
    dobj PRON he_Pron : Pron 9 (UsePron 9)
```

```
PredVP       : NP -> VP -> Cl    -- nsubj head
ComplV2      : V2 -> NP -> VP    -- head dobj
DetCN        : Det -> CN -> NP   -- det head
AdvS         : Adv -> S -> S     -- advcl head
SubjS        : Subj -> S -> Adv -- mark head
UsePron      : Pron -> NP
UseN         : N -> CN
DetQuantSg_ : Quant -> Det
UseClPrPos_ : Cl -> S
```

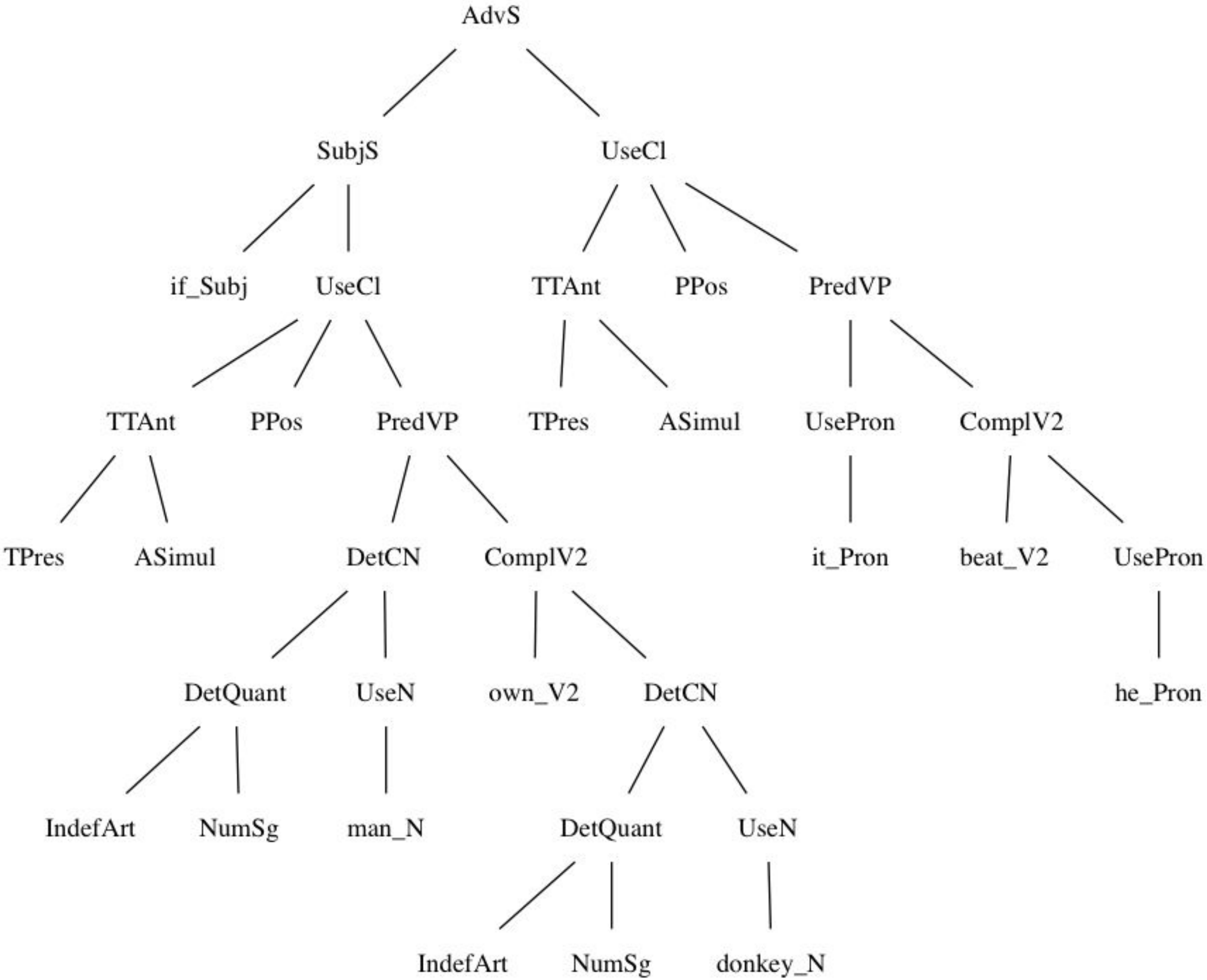
```
TRAVERSING THE TREE:

root VERB beat_V2 : V2 [beat_V : V] 8 (ComplV2 8 9) (PredVP 7 8) (UseClPrPos_ 8)
    advcl VERB own_V2 : V2 4  (ComplV2 4 6) (PredVP 3 4) (UseClPrPos_ 4) (AdvS 1 4)
        mark SCONJ if_Subj : Subj 1
        nsubj NOUN man_N : N 3         (UseN 3) (DetCN 2 3)
            det DET IndefArt : Quant 2 (DetQuantSg_ 2)
        dobj NOUN donkey_N : N 6       (UseN 6) (DetCN 5 6)
            det DET IndefArt : Quant 5 (DetQuantSg_ 5)
    nsubj PRON "it" : Cleft_ [it_Pron : Pron] 7 (UsePron 7)
    dobj PRON he_Pron : Pron 9 (UsePron 9)
```

```
PredVP      : NP -> VP -> Cl   -- nsubj head
ComplV2     : V2 -> NP -> VP   -- head dobj
DetCN       : Det -> CN -> NP  -- det head
AdvS        : Adv -> S -> S    -- advcl head
SubjS       : Subj -> S -> Adv -- mark head
UsePron     : Pron -> NP
UseN        : N -> CN
DetQuantSg_ : Quant -> Det
UseClPrPos_ : Cl -> S
```

```
TRAVERSING THE TREE:
                                                                          (AdvS 4 8)
root VERB beat_V2 : V2 [beat_V : V] 8 (ComplV2 8 9) (PredVP 7 8) (UseClPrPos_ 8)
    advcl VERB own_V2 : V2 4  (ComplV2 4 6) (PredVP 3 4) (UseClPrPos_ 4) (AdvS 1 4)
        mark SCONJ if_Subj : Subj 1
        nsubj NOUN man_N : N 3         (UseN 3) (DetCN 2 3)
            det DET IndefArt : Quant 2 (DetQuantSg_ 2)
        dobj NOUN donkey_N : N 6       (UseN 6) (DetCN 5 6)
            det DET IndefArt : Quant 5 (DetQuantSg_ 5)
    nsubj PRON "it" : Cleft_ [it_Pron : Pron] 7 (UsePron 7)
    dobj PRON he_Pron : Pron 9 (UsePron 9)
```

```
PredVP      : NP -> VP -> Cl    -- nsubj head
ComplV2     : V2 -> NP -> VP    -- head dobj
DetCN       : Det -> CN -> NP   -- det head
AdvS        : Adv -> S -> S     -- advcl head
SubjS       : Subj -> S -> Adv -- mark head
UsePron     : Pron -> NP
UseN        : N -> CN
DetQuantSg_ : Quant -> Det
UseClPrPos_ : Cl -> S
```

```
GF ABSTRACT SYNTAX TREE:

