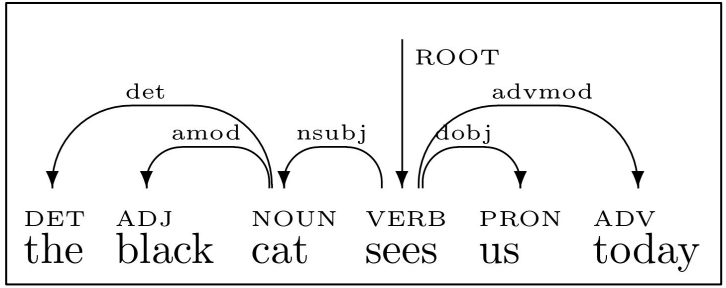
GF2UD and UD2GF
UD: *Universal* Dependencies

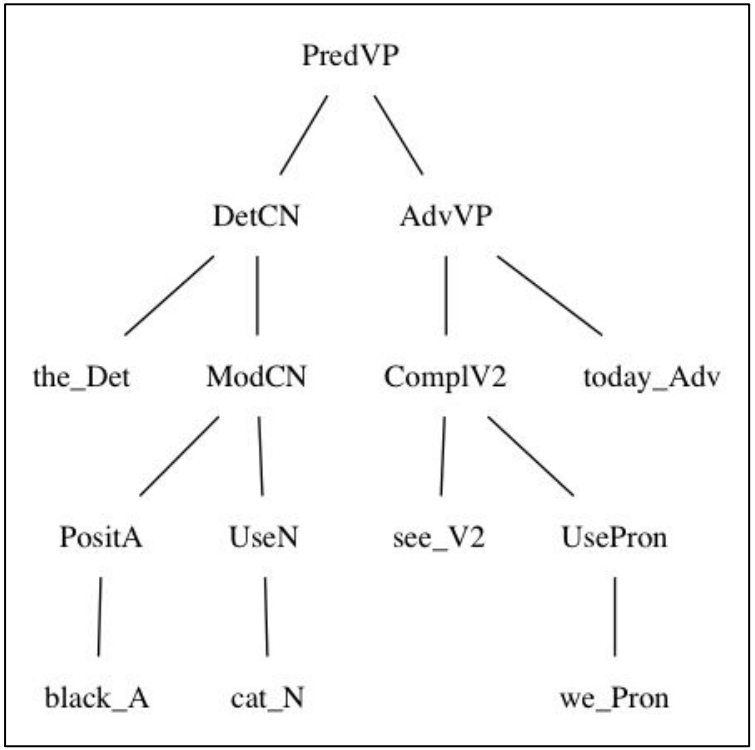
Prasanth Kolachina
GF Summer school, 2017

the black cat sees us today

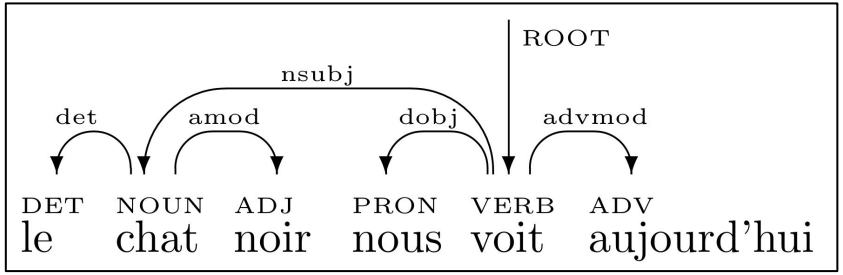
dependency parser



ud2gf



gf2ud



GF

le chat noir nous voit aujourd'hui

Universal Dependencies

Principles of Design

- UD needs to be satisfactory on linguistic analysis grounds for individual languages.
- UD needs to be good for linguistic typology, i.e., providing a suitable basis for bringing out cross-linguistic parallelism across languages and language families.
- UD must be suitable for rapid, consistent annotation by a human annotator.
- UD must be suitable for computer parsing with high accuracy.
- UD must be easily comprehended and used by a non-linguist …. (API grammar)
- UD must support well downstream language understanding tasks (relation extraction, reading comprehension, machine translation, ...).

Mission of Grammatical Framework

The mission of GF is to formalize

the grammars of the world

and

make them available for computer applications.

Universal Dependencies

A community-driven effort to annotate multilingual treebanks

Cross-lingual consistency in annotations across languages

17 Part-of-Speech tags ; 40 dependency labels ; morphological features

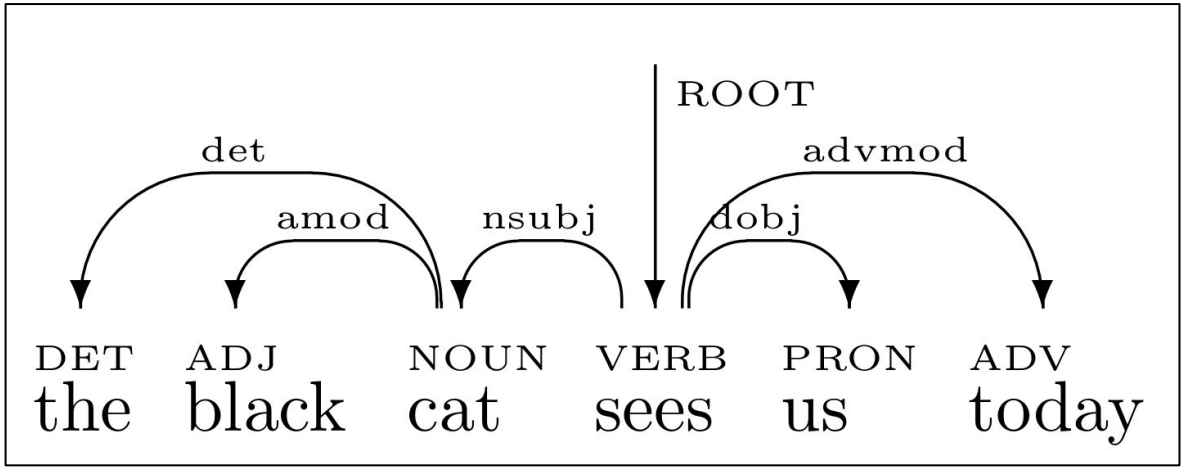
Annotated corpora released every 6 months;

Ongoing V2

50 Languages, 70 Treebanks

Afrikaans	49K	LF	-	□	□			□□
Ancient Greek	202K	LF	□	□	□			□
Ancient Greek-PROIEL	211K	LF	-	□	□			□□
Arabic	242K	LF	-	□	□			□
Arabic-NYUAD	629K	LF	-	□	□			□
Arabic-PUD	20K	LF	-	□	□			□□
Basque	121K	LF	□	□	□			□□
Belarusian	8K	LF	-	□	□			□
Bulgarian	156K	LF	□	□□	□			□□□□
Buryat	10K	LF	-	□	□			□□□
Catalan	530K	LF	□	□□	□			□
Chinese	123K	LF	□	□□	□			□
Chinese-CFL	7K	L	□	□	□			□
Chinese-PUD	21K	F	-	□	□			□□
Coptic	3K	LF	□	□	□			□□□
Croatian	197K	LF	-	□□	□			□□□
Czech	1,503K	LF	□	□□	□			□
Czech-CAC	493K	LF	□	□□	□			□□□□□
Czech-CLTT	37K	LF	□	□□	□			□
Czech-FicTree	166K	LF	□	□□	□			□
Czech-PUD	18K	LF	□	□	□			□□
Danish	100K	LF	□	□□	□			□□□□
Dutch	209K	LF	-	□	□			□
Dutch-LassySmall	101K	LF	-	□	□			□
English	254K	LFD	□	□	□			□□□
English-ESL	88K	L	□	□	□			□
English-LinES	82K		□	□□	□			□□□
English-PUD	21K	LF	□	□	□			□□
English-ParTUT	49K	LF	□	□□	□			□□□
Estonian	47K	LF	-	□□	□			□□□
Finnish	202K	LFD	□	□□	□			□□□□□□
Finnish-FTB	159K	LF	-	□□	□			□
Finnish-PUD	15K	LFD	-	□	□			□□
French	391K	LF	□	□□	□			□□□□
French-FTB	556K	LF	-	□	□			□□
French-PUD	24K	F	-	□	□			□□
French-ParTUT	27K	LF	□	□□	□			□□□
French-Sequoia	68K	LF	-	□	□			□□□□
Galician	138K	LF	□	□□	□			□□□□

Galician-TreeGal	23K	LF	□	□	□			□
German	287K	LF	-	□	□			□□□
German-PUD	20K	F	-	□	□			□□
Gothic	55K	LF	-	□	□			□
Greek	61K	LF	□	□□	□			□□□
Hebrew	115K	LF	-	□	□			□
Hindi	351K	LF	-	□	□			□
Hindi-PUD	23K	F	-	□	□			□□
Hungarian	42K	LF	□	□	□			□
Indonesian	121K	F	-	□	□			□□
Indonesian-PUD	25K	F	-	□	□			□□
Irish	23K	LF	□	□□	□			□□□□□
Italian	273K	LF	□	□□	□			□□□
Italian-PUD	22K	LF	-	□	□			□□
Italian-ParTUT	39K	LF	□	□□	□			□□□
Japanese	186K	LF	□	□	□			□□
Japanese-KTC	189K	L	□	□	□			□
Japanese-PUD	26K	LF	-	□	□			□□
Kazakh	10K	LF	□	□	□			□□
Korean	63K	F	□	□	□			□□
Korean-PUD	22K	F	-	□	□			□□
Korean-Sejong	89K	L	-	□	?			□
Kurmanji	10K	LF	-	□	□			□□
Latin	29K	LF	□	□	□			□□□
Latin-ITTB	291K	LF	-	□	□			□
Latin-PROIEL	171K	LF	-	□	□			□□
Latvian	54K	LF	-	□	□			□□□
Lithuanian	5K	LF	-	□	□			□□
Maltese	2K	F	-	□	□			□□
North Sami	55K	LF	-	□	□			□□
Norwegian-Bokmaal	310K	LF	□	□	□			□□□
Norwegian-Nynorsk	301K	LF	□	□	□			□□□
Old Church Slavonic	57K	LF	-	□	□			□
Persian	151K	LF	□	□□	□			□□□□□□□
Polish	82K	LF	-	□	□			□□□
Portuguese	210K	LF	□	□□	□			□□
Portuguese-BR	297K	F	-	□	□			□□
Portuguese-PUD	21K	F	-	□	□			□□
Romanian	218K	LF	□	□□	□			□□□□□□□
Russian	99K	LF	□	□□	□			□
Russian-PUD	19K	F	-	□	□			□□
Russian-SynTagRus	1,107K	LFD	□	□□	□			□□□
Sanskrit	1K	LF	-	□	□			□
Serbian	86K	LF	-	□□	□			□□