(AdvS
  (SubjS if_Subj
    (UseCl (TTAnt TPres ASimul) PPos
      (PredVP
        (DetCN (DetQuant IndefArt NumSg) (UseN man_N))
        (ComplV2 own_V2 (DetCN (DetQuant IndefArt NumSg) (UseNdonkey_N))))))
    (UseCl (TTAnt TPres ASimul) PPos
      (PredVP
        (UsePron it_Pron)
        (ComplV2 beat_V2 (UsePron he_Pron)))))
```

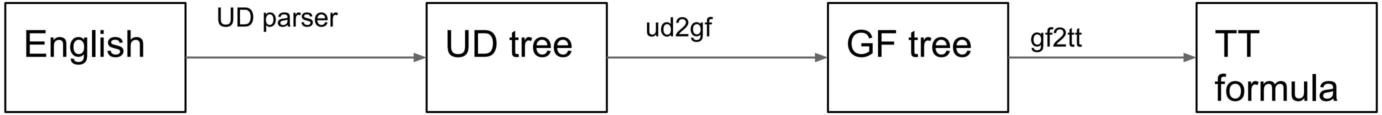

AdvS
SubjS
UseCl
if_Subj
UseCl
TTAnt
PPos
PredVP
TTAnt
PPos
PredVP
TPres
ASimul
UsePron
ComplV2
TPres
ASimul
DetCN
ComplV2
it_Pron
beat_V2
UsePron
DetQuant
UseN
own_V2
DetCN
he_Pron
IndefArt
NumSg
man_N
DetQuant
UseN
IndefArt
NumSg
donkey_N

si un homme possède un âne il le bat

si un homme possède un âne il le bat

if a man owns a donkey it beats him
if a man owns a donkey he beats it

Example pipeline 2

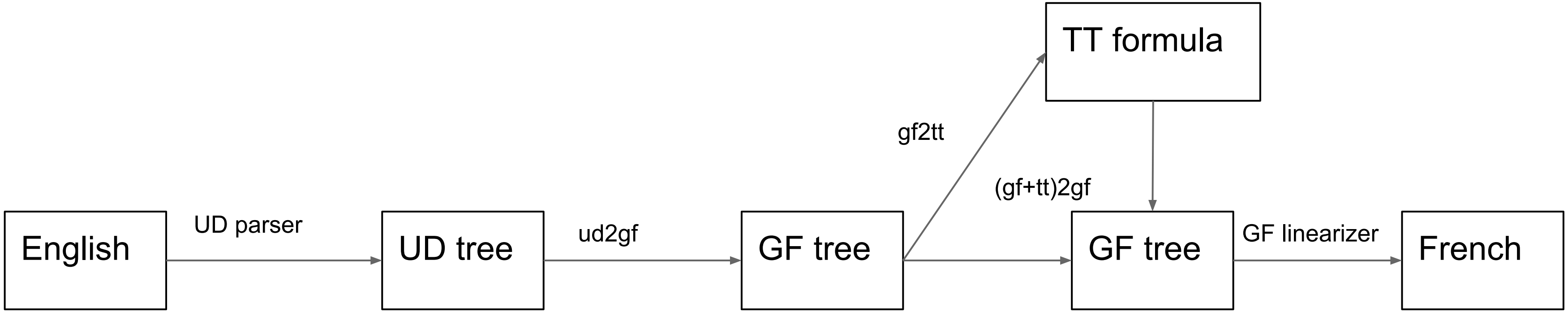


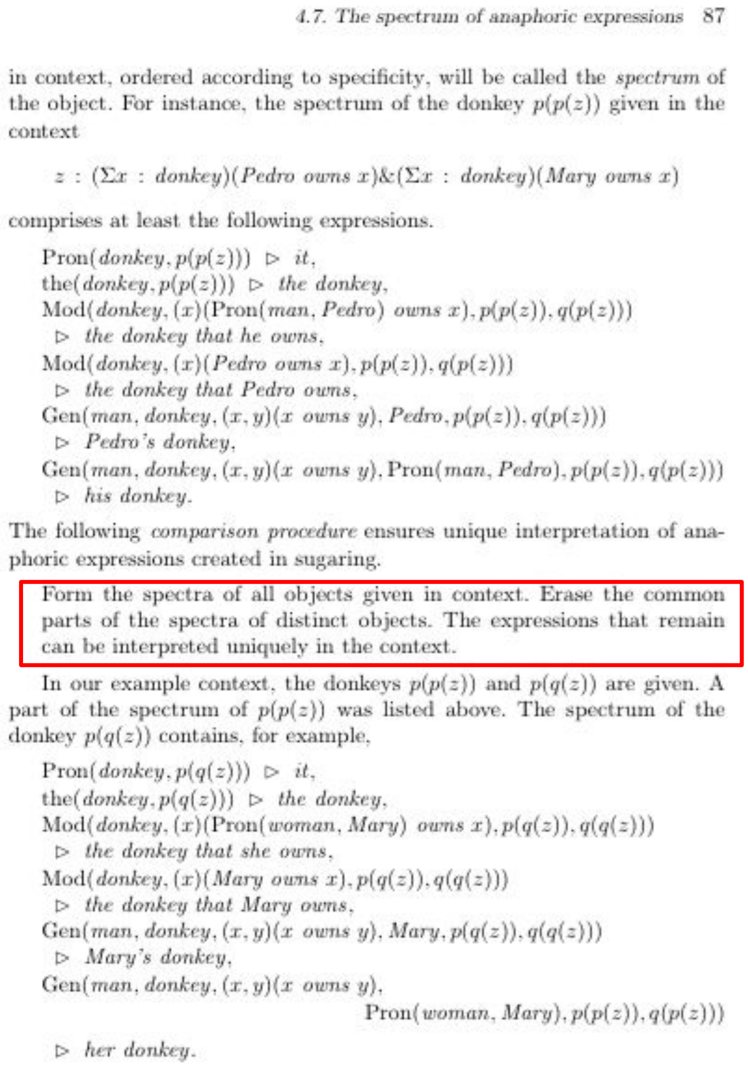
```
echo “if a man owns a donkey it beat he” | \
  \  udjpipe/scripts/pipeline.sh -l en \
    \  ud2gf -lEng -t10000 -k3000 -a1 -g1 -Dt -CUDTranslate.labels,UDTranslateEng.labels \
      \  runghc TTG.hs
```

```
iS :: GS -> Prop
iS s = case s of
  GUseCl _ pol cl -> iPol pol (iCl cl)
  GAdvS (GSubjS Gif_Subj a) b -> Pi (iS a) (\x -> iS b) --- non-compositional
  …
iCl :: GCl -> Prop
iCl s = case s of
  GPredVP np vp -> iNP np (iVP vp)
  …
iVP :: GVP -> Ind -> Prop
iVP vp x = case vp of
  GComplV2 v np -> iNP np (\y -> iV2 v x y)
  …
iNP :: GNP -> (Ind -> Prop) -> Prop
iNP np p = case np of
  GDetCN (GDetQuant GDefArt _) cn -> p (Def (iCN cn) [])
  GDetCN det cn -> iDet det (iCN cn) p
  …
iDet :: GDet -> Prop -> (Ind -> Prop) -> Prop
iDet det t p = case det of
  GsomeSg_Det -> Sigma t p
  every_Det -> Pi t p
  GDetQuant GIndefArt _ -> Sigma t p --- non-compositional
```

$$
\begin{aligned}
&(\Pi z :\\
&\qquad (\Sigma x : man\_N)(\Sigma y : donkey\_N)own\_V2(x,y))\\
&\quad beat\_V2(p(q(z)),p(z))
\end{aligned}
$$

Example pipeline 3





in context, ordered according to specificity, will be called the *spectrum* of the object. For instance, the spectrum of the donkey $p(p(z))$ given in the context

$$z : (\Sigma x : \mathit{donkey})(\mathit{Pedro\ owns}\ x)\&(\Sigma x : \mathit{donkey})(\mathit{Mary\ owns}\ x)$$

comprises at least the following expressions.

$\mathrm{Pron}(\mathit{donkey}, p(p(z)))\ \triangleright\ \mathit{it},$
$\mathrm{the}(\mathit{donkey}, p(p(z)))\ \triangleright\ \mathit{the\ donkey},$
$\mathrm{Mod}(\mathit{donkey}, (x)(\mathrm{Pron}(\mathit{man}, \mathit{Pedro})\ \mathit{owns}\ x), p(p(z)), q(p(z)))$
$\triangleright\ \mathit{the\ donkey\ that\ he\ owns},$
$\mathrm{Mod}(\mathit{donkey}, (x)(\mathit{Pedro\ owns}\ x), p(p(z)), q(p(z)))$
$\triangleright\ \mathit{the\ donkey\ that\ Pedro\ owns},$
$\mathrm{Gen}(\mathit{man}, \mathit{donkey}, (x, y)(x\ \mathit{owns}\ y), \mathit{Pedro}, p(p(z)), q(p(z)))$
$\triangleright\ \mathit{Pedro's\ donkey},$
$\mathrm{Gen}(\mathit{man}, \mathit{donkey}, (x, y)(x\ \mathit{owns}\ y), \mathrm{Pron}(\mathit{man}, \mathit{Pedro}), p(p(z)), q(p(z)))$
$\triangleright\ \mathit{his\ donkey}.$

The following *comparison procedure* ensures unique interpretation of anaphoric expressions created in sugaring.

> Form the spectra of all objects given in context. Erase the common parts of the spectra of distinct objects. The expressions that remain can be interpreted uniquely in the context.

In our example context, the donkeys $p(p(z))$ and $p(q(z))$ are given. A part of the spectrum of $p(p(z))$ was listed above. The spectrum of the donkey $p(q(z))$ contains, for example,

$\mathrm{Pron}(\mathit{donkey}, p(q(z)))\ \triangleright\ \mathit{it},$
$\mathrm{the}(\mathit{donkey}, p(q(z)))\ \triangleright\ \mathit{the\ donkey},$
$\mathrm{Mod}(\mathit{donkey}, (x)(\mathrm{Pron}(\mathit{woman}, \mathit{Mary})\ \mathit{owns}\ x), p(q(z)), q(q(z)))$
$\triangleright\ \mathit{the\ donkey\ that\ she\ owns},$
$\mathrm{Mod}(\mathit{donkey}, (x)(\mathit{Mary\ owns}\ x), p(q(z)), q(q(z)))$
$\triangleright\ \mathit{the\ donkey\ that\ Mary\ owns},$
$\mathrm{Gen}(\mathit{man}, \mathit{donkey}, (x, y)(x\ \mathit{owns}\ y), \mathit{Mary}, p(q(z)), q(q(z)))$
$\triangleright\ \mathit{Mary's\ donkey},$
$\mathrm{Gen}(\mathit{man}, \mathit{donkey}, (x, y)(x\ \mathit{owns}\ y),$
$\mathrm{Pron}(\mathit{woman}, \mathit{Mary}), p(p(z)), q(p(z)))$
$\triangleright\ \mathit{her\ donkey}.$

AR, Type Theoretical Grammar, OUP 1994

```
GF ABSTRACT SYNTAX TREE:

(AdvS
  (SubjS if_Subj
    (UseCl (TTAnt TPres ASimul) PPos
      (PredVP
        (DetCN (DetQuant IndefArt NumSg) (UseN man_N))
        (ComplV2 own_V2 (DetCN (DetQuant IndefArt NumSg) (UseN
donkey_N))))))
    (UseCl (TTAnt TPres ASimul) PPos
      (PredVP
        (UsePron it_Pron)
        (ComplV2 beat_V2 (UsePron he_Pron)))))
```

```
GF ABSTRACT SYNTAX TREE:

(AdvS
  (SubjS if_Subj
    (UseCl (TTAnt TPres ASimul) PPos
      (PredVP
        (DetCN (DetQuant IndefArt NumSg) (UseN man_N))
        (ComplV2 own_V2 (DetCN (DetQuant IndefArt NumSg) (UseN
donkey_N))))))
    (UseCl (TTAnt TPres ASimul) PPos
      (PredVP
        (DetCN (DetQuant DefArt NumSg) (UseN donkey_N))
        (ComplV2 beat_V2 (UsePron he_Pron)))))
```

si un homme possède un âne l’âne le bat

“if a man owns a donkey the donkey beats him”

Research goals

Linguistics:

- What are the structures of language?

Research goals

Linguistics:

- What are the structures of language?

Strong AI:

- Can machines learn to perform like humans?

Research goals

Linguistics:

- What are the structures of language?

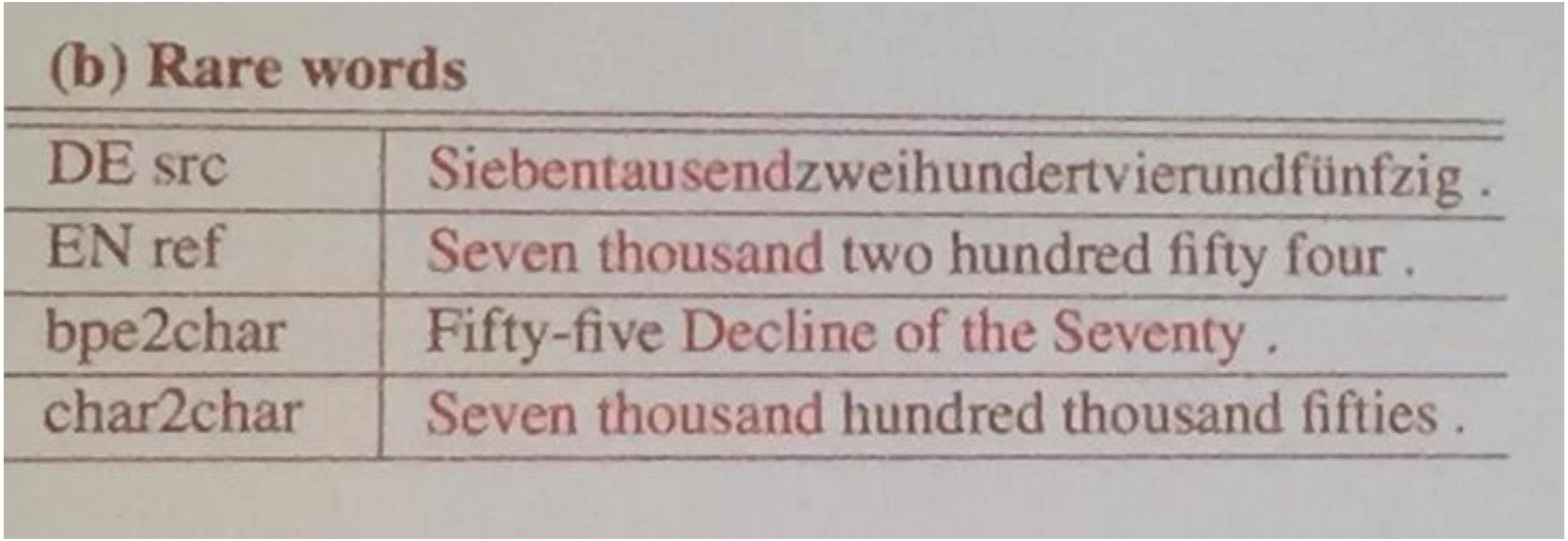
Strong AI:

- Can machines learn to perform like humans?

Engineering:

- How best to build systems that work?

Don’t guess if you know.



(b) Rare words

DE src	Siebentausendzweihundertvierundfünfzig .
EN ref	Seven thousand two hundred fifty four .
bpe2char	Fifty-five Decline of the Seventy .
char2char	Seven thousand hundred thousand fifties .

Descartes, letter to Mersenne, 1629

In a single day one can learn to name every one of the infinite series of numbers, and thus to **write infinitely many different words in an unknown language**.

```
ResourceDemo: num (pot3plus (pot1as2 (pot0as1 (pot0 n7))) (pot2plus (pot0 n2) (pot1plus n5 (pot0 n4))))
ResourceDemoAfr: sevenduisend tweehonderdvierenvyftig
ResourceDemoBul: седем хиляди двеста петдесет и четирима
ResourceDemoCat: set mil dos -cents cinquanta- quatre
ResourceDemoChi: 七 千 两 百 五 十 四
ResourceDemoDan: syv tusind og to hundrede og fire og halvtreds
ResourceDemoDut: zevenduizend tweehonderdvierenvijftig
ResourceDemoEng: seven thousand two hundred and fifty-four
ResourceDemoEst: seitse tuhat kakssada viiskümmend neli
ResourceDemoFin: seitsemäntuhatta kaksisataaviisikymmentäneljä
ResourceDemoFre: sept mille deux cent cinquante-quatre
ResourceDemoGer: siebentausend zweihundertvierundfünfzig
ResourceDemoGre: εφτά χιλιάδες διακόσιοι πενήντα τέσσερεις
ResourceDemoHin: सात हज़ार दो सौ चwwन
ResourceDemoIce: sjö þúsund tvö hundrað fimmtugasti og fjórði
ResourceDemoIta: settemila e duecentocinquantaquattro
ResourceDemoJpn: 七 千 二 百 五 十 四
ResourceDemoLav: septiņi tūkstoši divi simti piecdesmit četri
ResourceDemoMlt: sebat elef u mitejn u erbgħa u ħamsin
ResourceDemoMon: долоон мянга хоёр зуун тавин дөрөв
ResourceDemoNep: सात हजार दुई सय चवन् न
ResourceDemoNno: sju tusen og to hundre og femti fire
ResourceDemoNor: sju tusen og to hundre og femti fire
ResourceDemoPes: هفت هزار و دویست و پنجاه و چهار
ResourceDemoPnb: ست ہزار دو سو چوتنجا
ResourceDemoPol: siedem tysięcy dwieście pięćdziesiąt cztery
ResourceDemoRon: şapte mii două sute cincizeci şi patru
ResourceDemoRus: семь тысяч двести пятьдесят четыре
ResourceDemoSnd: ست هزار پ سو چوونجاھ
ResourceDemoSpa: siete mil doscientos cincuenta y cuatro
ResourceDemoSwe: sjutusen tvåhundra femtiofyra
ResourceDemoTha: เจ็ด พัน สอง ร้อย ห้า สิบ สี่
ResourceDemoUrd: سات ہزار دو سو چوون
```

Hammarström & Ranta, Cardinal Numerals Revisited in GF, 2004

Descartes, letter to Mersenne, 1629

In a single day one can learn to name every one of the infinite series of numbers, and thus to **write infinitely many different words in an unknown language.** The same could be done for all the other words necessary to express all the other things which fall within the purview of the human mind.

Descartes, letter to Mersenne, 1629