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

Clausal Predicates

nsubj csubj

dobj iobj
ccomp xcomp

Coordination

conj cc punct

Passive voice

nsubjpass csubjpass

auxpass

Adverbials

advmod nmod
advcl

Copulas and special marker

mark cop

Auxiliary verbs and negation

aux neg

Noun dependents

det nummod
amod appos
neg nmod

acl

case

Compounding

compound mwe
name

Other

root dep

Unknowns

list dislocated parataxis remnant reparandum

Clausal Predicates

nsubj

dobj iobj

Copulas

cop

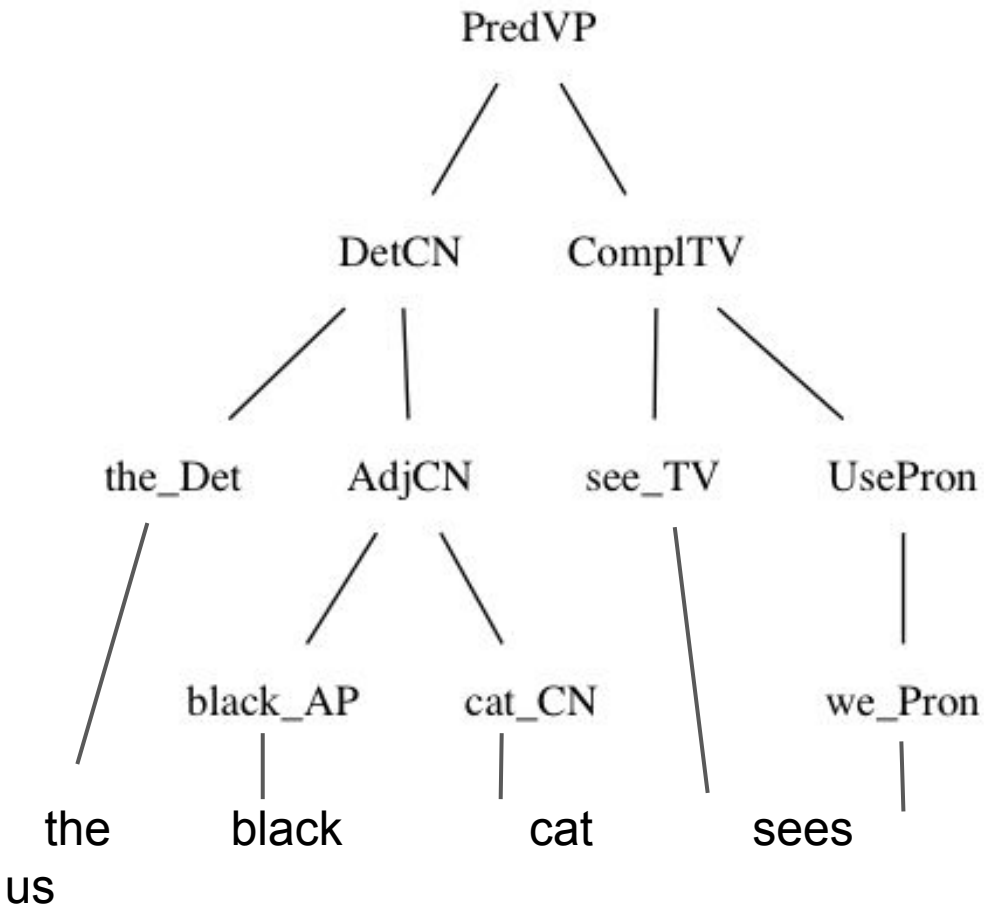
Auxiliary verbs and negation

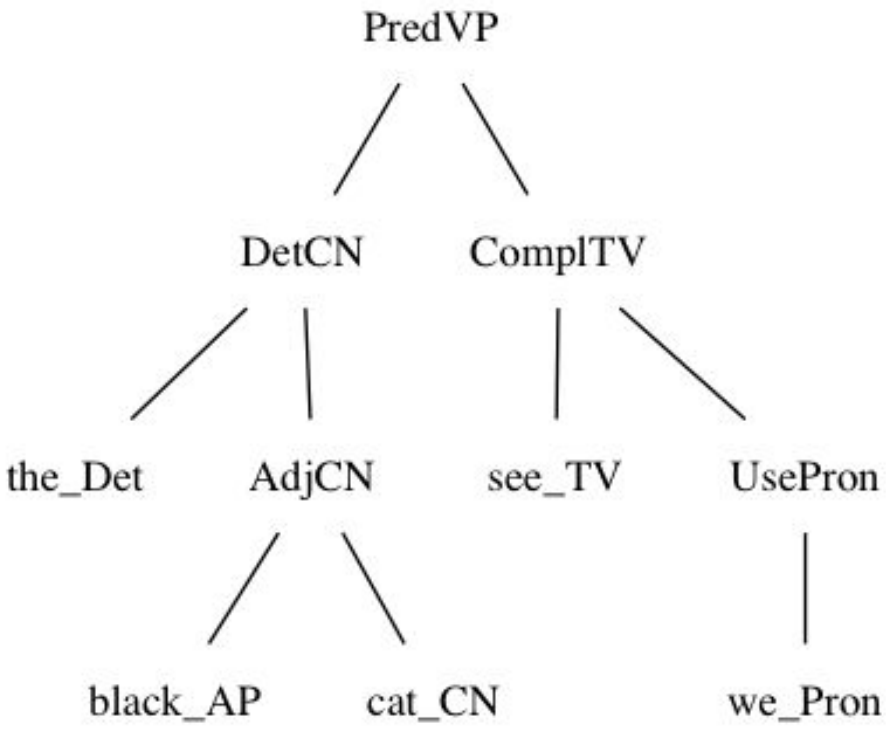
aux neg

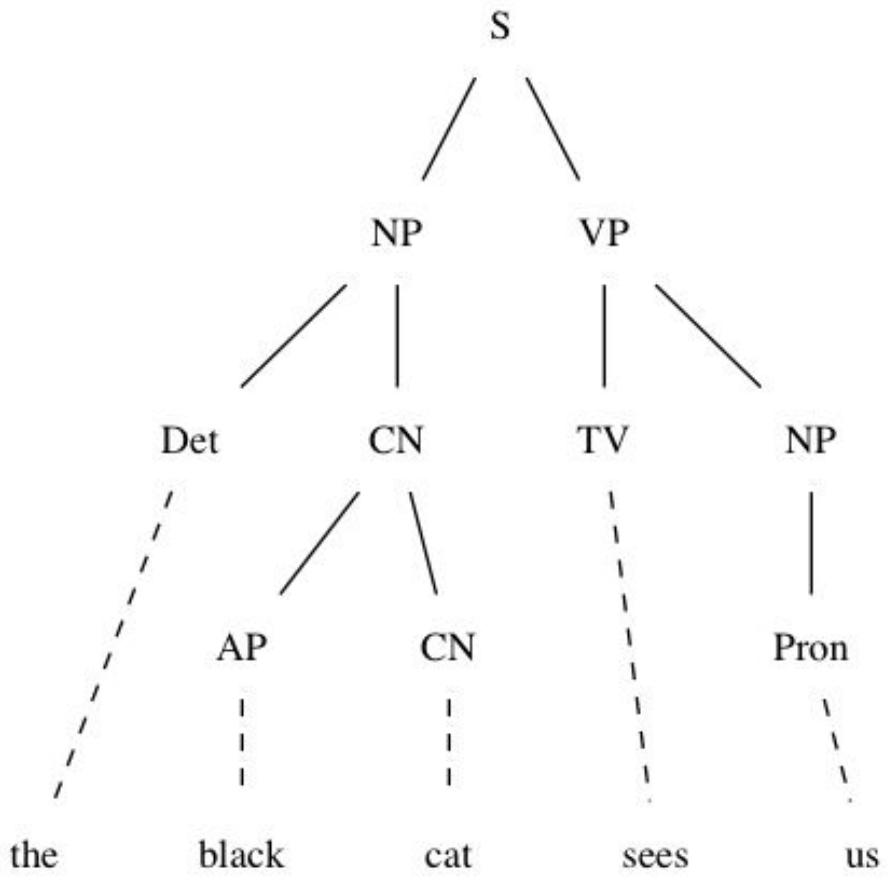
Noun dependents

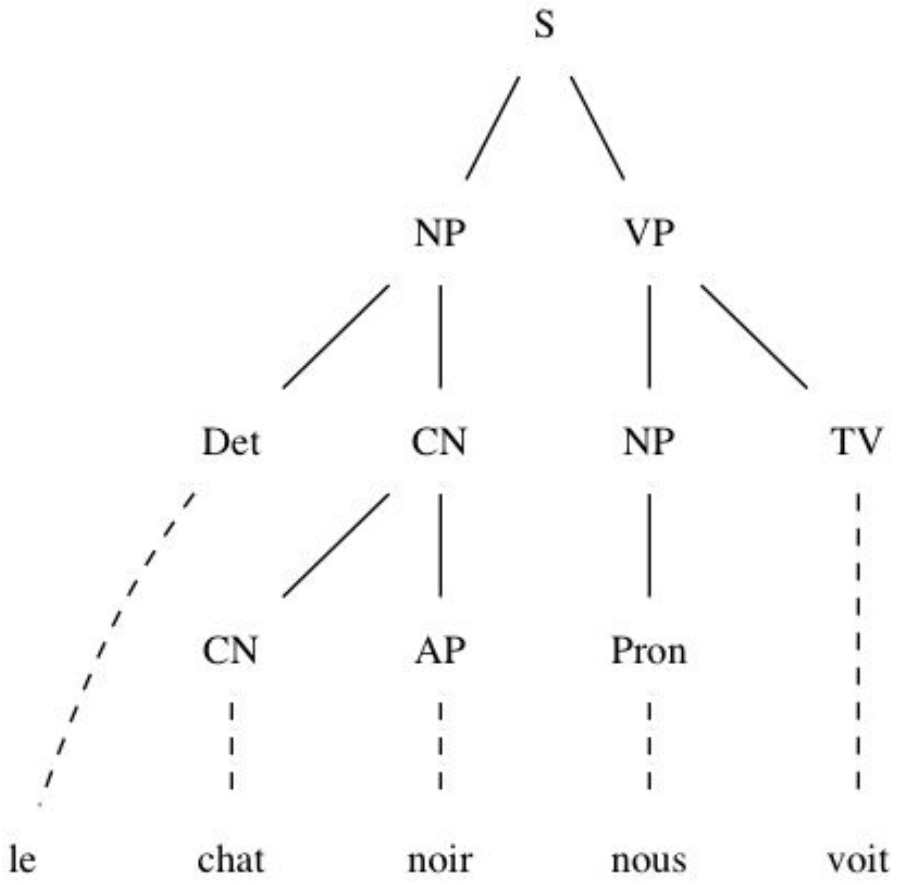
det amod

Structures in GF


PredVP
DetCN
ComplTV
the_Det
AdjCN
see_TV
UsePron
black_AP
cat_CN
we_Pron
the
black
cat
sees
us