In a single day one can learn to name every one of the infinite series of numbers, and thus to write infinitely many different words in an unknown language. The same could be done for all the other words necessary to express all the other things which fall within the purview of the human mind.

the discovery of s**uch a language depends upon the true philosophy**. For without that philosophy it is impossible to number and order all the thoughts of men or even to separate them out into clear and simple thoughts

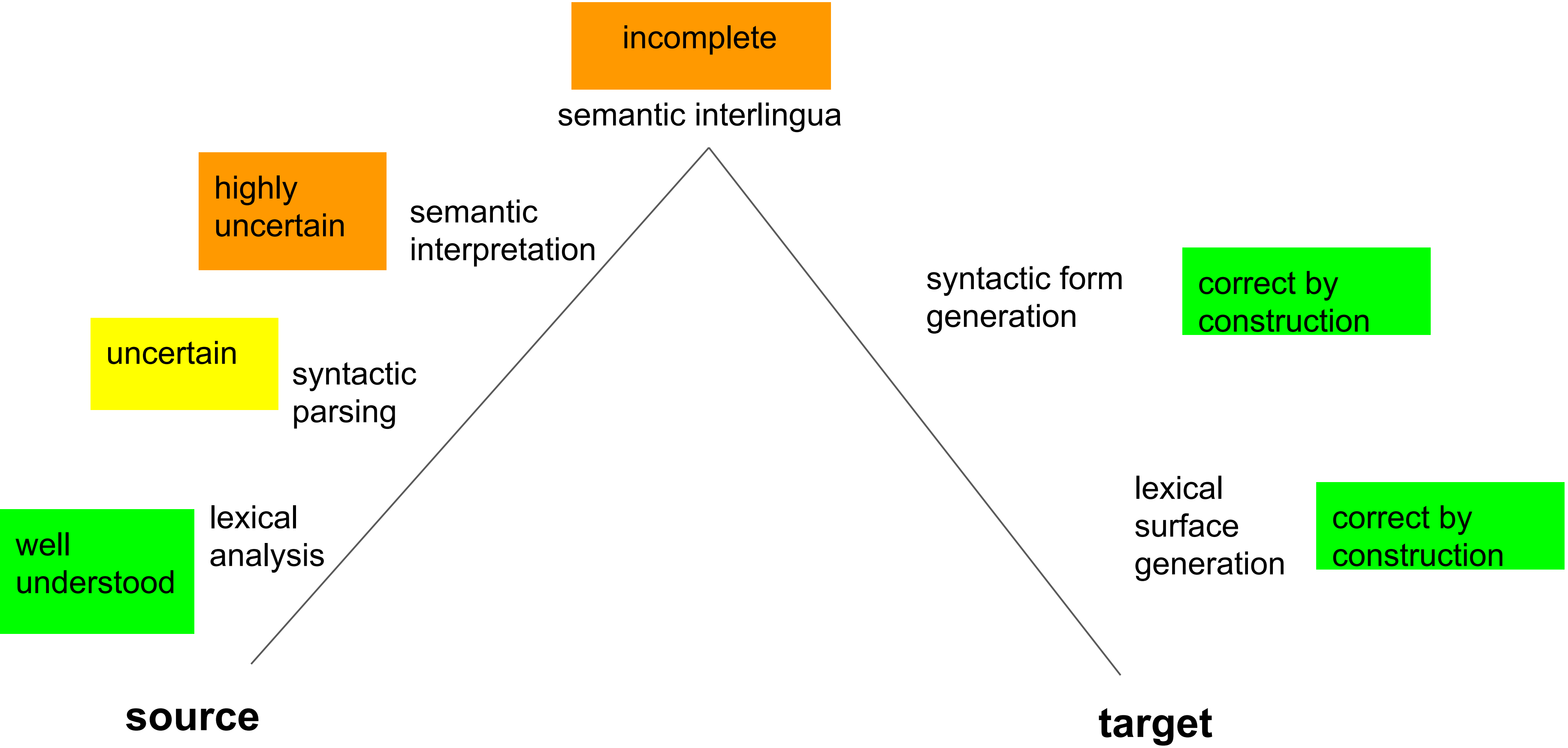
Descartes, letter to Mersenne, 1629

In a single day one can learn to name every one of the infinite series of numbers, and thus to write infinitely many different words in an unknown language. The same could be done for all the other words necessary to express all the other things which fall within the purview of the human mind.

the discovery of such a language depends upon the true philosophy. For without that philosophy it is impossible to number and order all the thoughts of men or even to separate them out into clear and simple thoughts,

But do not hope ever to see such a language in use. For that, the order of nature would have to change so that t**he world turned into a terrestrial paradise**; and that is too much to suggest outside of fairyland.

http://www.autodidactproject.org/other/descartes-lg1.html



incomplete

semantic interlingua

highly uncertain

semantic interpretation

uncertain

syntactic parsing

well understood

lexical analysis

syntactic form generation

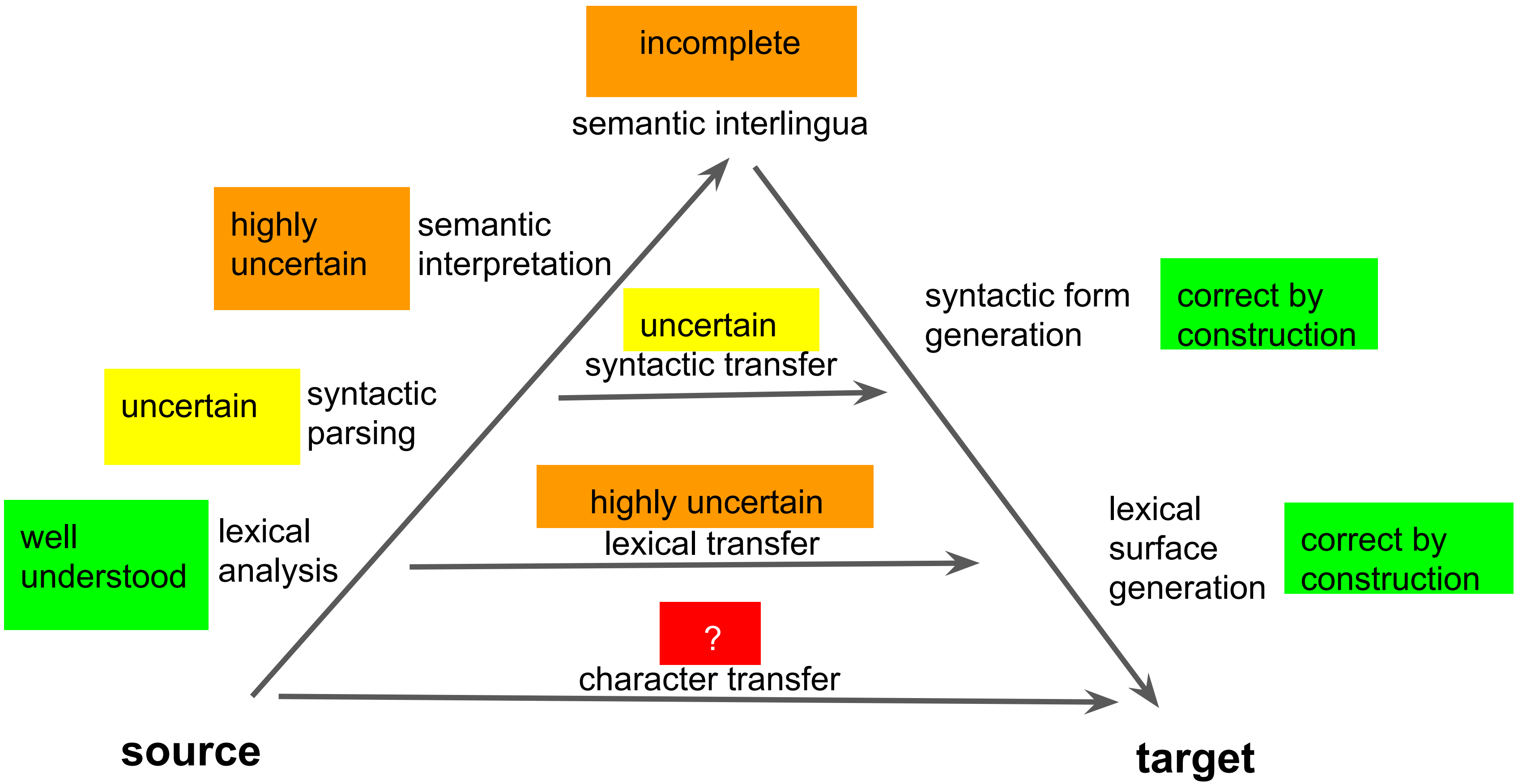
correct by construction

lexical surface generation

correct by construction

source

target



incomplete

semantic interlingua

highly uncertain

semantic interpretation

uncertain

syntactic transfer

syntactic form generation

correct by construction

uncertain

syntactic parsing

highly uncertain

lexical transfer

well understood

lexical analysis

lexical surface generation

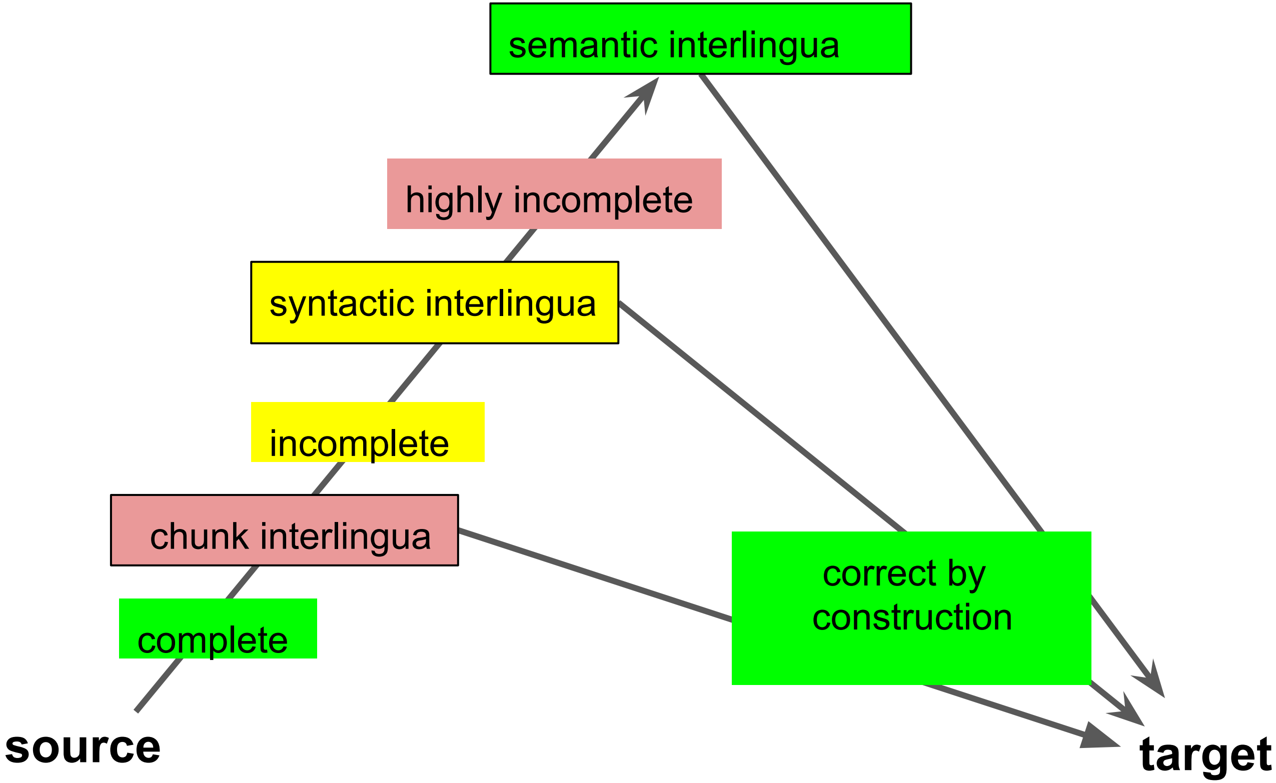
correct by construction

?

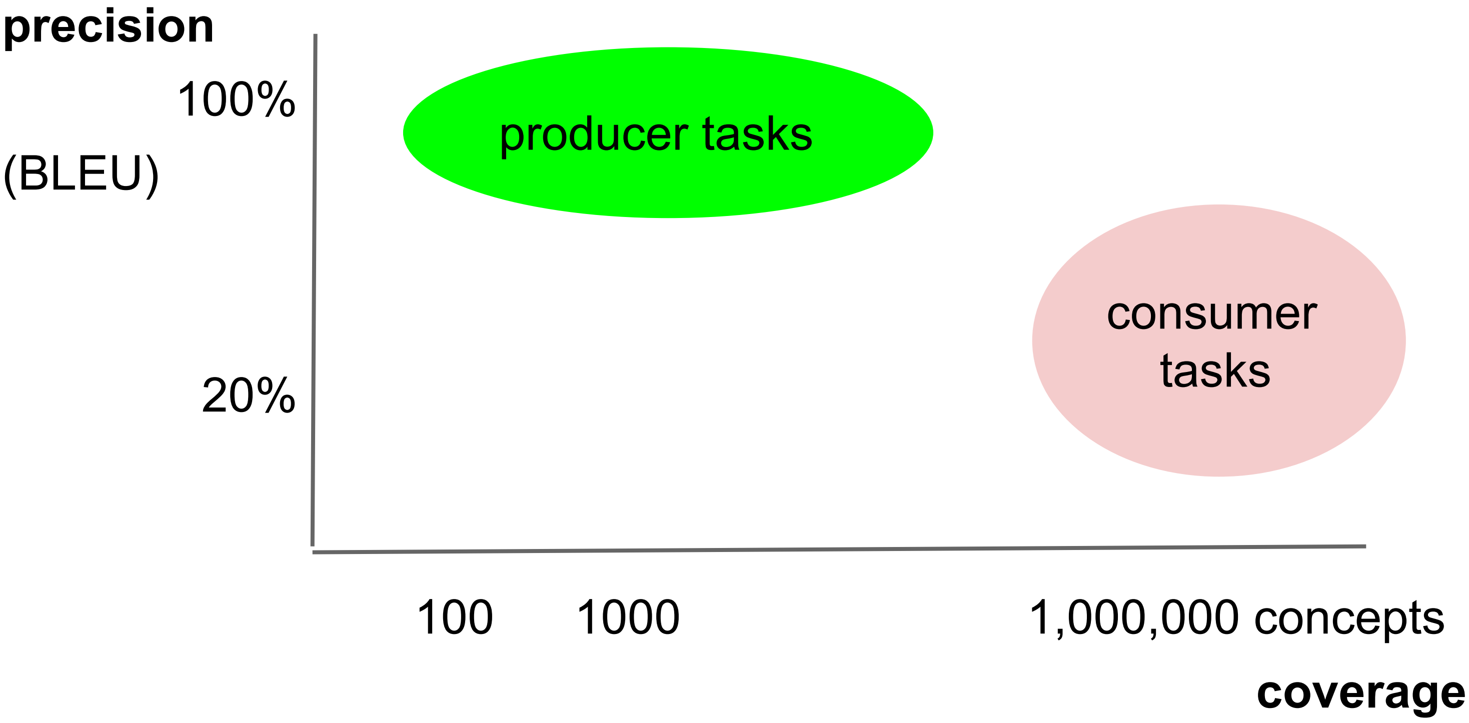
character transfer

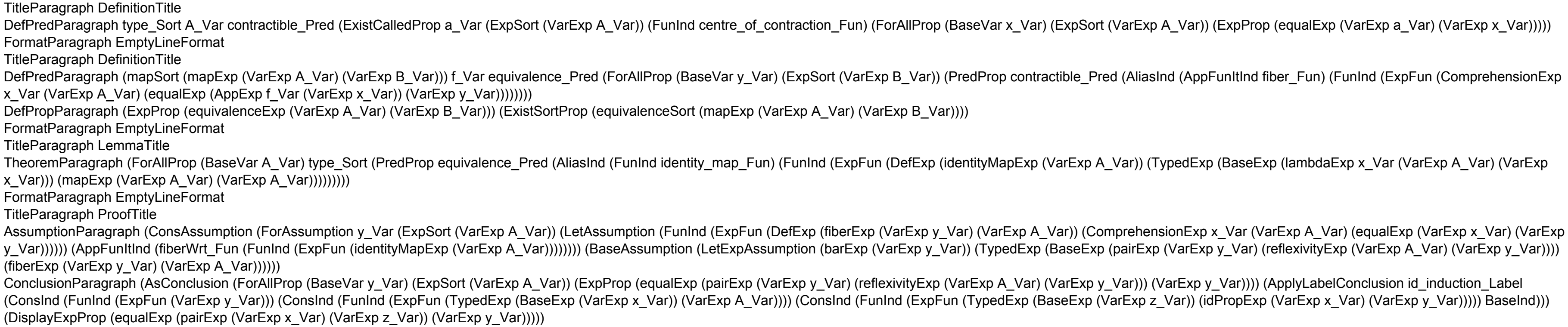
source

target

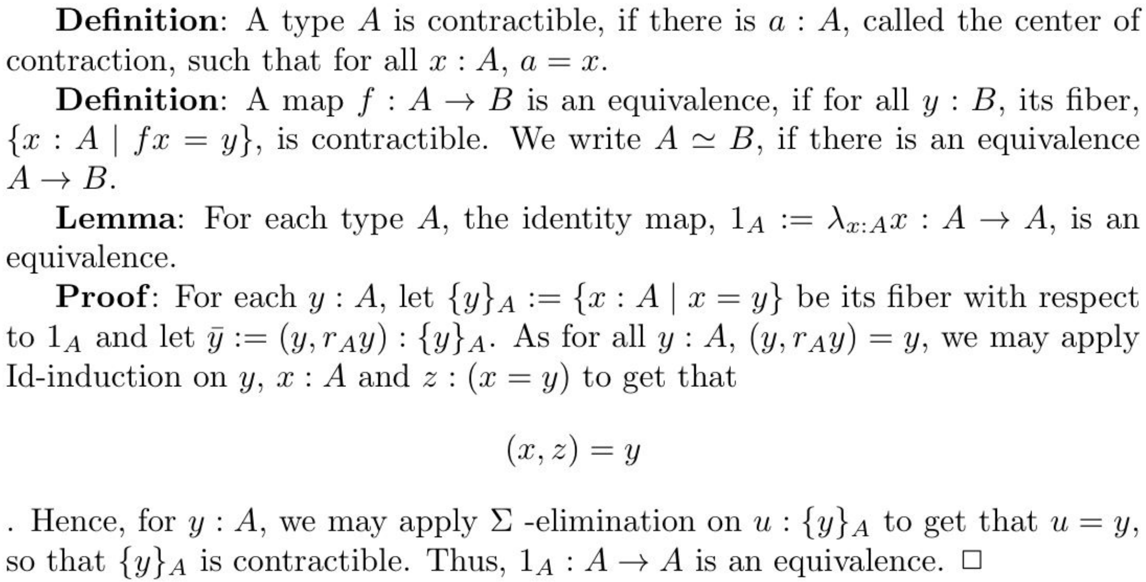

semantic interlingua
highly incomplete
syntactic interlingua
incomplete
chunk interlingua
complete
correct by construction
source
target

Producer vs. consumer task





```
TitleParagraph DefinitionTitle
DefPredParagraph type_Sort A_Var contractible_Pred (ExistCalledProp a_Var (ExpSort (VarExp A_Var)) (FunInd centre_of_contraction_Fun) (ForAllProp (BaseVar x_Var) (ExpSort (VarExp A_Var)) (ExpProp (equalExp (VarExp a_Var) (VarExp x_Var)))))
FormatParagraph EmptyLineFormat
TitleParagraph DefinitionTitle
DefPredParagraph (mapSort (mapExp (VarExp A_Var) (VarExp B_Var))) f_Var equivalence_Pred (ForAllProp (BaseVar y_Var) (ExpSort (VarExp B_Var)) (PredProp contractible_Pred (AliasInd (AppFunItInd fiber_Fun) (FunInd (ExpFun (ComprehensionExp x_Var (VarExp A_Var) (equalExp (AppExp f_Var (VarExp x_Var)) (VarExp y_Var))))))))
DefPropParagraph (ExpProp (equivalenceExp (VarExp A_Var) (VarExp B_Var))) (ExistSortProp (equivalenceSort (mapExp (VarExp A_Var) (VarExp B_Var))))
FormatParagraph EmptyLineFormat
TitleParagraph LemmaTitle
TheoremParagraph (ForAllProp (BaseVar A_Var) type_Sort (PredProp equivalence_Pred (AliasInd (FunInd identity_map_Fun) (FunInd (ExpFun (DefExp (identityMapExp (VarExp A_Var)) (TypedExp (BaseExp (lambdaExp x_Var (VarExp A_Var) (VarExp x_Var))) (mapExp (VarExp A_Var) (VarExp A_Var)))))))))
FormatParagraph EmptyLineFormat
TitleParagraph ProofTitle
AssumptionParagraph (ConsAssumption (ForAssumption y_Var (ExpSort (VarExp A_Var)) (LetAssumption (FunInd (ExpFun (DefExp (fiberExp (VarExp y_Var) (VarExp A_Var)) (ComprehensionExp x_Var (VarExp A_Var) (equalExp (VarExp x_Var) (VarExp y_Var)))))) (AppFunItInd (fiberWrt_Fun (FunInd (ExpFun (identityMapExp (VarExp A_Var)))))))) (BaseAssumption (LetExpAssumption (barExp (VarExp y_Var)) (TypedExp (BaseExp (pairExp (VarExp y_Var) (reflexivityExp (VarExp A_Var) (VarExp y_Var)))) (fiberExp (VarExp y_Var) (VarExp A_Var))))))
ConclusionParagraph (AsConclusion (ForAllProp (BaseVar y_Var) (ExpSort (VarExp A_Var)) (ExpProp (equalExp (pairExp (VarExp y_Var) (reflexivityExp (VarExp A_Var) (VarExp y_Var))) (VarExp y_Var)))) (ApplyLabelConclusion id_induction_Label (ConsInd (FunInd (ExpFun (VarExp y_Var))) (ConsInd (FunInd (ExpFun (TypedExp (BaseExp (VarExp x_Var)) (VarExp A_Var)))) (ConsInd (FunInd (ExpFun (TypedExp (BaseExp (VarExp z_Var)) (idPropExp (VarExp x_Var) (VarExp y_Var))))) BaseInd))) (DisplayExpProp (equalExp (pairExp (VarExp x_Var) (VarExp z_Var)) (VarExp y_Var)))))
ConclusionSoThatParagraph (ForConclusion (BaseVar y_Var) (ExpSort (VarExp A_Var)) (Ap
BaseInd) (ExpProp (equalExp (VarExp u_Var) (VarExp y_Var))))) (PredProp contractible_Pre
ConclusionParagraph (PropConclusion (PredProp equivalence_Pred (FunInd (ExpFun (Type
QEDParagraph
```



Definition: A type A is contractible, if there is $a : A$, called the center of contraction, such that for all $x : A$, $a = x$.

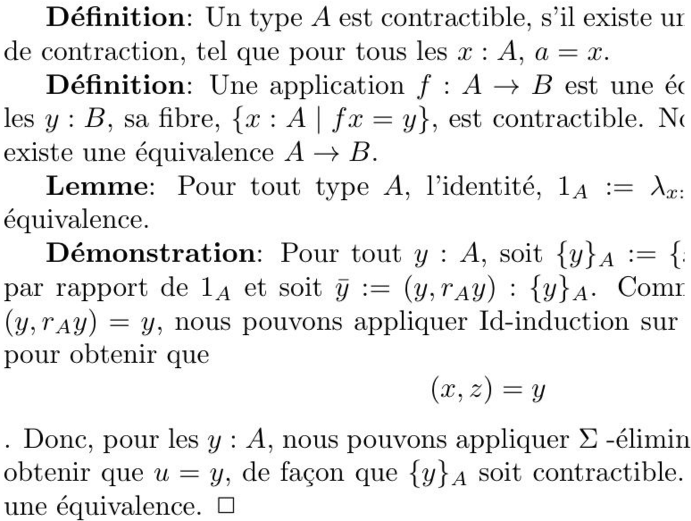
Definition: A map $f : A \to B$ is an equivalence, if for all $y : B$, its fiber, $\{x : A \mid fx = y\}$, is contractible. We write $A \simeq B$, if there is an equivalence $A \to B$.