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
the
black
cat
sees
us

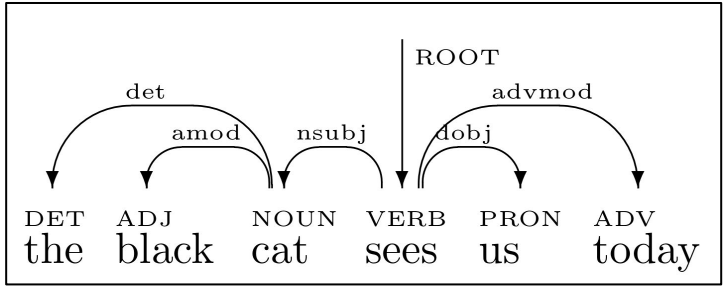

S
NP
VP
Det
CN
NP
TV
CN
AP
Pron
le
chat
noir
nous
voit

Rationale

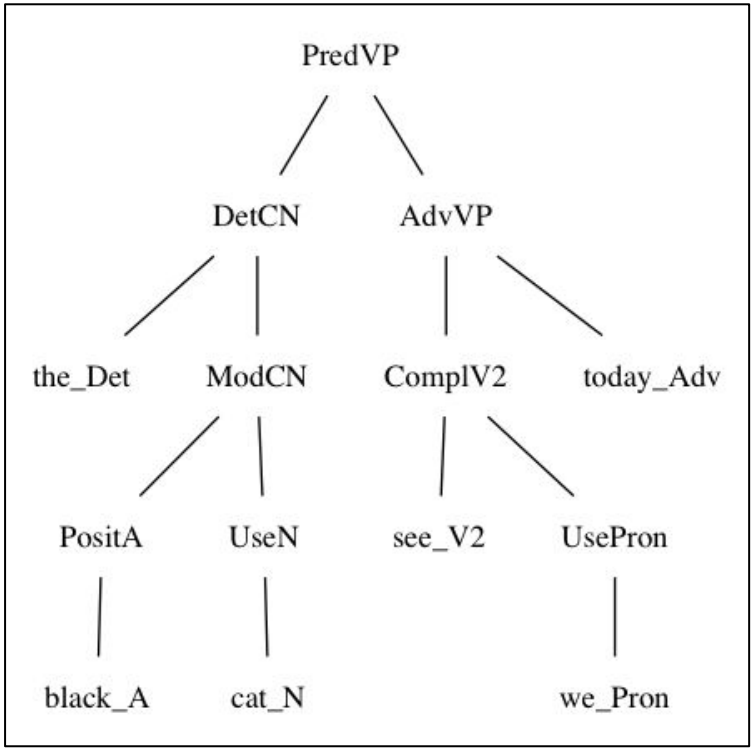
	dependencies	GF
parsing robustness	**robust**	brittle
parsing speed	**fast**	slow
semantics	loose	**compositional**
generation	?	**accurate**

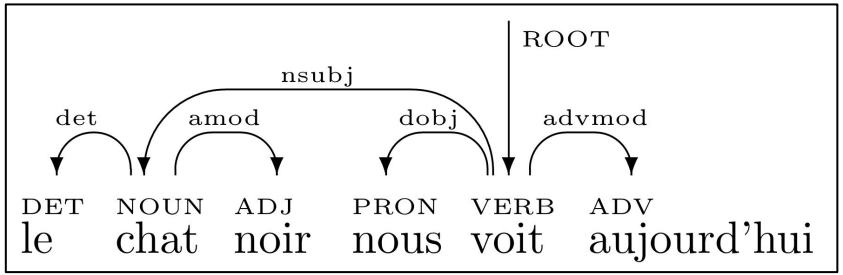
the black cat sees us today

dependency parser



ud2gf





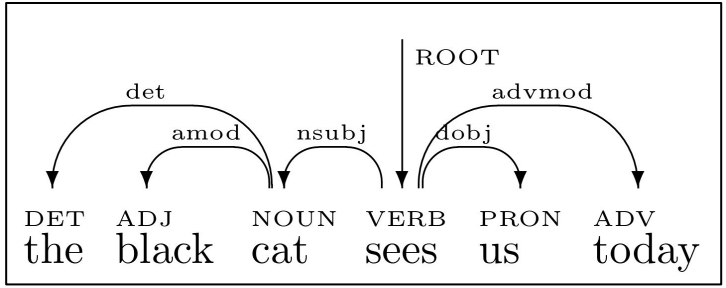
gf2ud

GF

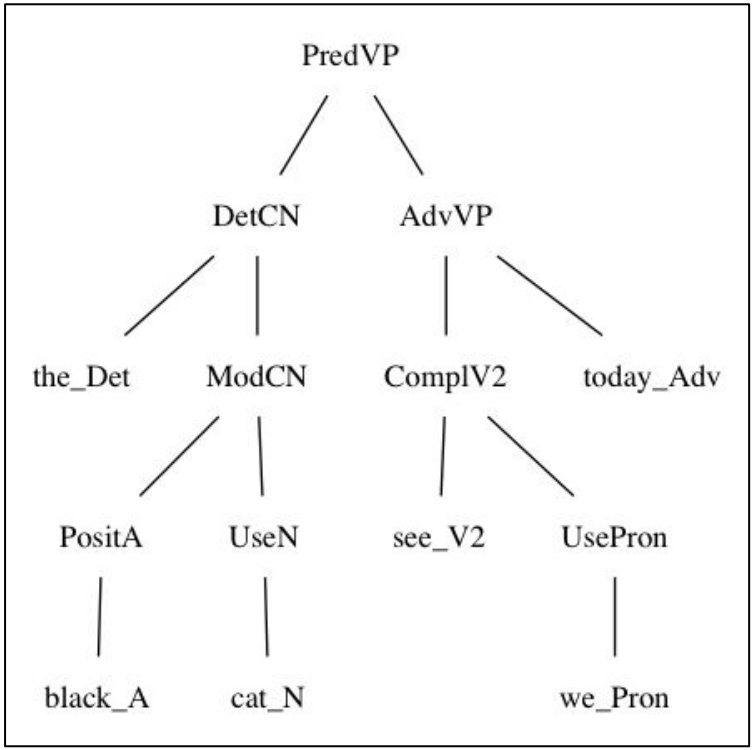
le chat noir nous voit aujourd'hui

the black cat sees us today

dependency parser



ud2gf



sem

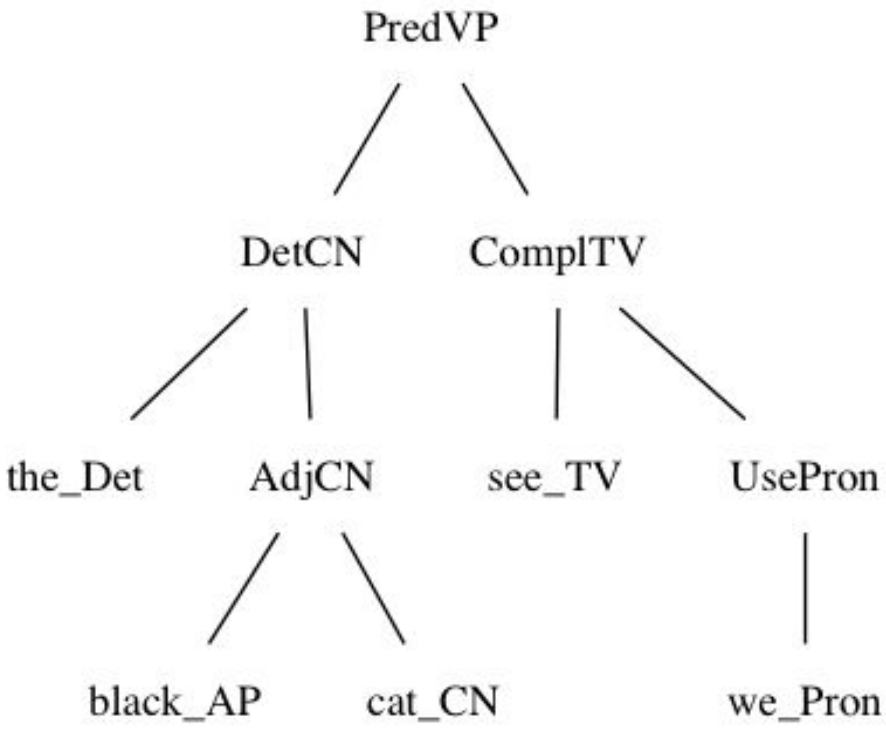
GF

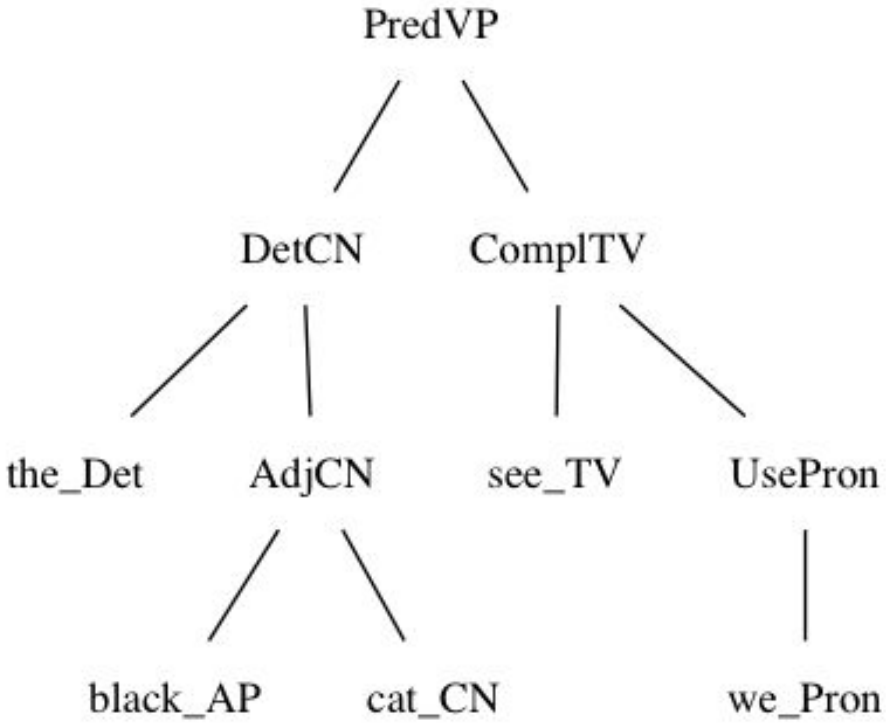
$\exists$!A.(cat(A) & MODIFIER(black,A)&
($\exists$ B.(see(B) & SUBJECT(B)=A &
OBJECT(B) = we &
MODIFIER(today,B))))

le chat noir nous voit aujourd'hui

GF2UD

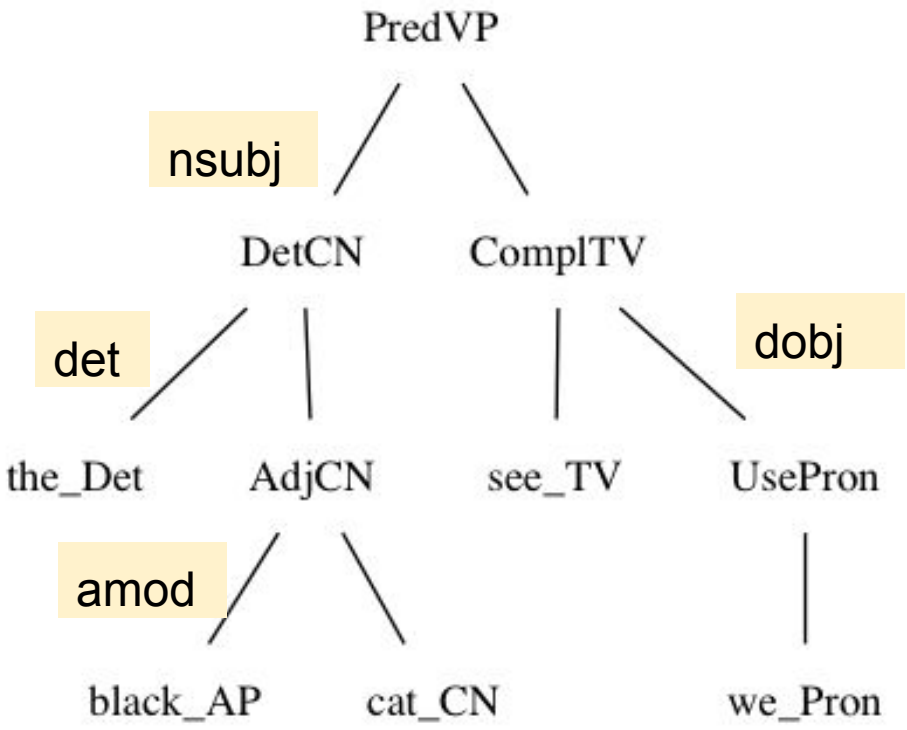
grammatical roles to arguments and hide functions


PredVP
DetCN
ComplTV
the_Det
AdjCN
see_TV
UsePron
black_AP
cat_CN
we_Pron



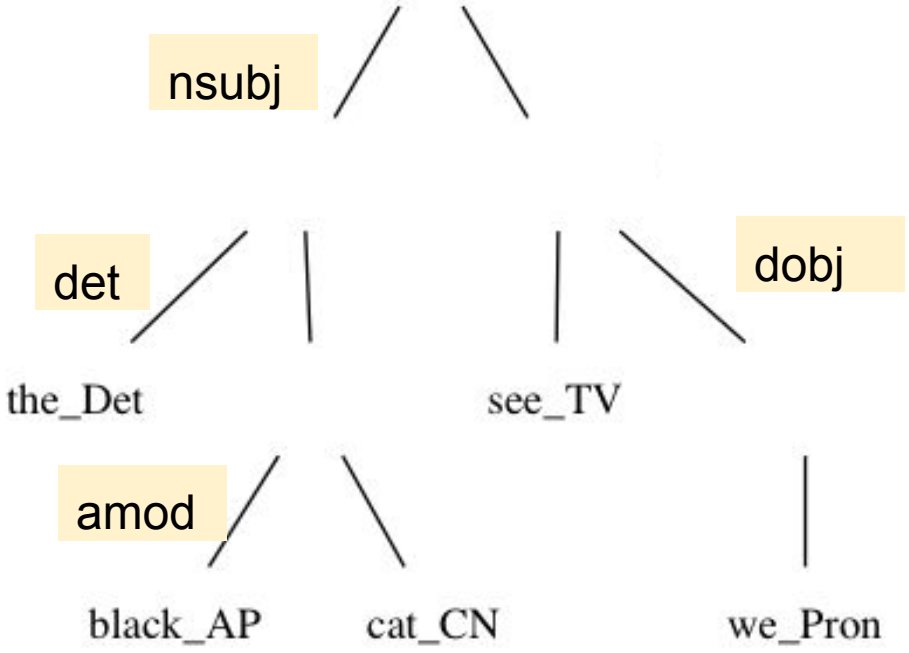
Dependency configuration

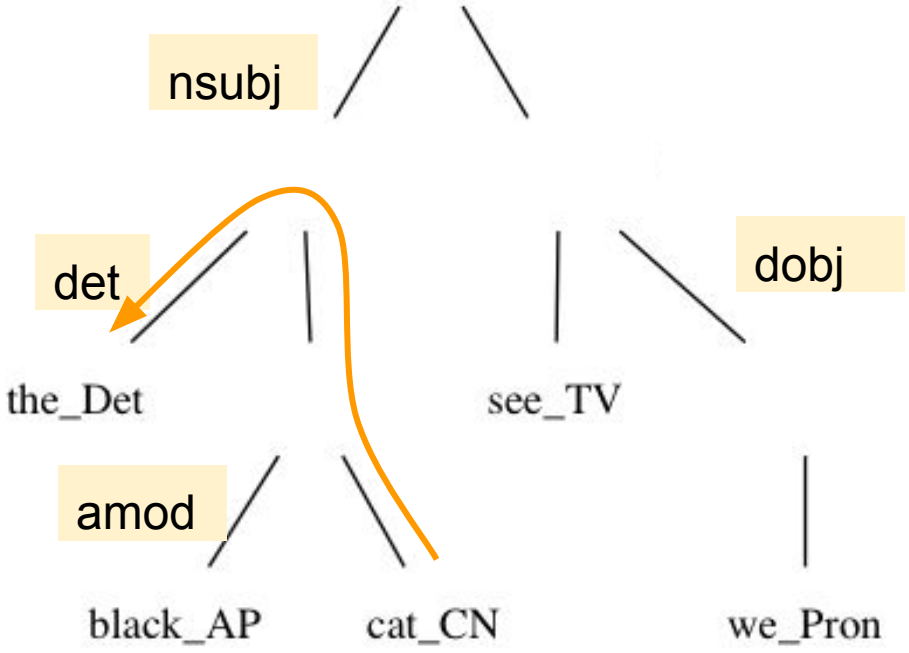
```
PredVP  nsubj head
ComplTV  head  dobj
DetCN   det   head
AdjCN  amod  head
```

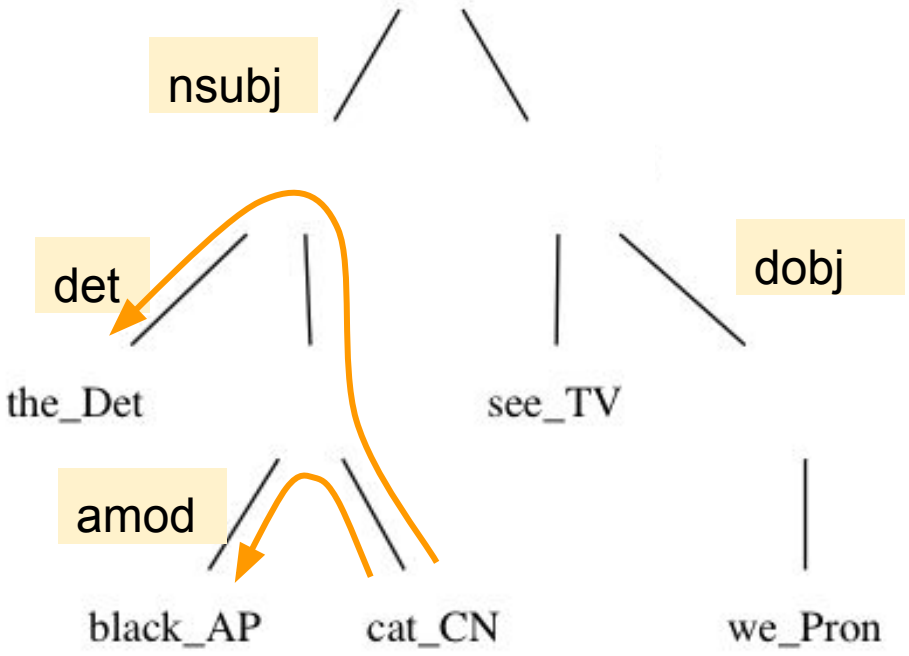


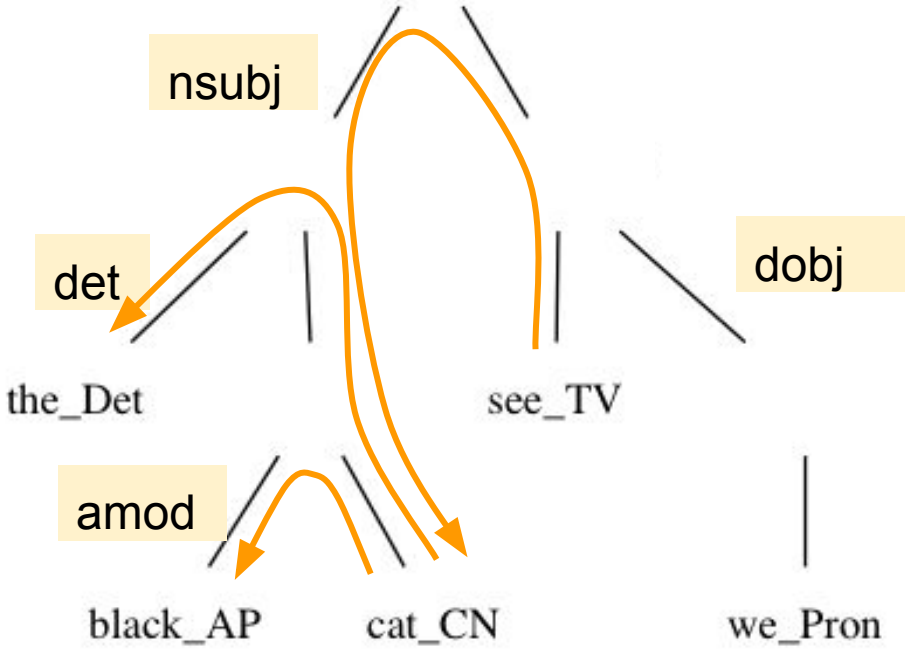
Dependency configuration

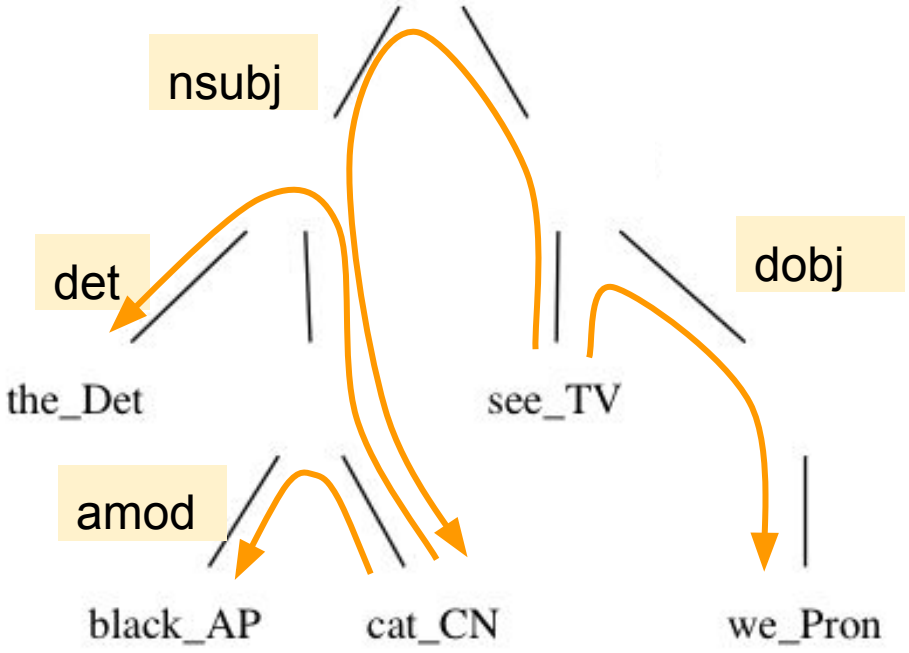
```
PredVP   nsubj head
ComplTV  head  dobj
DetCN    det   head
AdjCN    amod  head
```