Lemma: For each type A, the identity map, $1_A := \lambda_{x:A} x : A \to A$, is an equivalence.

Proof: For each $y : A$, let $\{y\}_A := \{x : A \mid x = y\}$ be its fiber with respect to 1_A and let $\bar{y} := (y, r_A y) : \{y\}_A$. As for all $y : A$, $(y, r_A y) = y$, we may apply Id-induction on y, $x : A$ and $z : (x = y)$ to get that

$$(x, z) = y$$

. Hence, for $y : A$, we may apply Σ -elimination on $u : \{y\}_A$ to get that $u = y$, so that $\{y\}_A$ is contractible. Thus, $1_A : A \to A$ is an equivalence. □



Définition: Un type A est contractible, s'il existe ur
de contraction, tel que pour tous les $x : A$, $a = x$.

Définition: Une application $f : A \to B$ est une éc
les $y : B$, sa fibre, $\{x : A \mid fx = y\}$, est contractible. Nc
existe une équivalence $A \to B$.

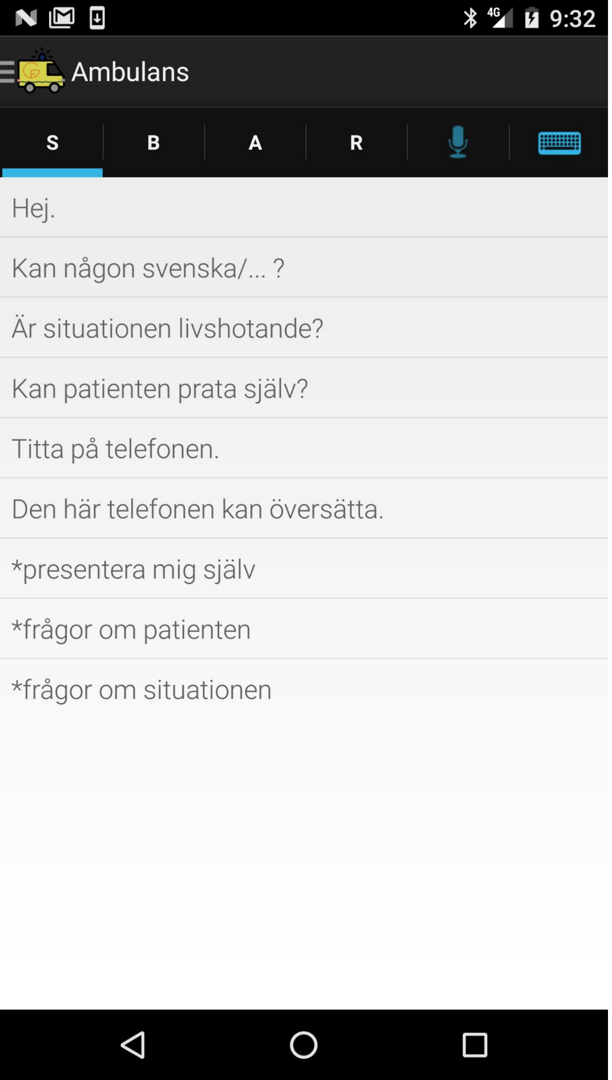
Lemme: Pour tout type A, l'identité, $1_A := \lambda_{x:}$
équivalence.

Démonstration: Pour tout $y : A$, soit $\{y\}_A := \{$
par rapport de 1_A et soit $\bar{y} := (y, r_A y) : \{y\}_A$. Comn
$(y, r_A y) = y$, nous pouvons appliquer Id-induction sur
pour obtenir que

$$(x, z) = y$$

. Donc, pour les $y : A$, nous pouvons appliquer Σ -élimination sur $u : \{y\}_A$ pour obtenir que $u = y$, de façon que $\{y\}_A$ soit contractible. Alors, $1_A : A \to A$ est une équivalence. □

https://github.com/GrammaticalFramework/gf-contrib/tree/master/homotopy-typetheory



Hej.

Kan någon svenska/... ?

Är situationen livshotande?

Kan patienten prata själv?

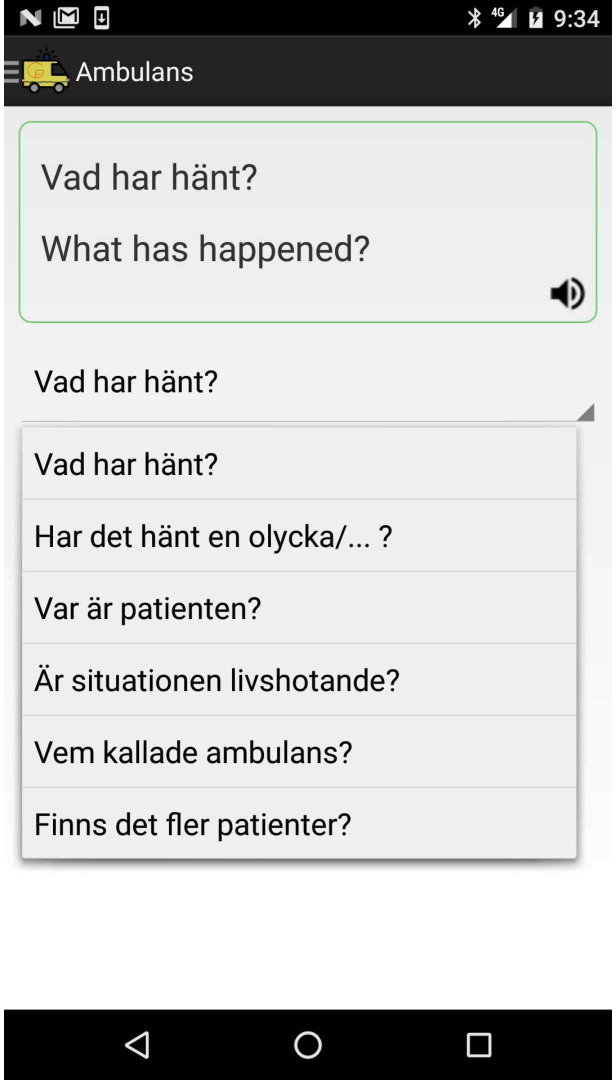
Titta på telefonen.

Den här telefonen kan översätta.

*presentera mig själv

*frågor om patienten

*frågor om situationen



Vad har hänt?

What has happened?

Vad har hänt?

Vad har hänt?

Har det hänt en olycka/... ?

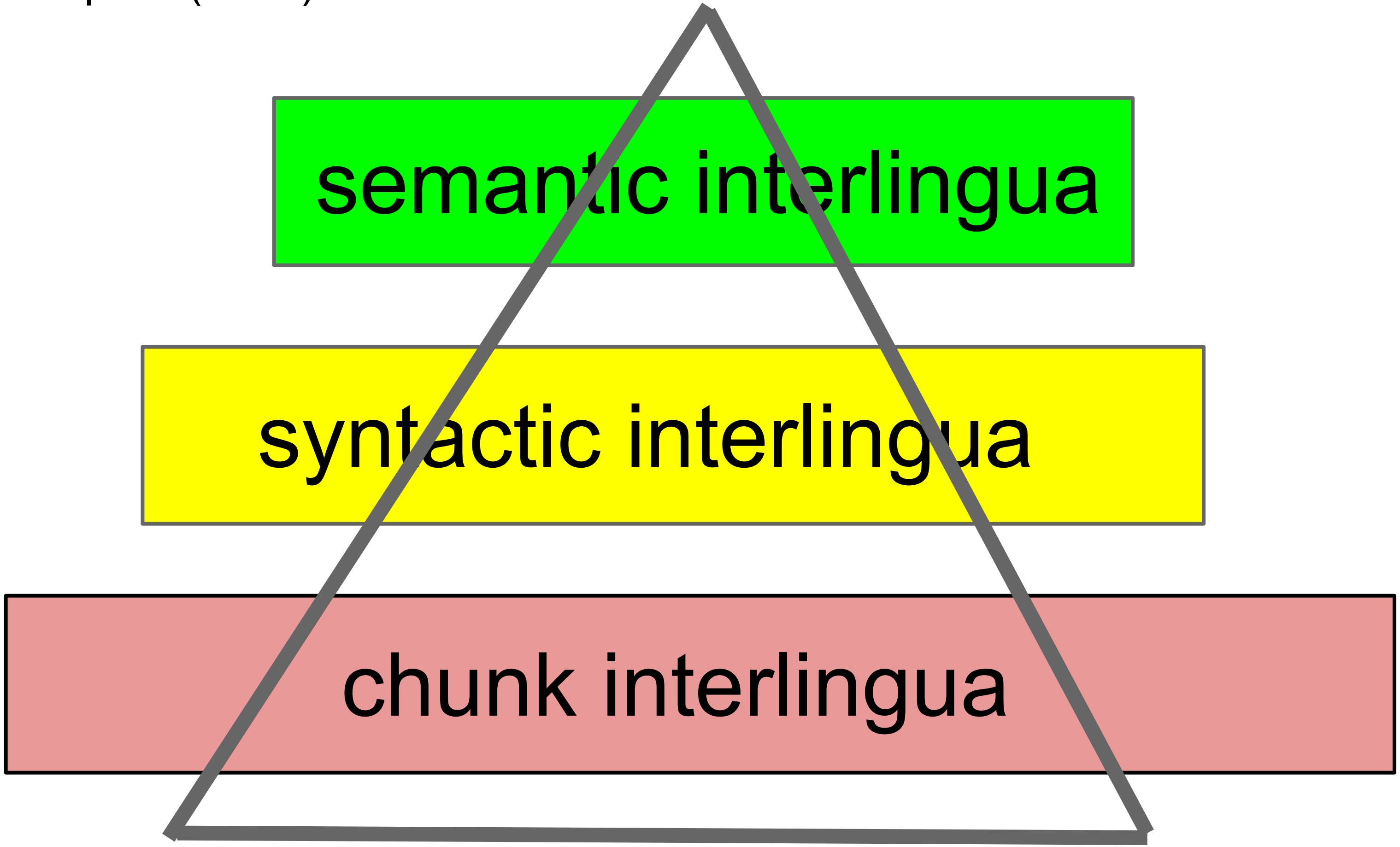
Var är patienten?

Är situationen livshotande?

Vem kallade ambulans?

Finns det fler patienter?

Cf. Vauquois (1968)

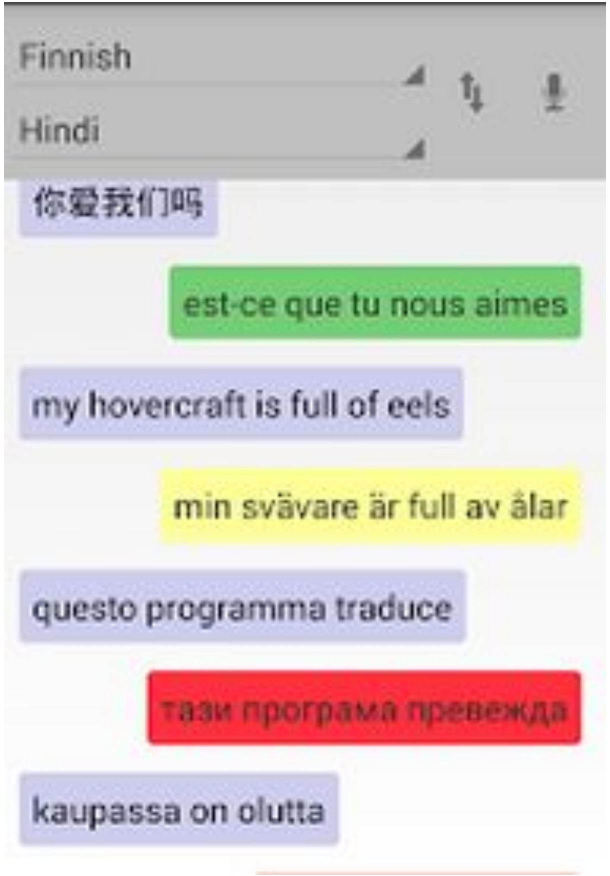


semantic interlingua

syntactic interlingua

chunk interlingua

GF Offline Translator



你爱我们吗

est-ce que tu nous aimes

my hovercraft is full of eels

min svävare är full av ålar

questo programma traduce

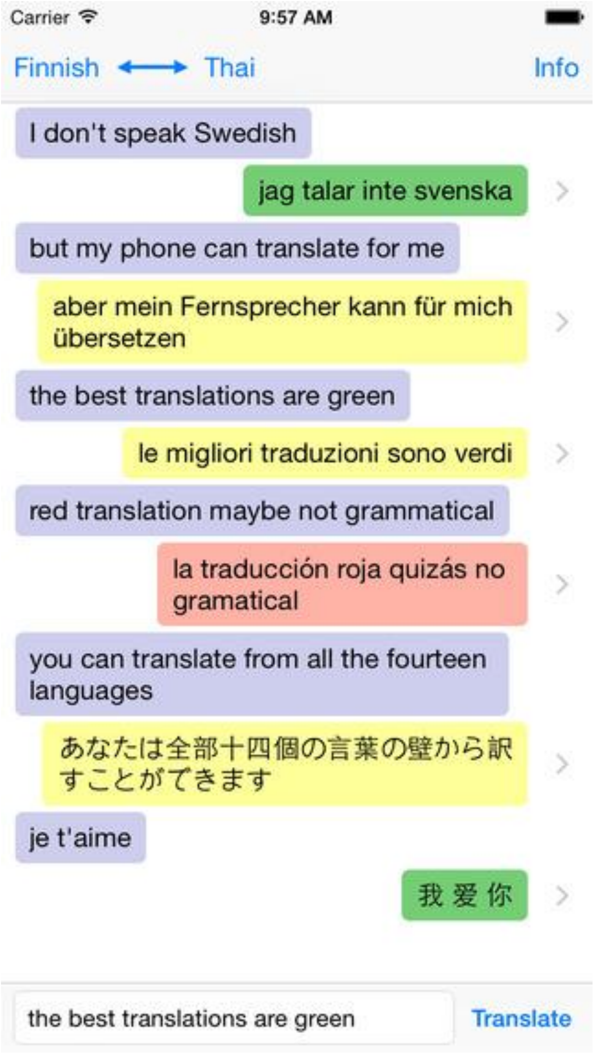
тази програма превежда

kaupassa on olutta

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.



I don't speak Swedish

jag talar inte svenska

but my phone can translate for me

aber mein Fernsprecher kann für mich übersetzen

the best translations are green

le migliori traduzioni sono verdi

red translation maybe not grammatical

la traducción roja quizás no gramatical

you can translate from all the fourteen languages

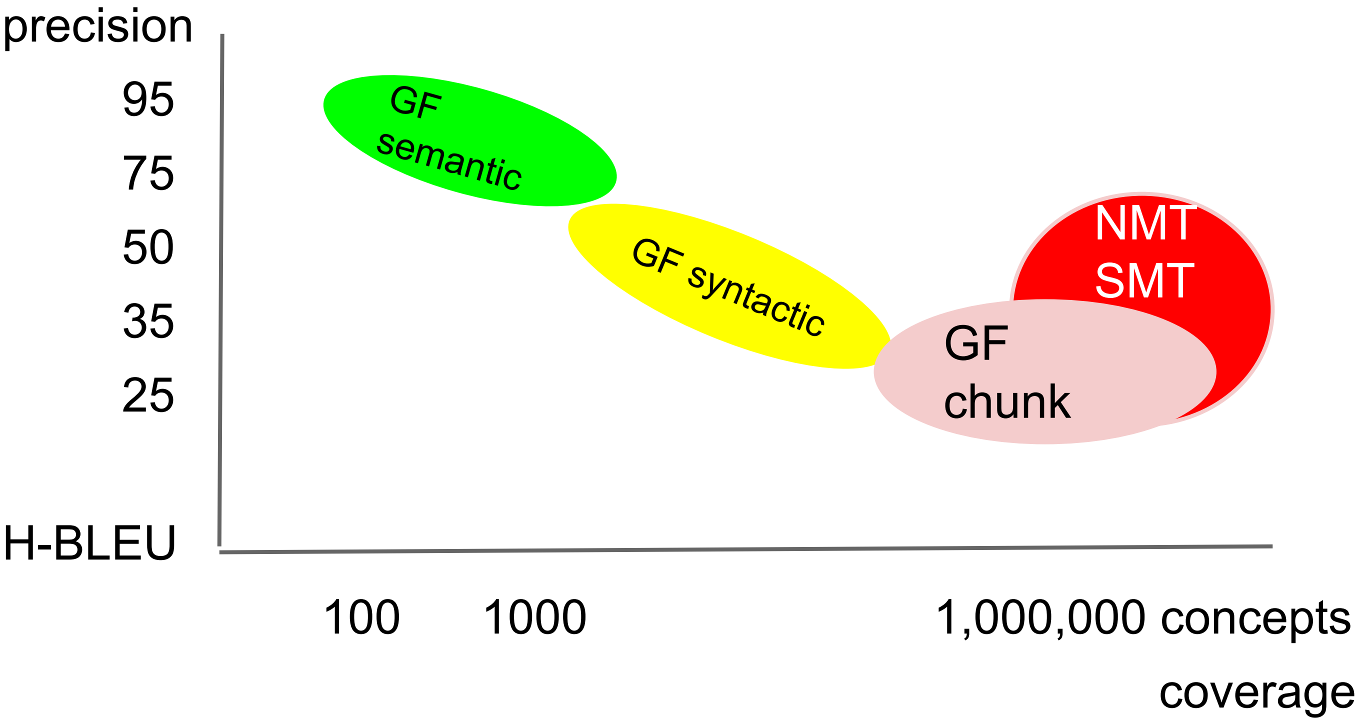
あなたは全部十四個の言葉の壁から訳すことができます

je t'aime

我 爱 你

the best translations are green

Quality degradation

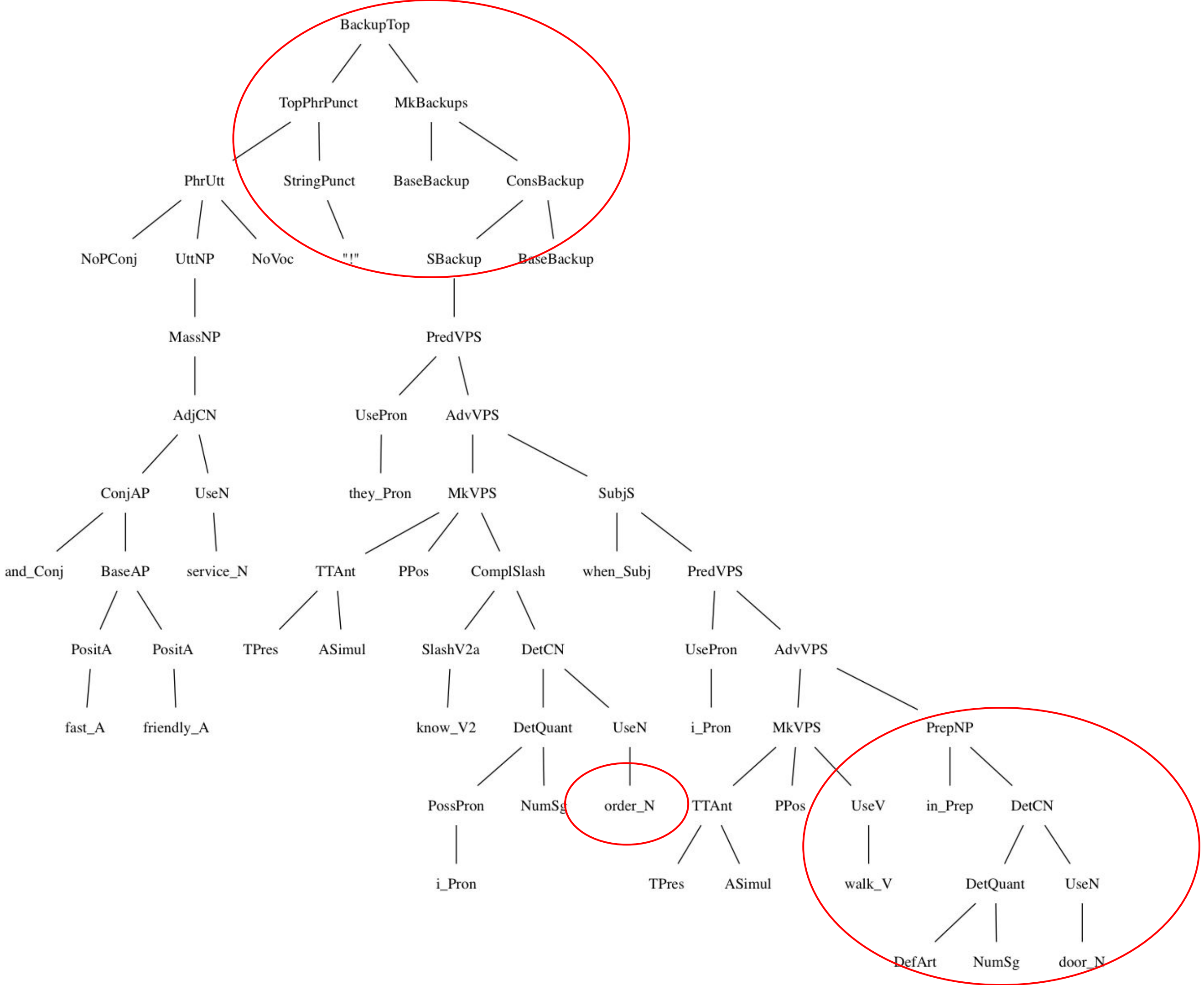


STRING: Fast and friendly service , they know my order when I walk in the door !