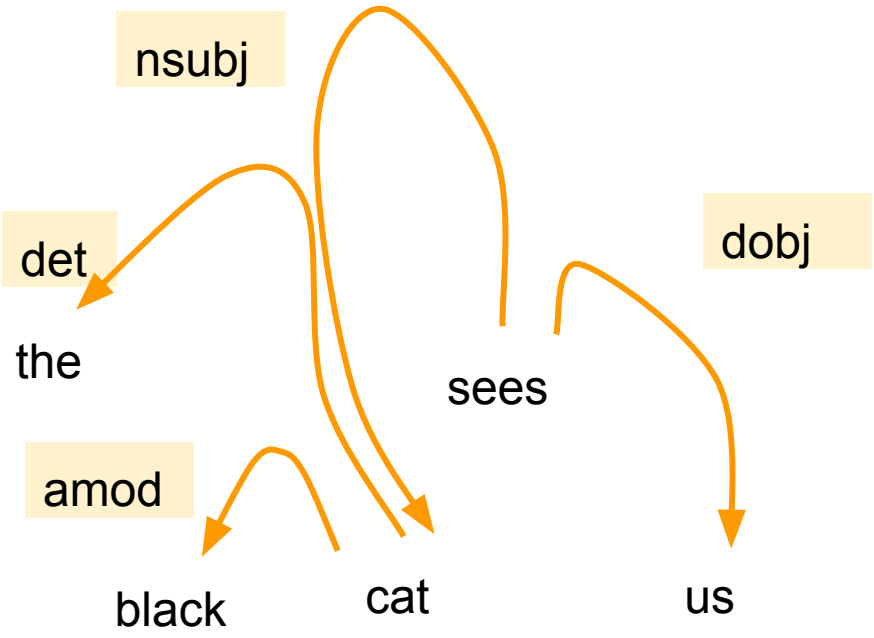

nsubj
det
dobj
the_Det
see_TV
amod
black_AP
cat_CN
we_Pron

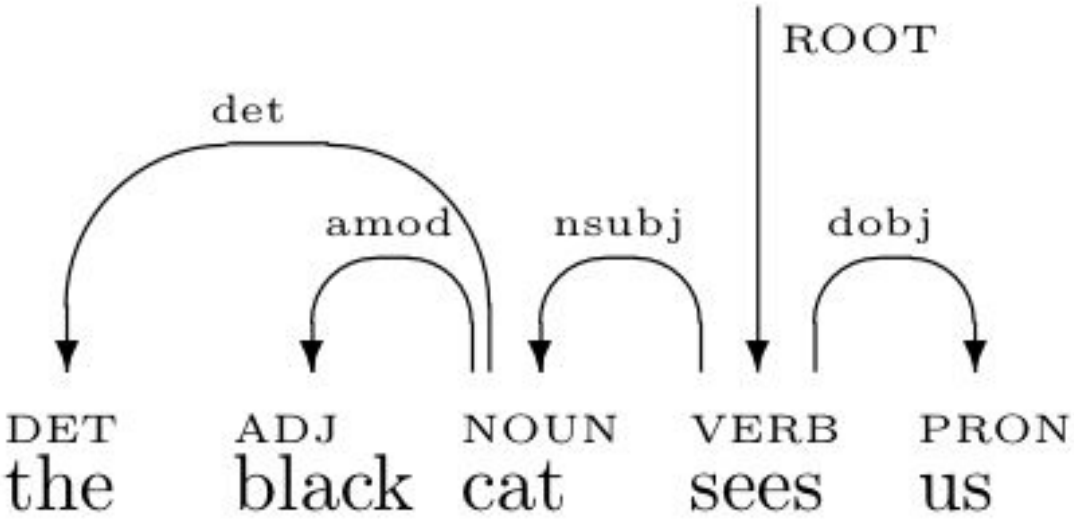

nsubj
det
dobj
the_Det
see_TV
amod
black_AP
cat_CN
we_Pron


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

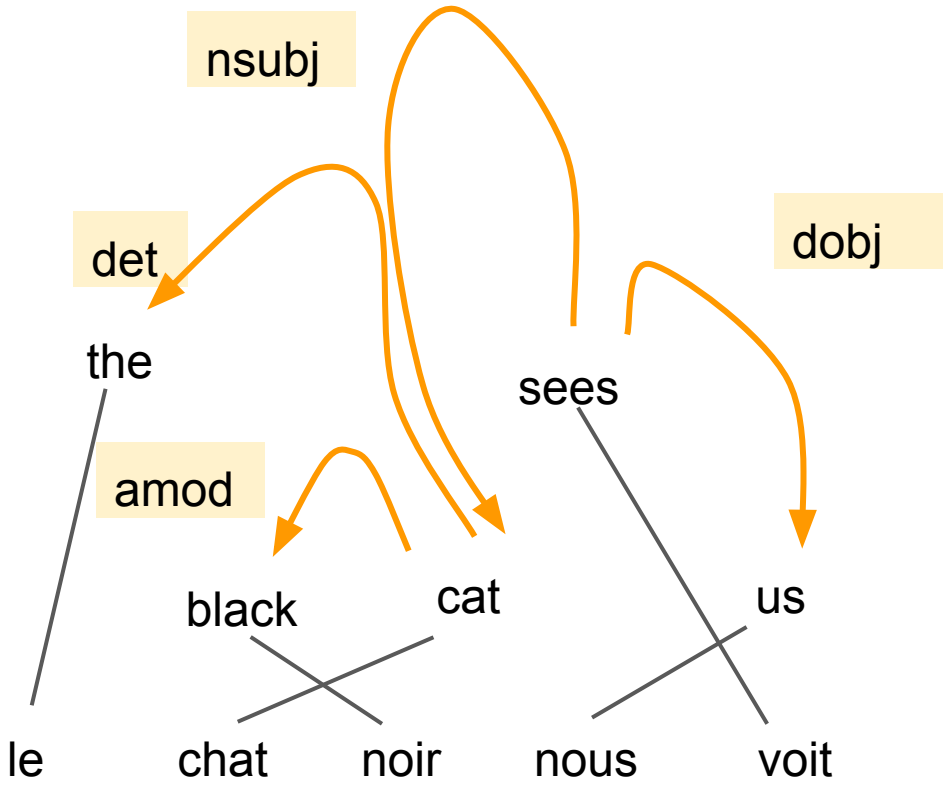

nsubj
det
dobj
the_Det
see_TV
amod
black_AP
cat_CN
we_Pron

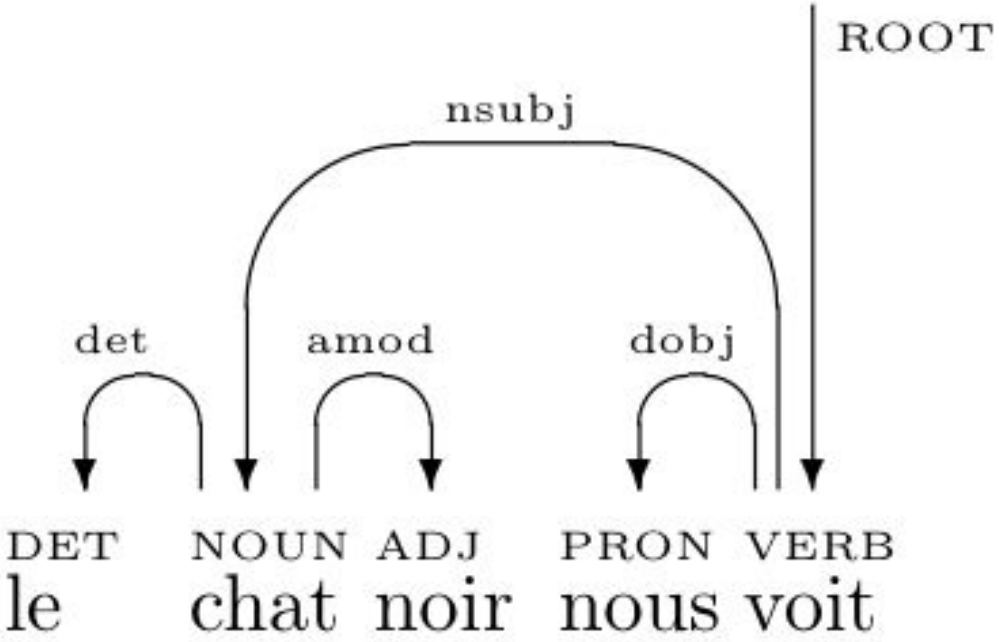

nsubj
det
dobj
the
sees
amod
black
cat
us



POS configuration

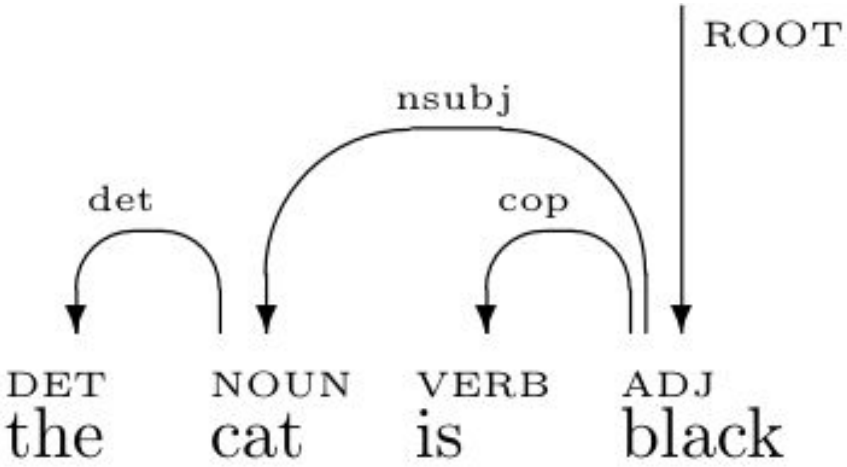
Det	DET
AP	ADJ
CN	NOUN
TV	VERB
Pron	PRON


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

Syncategorematic words

- pinpointing a difference in the ways of thinking:
  - dependency grammar is about words,
  - GF is about meanings


ROOT
nsubj
det
cop
DET
NOUN
VERB
ADJ
the
cat
is
black

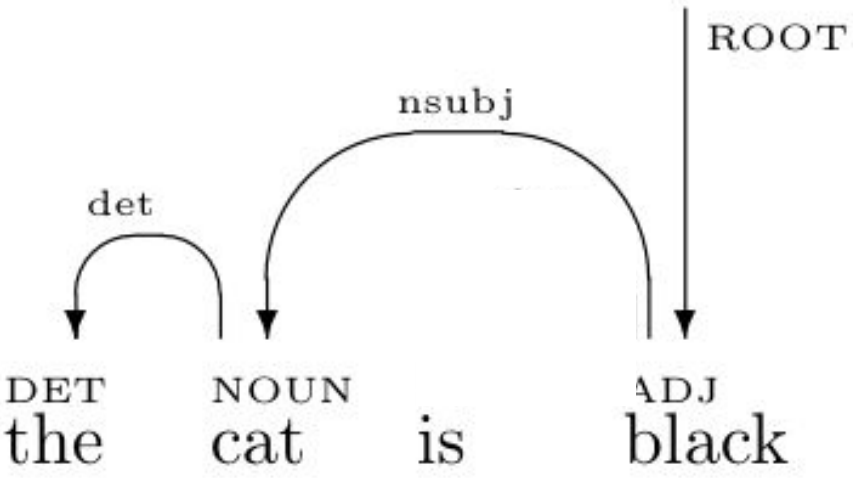
categorematic: word with its own category and function

```
fun cat_CN : CN
lin cat_CN = “cat”
```

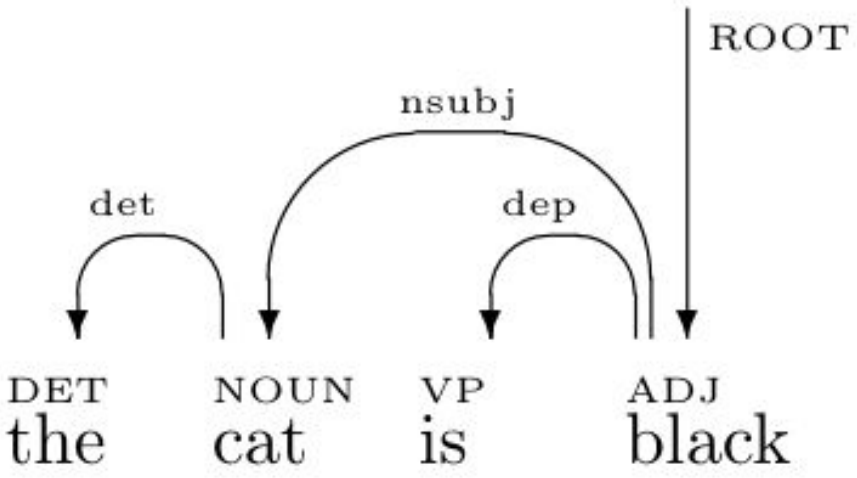
syncategorematic: word that is “between categories”

```
fun ComplAP : AP -> VP
lin   ComplAP ap = “is” ++ AP
```

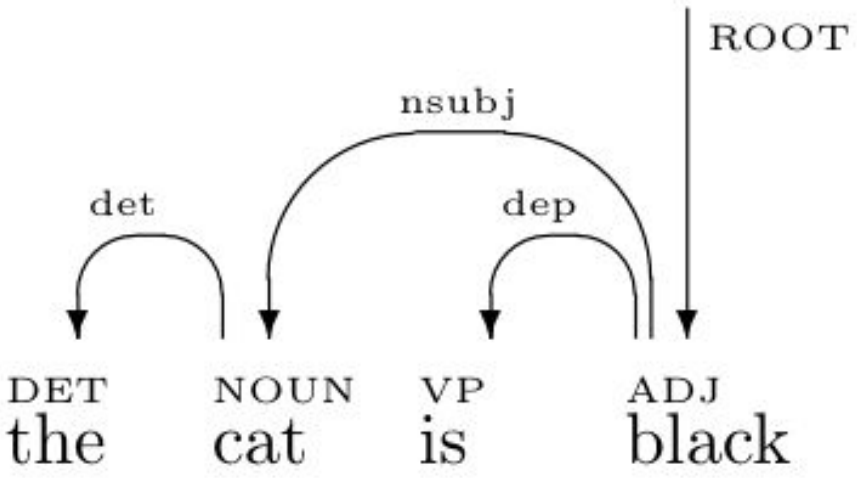
No semantics (`fun`) of its own. Not an argument. No label.


ROOT
nsubj
det
DET
NOUN
ADJ
the
cat
is
black

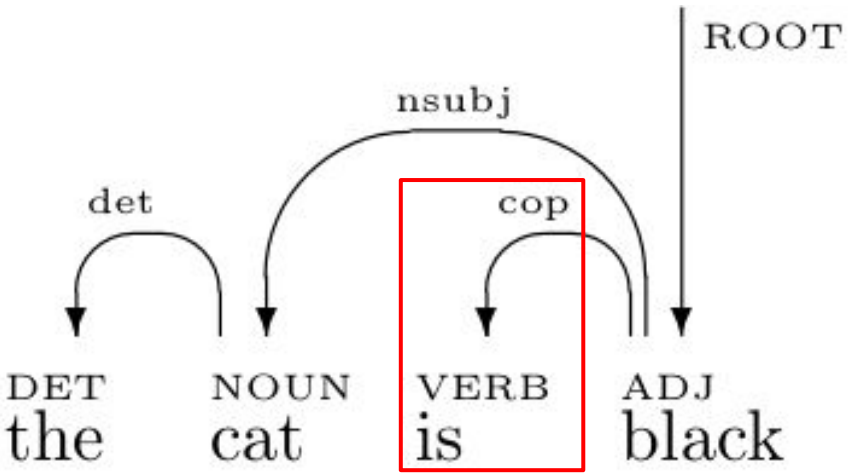
adding default labels



we get



UD wants



Other syncategorematic words

- negation words
- tense auxiliaries
- infinitive marks
- (sometimes) prepositions

Extended dependency configuration

abstract → abstract | concrete

local → local | nonlocal

- more complicated, not universal
+ less work than rewriting the grammar anyway
+ UD is still undergoing changes

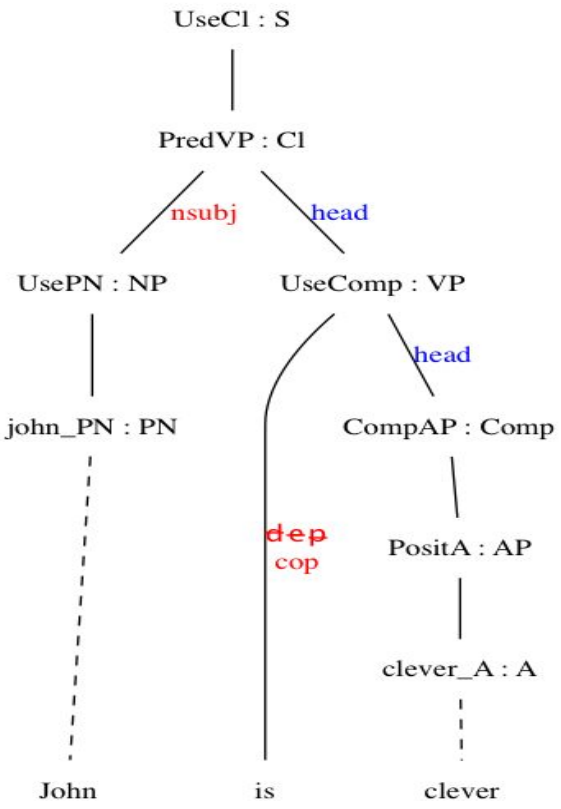
Concrete configs

UseComp in English

UseComp head {“is”, “was”, “be”, “are”} cop head

In Swedish

UseComp head {“ar”, “var”, “vara”, “varit”} cop head



Local Concrete configurations

Mappings defined on linearization of an abstract function for a specific language

These are necessary because of the ``level of abstraction'' in GF abstract syntax

The mappings specify re-labelling operations

- relabel an existing edge with new label
- modify an existing edge by changing the head and adding a new label

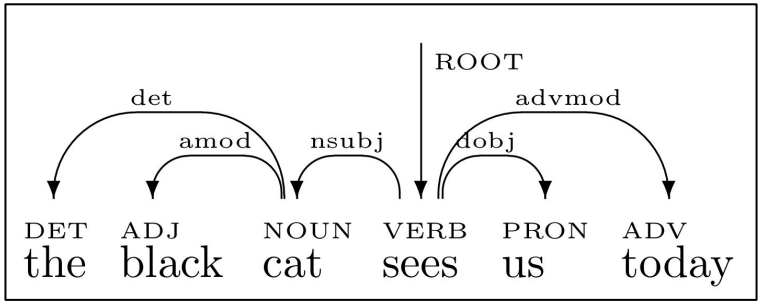
These operations match a set of words, or a record field or match anything

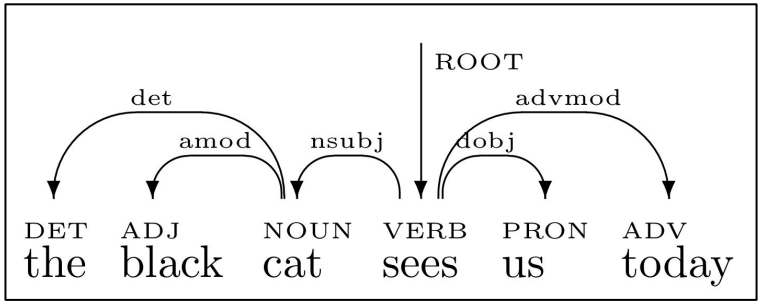
Demo ?

```
> parse “the cat sees us” | visual_dep -output=conll -file=ud.labels
```

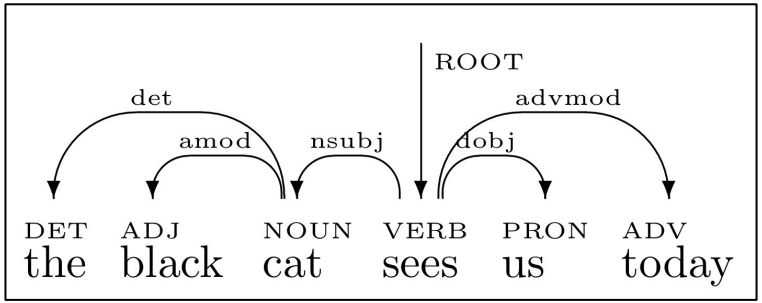
```
1     the    the_Det      DET    Det    _     2     det    _     _
2     cat    cat_CN NOUN  CN     _      3     nsubj  _     _
3     sees   see_TV VERB  TV     _      0     dep    _     _
4     us     we_Pron      PRON   Pron   _     3     dobj   _     _
```

UD2GF


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



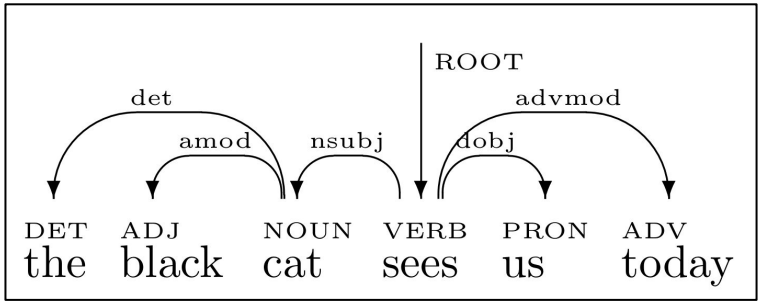
1	the	the	DET	_	3	det
2	black	black	ADJ	_	3	amod
3	cat	cat	NOUN	_	4	nsubj
4	sees	see	VERB	_	0	root
5	us	we	PRON	_	4	dobj
6	today	today	ADV	_	4	advmod



1	the	the	DET	_	3	det
2	black	black	ADJ	_	3	amod
3	cat	cat	NOUN	_	4	nsubj
4	sees	see	VERB	_	0	root
5	us	we	PRON	_	4	dobj
6	today	today	ADV	_	4	advmod

tree

```
root see VERB _ 4

    nsubj cat NOUN _ 3

        det the DET _ 1

        amod black ADJ _ 2

    dobj we PRON _ 5

    advmod today ADV _ 6
```