```
root NOUN service_N : N [] {} (4) 4
    amod ADJ fast_A : A [] {} (1) 1
        cc CONJ "and" : Conjand_ [and_Conj : Conj] {} (2) 2
        conj ADJ friendly_A : A [] {} (3) 3
    punct PUNCT "," : Comma_ [] {} (5) 5
    parataxis VERB know_VQ : VQ [know_VS : VS, know_V2 : V2, know_V : V] {} (7) 7
        nsubj PRON they_Pron : Pron [theyFem_Pron : Pron] {} (6) 6
        dobj NOUN order_N : N [] {} (9) 9
            nmod:poss PRON i_Pron : Pron [] {} (8) 8
        advcl VERB walk_V2 : V2 [walk_V : V] {} (12) 12
            mark ADV when_Subj : Subj [when_IAdv : IAdv] {} (10) 10
            nsubj PRON i_Pron : Pron [iFem_Pron : Pron] {} (11) 11
            nmod NOUN door_N : N [] {} (15) 15
                case ADP in_Prep : Prep [] {} (13) 13
                det DET DefArt : Quant [] {} (14) 14
    punct PUNCT StringPN "!" : PN [StringPunct "!" : Punct] {} (16) 16
```

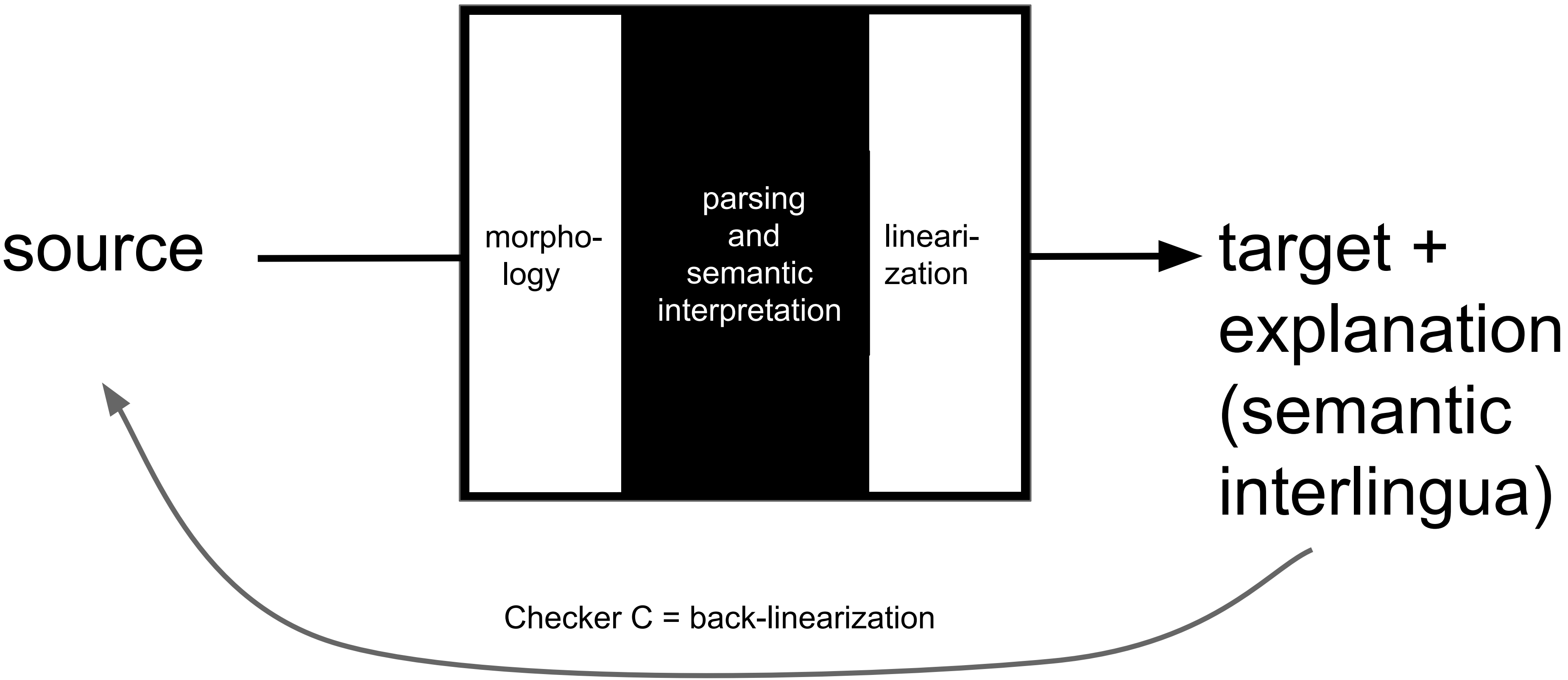
```
Eng: fast and friendly service "!" [ they know my order when I walk in the door ]
Fin: nopea ja ystävällinen palvelu "!" [ he tuntevat minun järjestykseni kun minä kävelen ovessa ]
Swe: snabb och vänlig tjänst "!" [ de känner min ordning när jag går i dörren ]
```

```
PARSED: 16/16   WITHOUT BACKUP: 6/16
```


BackupTop
TopPhrPunct
MkBackups
PhrUtt
StringPunct
BaseBackup
ConsBackup
NoPConj
UttNP
NoVoc
"!"
SBackup
BaseBackup
MassNP
PredVPS
AdjCN
UsePron
AdvVPS
ConjAP
UseN
they_Pron
MkVPS
SubjS
and_Conj
BaseAP
service_N
TTAnt
PPos
ComplSlash
when_Subj
PredVPS
PositA
PositA
TPres
ASimul
SlashV2a
DetCN
UsePron
AdvVPS
fast_A
friendly_A
know_V2
DetQuant
UseN
i_Pron
MkVPS
PrepNP
PossPron
NumSg
order_N
TTAnt
PPos
UseV
in_Prep
DetCN
i_Pron
TPres
ASimul
walk_V
DetQuant
UseN
DefArt
NumSg
door_N

Take home

XMT



source

morpho-logy

parsing and semantic interpretation

lineari-zation

target + explanation (semantic interlingua)

Checker C = back-linearization