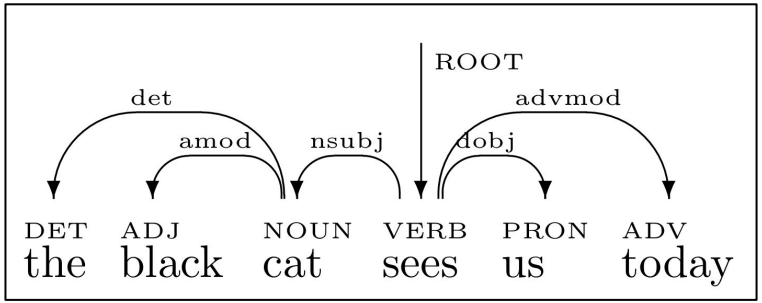
1	the	the	DET	_	3	det
2	black	black	ADJ	_	3	amod
3	cat	cat	NOUN	_	4	nsubj
4	sees	see	VERB	_	0	root
5	us	we	PRON	_	4	dobj
6	today	today	ADV	_	4	advmod

tree

```
root see VERB _ 4
    nsubj cat NOUN _ 3
        det the DET _ 1
        amod black ADJ _ 2
    dobj we PRON _ 5
    advmod today ADV _ 6
```

lexicon

see_V2	“see”
cat_N	“cat”
the_Det	“the”
black_A	“black”
we_Pron	“we”
today_Adv	“today”



1	the	the	DET	_	3	det
2	black	black	ADJ	_	3	amod
3	cat	cat	NOUN	_	4	nsubj
4	sees	see	VERB	_	0	root
5	us	we	PRON	_	4	dobj
6	today	today	ADV	_	4	advmod

tree

```
root see VERB _ 4
    nsubj cat NOUN _ 3
        det the DET _ 1
        amod black ADJ _ 2
    dobj we PRON _ 5
    advmod today ADV _ 6
```

lexicon

see_V2	“see”
cat_N	“cat”
the_Det	“the”
black_A	“black”
we_Pron	“we”
today_Adv	“today”

lexically annotated tree

```
root see_V2 V2 4
    nsubj cat_N N 3
        det the_Det Det 1
        amod black_A A 2
    dobj we_Pron Pron 5
    advmod today_Adv Adv 6
```

```
tree

root see_V2 V2 4
   nsubj cat_N N 3
      det the_Det Det 1
      amod black_A A 2
   dobj we_Pron Pron 5
   advmod today_Adv Adv 6
```

Postorder traversal: subtrees before their head

Invariant: every node has a valid GF tree

Goal: total GF tree at root

```
tree


root see_V2 V2 4

   nsubj cat_N N 3

      det the_Det Det 1

      amod black_A A 2

   dobj we_Pron Pron 5

   advmod today_Adv Adv 6
```

A node is done when no more functions apply

tree

```
root see_V2 V2 4
    nsubj (UseN 3) [cat_N] CN 3
        det the_Det Det 1
        amod (PositA 2) [black_A] AP 2
    dobj (UsePron 5) [we_Pron] NP 5
    advmod today_Adv Adv 6
```

endo

```
ModCN 2 3
```

exo

```
DetCN 1 3
```

when an endocentric function applies, use it first

tree

```
root see_V2 V2 4
    nsubj (ModCN 2 3) [(UseN 3),cat_N] CN 3
        det the_Det Det 1
        amod (PositA 2) [black_A] AP 2
    dobj (UsePron 5) [we_Pron] NP 5
    advmod today_Adv Adv 6
```

exo

```
DetCN 1 3
```

tree

```
root see_V2 V2 4
   nsubj (DetCN 1 3) [(ModCN 2 3),(UseN 3),cat_N] NP 3
      det the_Det Det 1
      amod (PositA 2) [black_A] AP 2
   dobj (UsePron 5) [we_Pron] NP 5
   advmod today_Adv Adv 6
```

```
tree

root (PredVP 3 4) [(AdvVP 4 6),(ComplV2 4 5),see_V2] VP 4
   nsubj (DetCN 1 3) [(ModCN 2 3),(UseN 3),cat_N] NP 3
      det the_Det Det 1
      amod (PositA 2) [black_A] AP 2
   dobj (UsePron 5) [we_Pron] NP 5
   advmod today_Adv Adv 6
```

Root node contains a complete GF tree

Problems

Ambiguity

There can be several candidate Functions and Categories.

Incompleteness

The tree may have nodes not referenced from the AST.

Problems and solutions

Ambiguity

There can be several candidate Functions and Categories.

Maintain a list of trees at each node, not just one tree.

Incompleteness

The tree may have nodes not referenced from the AST.

Auxiliary rules for **syntcategorematic words**.

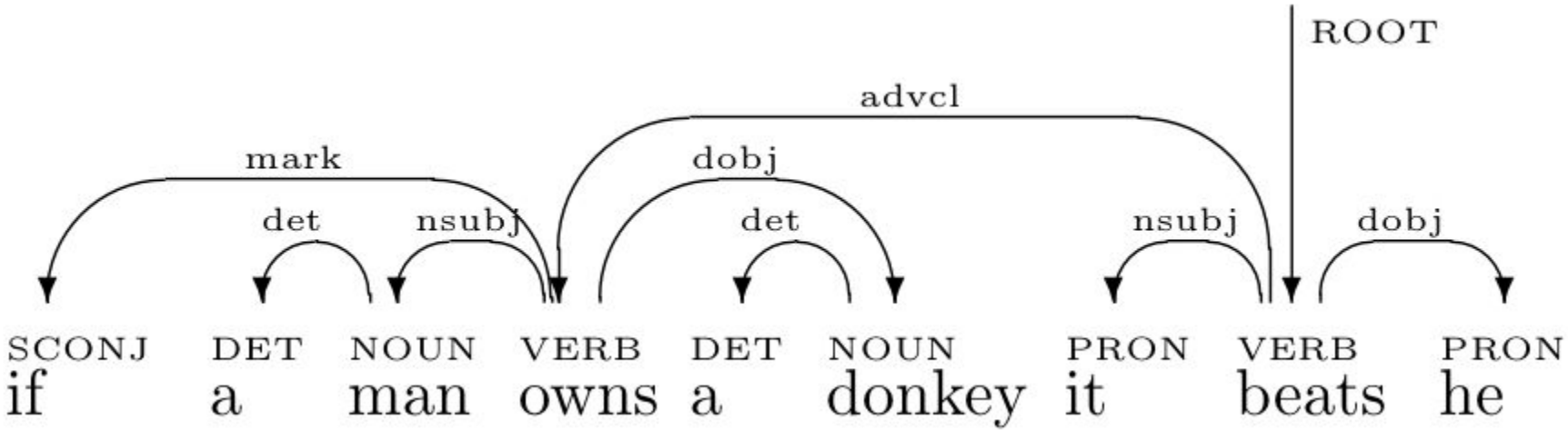
Backup functions attached as adverbial modifiers to AST nodes.

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
```

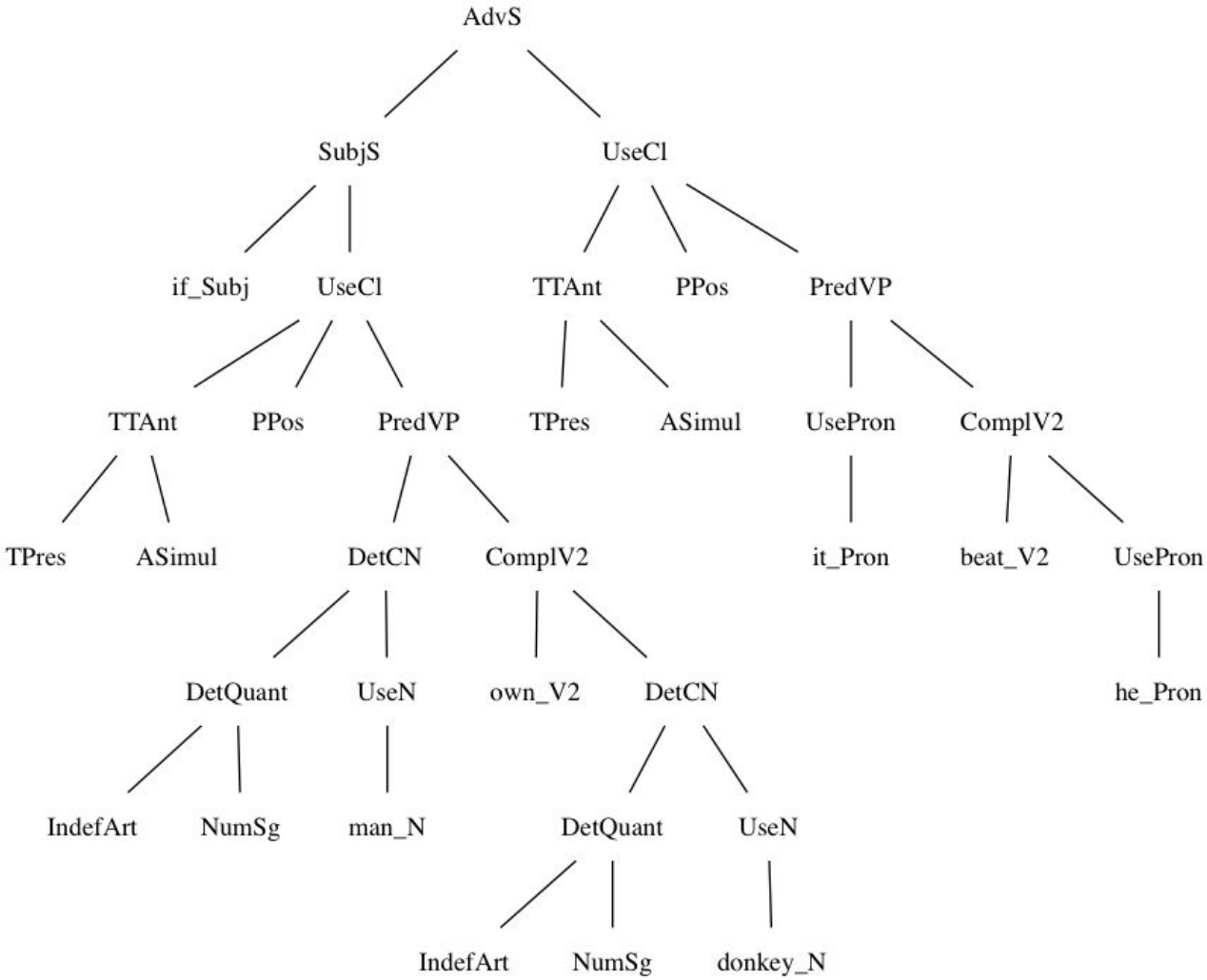
Eng: fast and friendly service "!" [they know my order when I walk in the door]
Fin: nopea ja ystävällinen palvelu "!" [he tuntevat minun järjestykseni kun minä kävelen ovessa]
Swe: snabb och vänlig tjänst "!" [de känner min ordning när jag går i dörren]

PARSED: 16/16 WITHOUT BACKUP: 6/16


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

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	_	_
```


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

Experiments

Analysing and converting UD treebanks

- English, Finnish, Swedish

Connecting GF generation to UD parser front-end

Results

language	#trees	#confs	%cov'd	%int'd
English	2077	45	98	75
Finnish	648	20	91	60
Finnish*	648	0	81	59
Swedish	1219	35	94	68
Swedish*	1219	0	84	63

UD_English/en-ud-test.conllu

UD_Finnish/fi-ud-test.conllu

UD_Swedish/sv-ud-test.conllu

Demo and End-user Applications

GF2UD

```
> parse “the cat sees us” | visual_dep -output=conll -file=ud.labels
```

```
1     the    the_Det      DET    Det    _     2     det    _     _
2     cat    cat_CN NOUN  CN     _      3     nsubj  _     _
3     sees   see_TV VERB  TV     _      0     root   _     _
4     us     we_Pron      PRON   Pron   _     3     dobj   _     _
```

UD2GF

```
$ ud2gf
    -lEng -t10000 -k3000 -a1 -g1 -Dscamifgtn
    -CUDTranslate.labels,UDTranslateEng.labels
            treebanks/UD_English/en-ud-test.conllu
```

https://github.com/GrammaticalFramework/gf-contrib/tree/master/ud2gf

UD pipelines

SyntaxNet : Google’s parser -- not lemmatizer

Stanford CORENLP -- no morphological analysis

UDPipe

Inhouse Graph-based Parsing pipeline

Questions ?

Non-local Abstract mappings

Expression patterns correspond to sub-trees or multi-level rules in GF Abstract Syntax

Have higher precedence that the corresponding local rule for the top function

But, we could get rid of these non-local mappings by re-engineering the RGL Abstract Syntax quite easily

- Could result in a increase of grammar size

Non-local Abstract

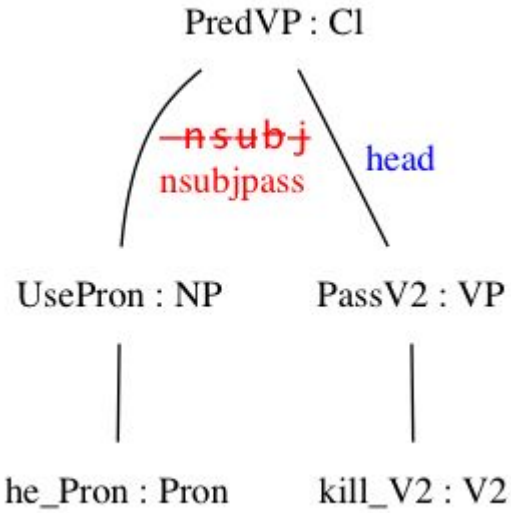
mappings

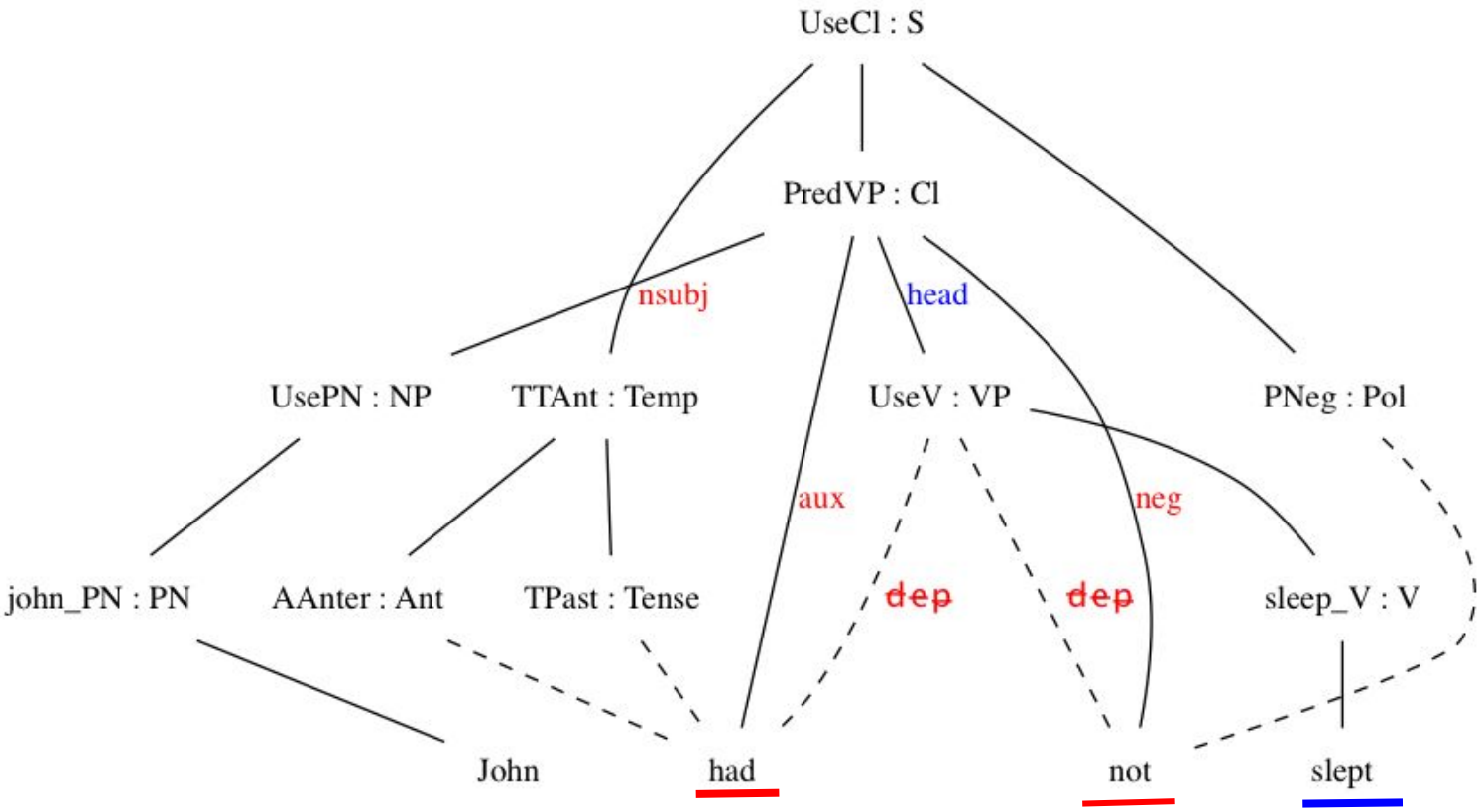
(PredVP ? (PassV2 ?)) nsubjpass head

PredVP	nsubj	head

(PredSCVP ? (PassV2 ?)) csubjpass head

PredSCVP	csubj	head




UseCl : S
PredVP : Cl
nsubj
head
UsePN : NP
TTAnt : Temp
UseV : VP
PNeg : Pol
aux
neg
john_PN : PN
AAnter : Ant
TPast : Tense
dep
dep
sleep_V : V
John
had
not
slept

Some sources of non-universal configurations

`ComplV2 : V2 -> NP -> VP`	`head (dobj | iobj | nmod)`
`ComplVV : VV -> VP -> VP`	`aux head | head xcomp`
	`| head mark xcomp`
`ExistNP : NP -> Cl`	“there is”, “det finns”,...

Non-Local Concrete mappings

Syncategoramatic words introduced by non-local rule

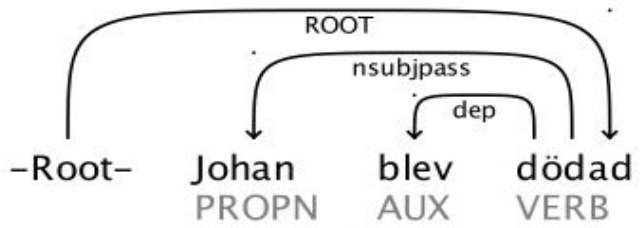
The same expression patterns from Non-local Abstract rules are used, to specify relabelling operations

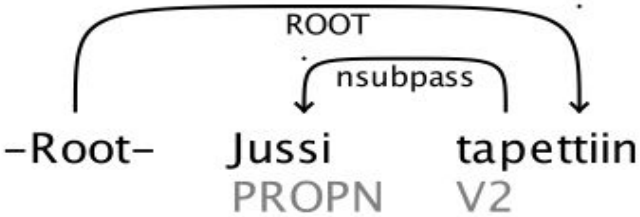
(UseCl ? PNeg ?) head {“not”, “n’t”} neg head

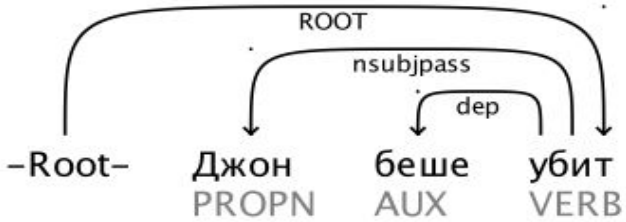
(UseCl ? ? ?) head {*} aux head

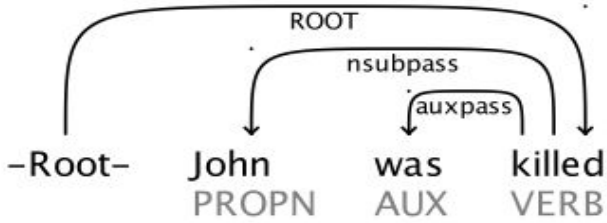
Auxiliaries for passive voice constructions?

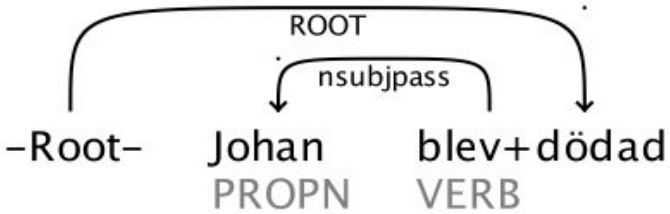
(UseCl ? ? (PredVP ? (PassV2 ?))) auxpass head

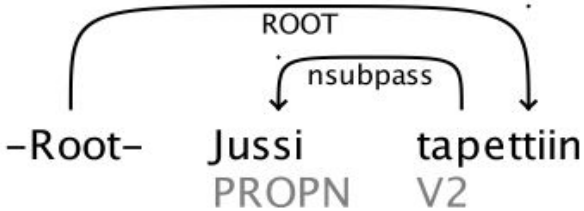

ROOT
nsubjpass
dep
-Root-
Johan
blev
dödad
PROPN
AUX
VERB

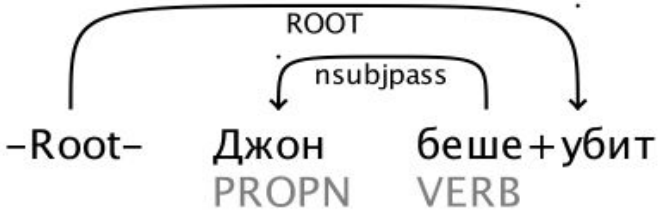

ROOT
nsubpass
-Root-
Jussi
tapettiin
PROPN
V2

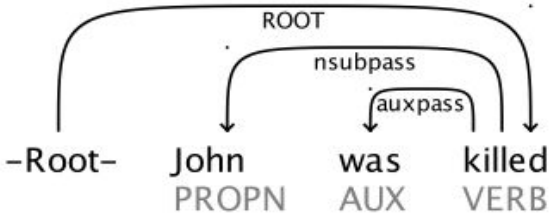

ROOT
nsubjpass
dep
-Root-
Джон
беше
убит
PROPN
AUX
VERB


ROOT
nsubpass
auxpass
-Root-
John
was
killed
PROPN
AUX
VERB


ROOT
nsubjpass
-Root-
Johan
blev+dödad
PROPN
VERB


ROOT
nsubpass
-Root-
Jussi
tapettiin
PROPN
V2


ROOT
nsubjpass
-Root-
Джон
беше+убит
PROPN
VERB


ROOT
nsubpass
auxpass
-Root-
John
was
killed
PROPN
AUX
VERB