

# EDAN20

## Language Technology

<http://cs.lth.se/edan20/>  
Chapter 11: Syntactic Formalisms

Pierre Nugues

Lund University  
[Pierre.Nugues@cs.lth.se](mailto:Pierre.Nugues@cs.lth.se)  
[http://cs.lth.se/pierre\\_nugues/](http://cs.lth.se/pierre_nugues/)

September 23, 2014



# Syntax

Syntax has been the core of linguistics in the US and elsewhere for many years

Noam Chomsky, professor at the MIT, has had an overwhelming influence, sometimes misleading

*Syntactic structures* (1957) has been a cult book for the past generation of linguists

Syntax can be divided into two parts:

- Formalism – How to represent syntax
- Parsing – How to get the representation of a sentence



# Syntactic Formalisms

The two most accepted formalisms use a tree representation:

- One is based on the idea of constituents
- Another is based on dependencies between words. Trees have originally been called stemmas

They are generally associated respectively to Chomsky and Tesnière. Later, constituent grammars evolved into unification grammars



# Constituency

Constituency can be expressed by context-free grammars. They are defined by

- ➊ A set of designated start symbols,  $\Sigma$ , covering the sentences to parse. This set can be reduced to a single symbol, such as `sentence`, or divided into more symbols: `declarative_sentence`, `interrogative_sentence`.
- ➋ A set of nonterminal symbols enabling the representation of the syntactic categories. This set includes the sentence and phrase categories.
- ➌ A set of terminal symbols representing the vocabulary: words of the lexicon, possibly morphemes.
- ➍ A set of rules,  $F$ , where the left-hand-side symbol of the rule is rewritten in the sequence of symbols of the right-hand side.



# DCG

These grammars can be mapped to DCG rules as for

*The boy hit the ball*

sentence --> np, vp.

np --> t, n.

vp -- verb, np.

t --> [the].

n --> [man] ; [ball] ; etc.

verb --> [hit] ; [took] ; etc.

Generation of sentences is one of the purposes of grammar according to Chomsky



# Chomsky Normal Form

In some parsing algorithms, it is necessary to have rules in the Chomsky normal form (CNF) with two right-hand-side symbols

Non-CNF rules:

$lhs \rightarrow rhs1, rhs2, rhs3.$

can be converted into a CNF equivalent:

$lhs \rightarrow rhs1, lhs\_aux.$

$lhs\_aux \rightarrow rhs2, rhs3.$



# Transformations

Rearrangement of sentences according to some syntactic relations:  
active/passive, declarative/interrogative, etc.

Transformations use rules – transformational rules or T rules –

*The boy will hit the ball/the ball will be (en) hit by the boy*

T1: np1, aux, v, np2 --->  
np2, aux, [be], [en], v, [by], np1



# Transformations





# Syntactic Categories (Penn Treebank)

	Categories	Description
1.	ADJP	Adjective phrase
2.	ADVP	Adverb phrase
3.	NP	Noun phrase
4.	PP	Prepositional phrase
5.	S	Simple declarative clause
6.	SBAR	Clause introduced by subordinating conjunction or 0
7.	SBARQ	Direct question introduced by <i>wh</i> -word or phrase
8.	SINV	Declarative sentence with subject-aux inversion
9.	SQ	Subconstituent of SBARQ excluding <i>wh</i> -word or phrase
10.	VP	Verb phrase
11.	WHADVP	<i>wh</i> -adverb phrase
12.	WHNP	<i>wh</i> -noun phrase
13.	WHPP	<i>wh</i> -prepositional phrase
14.	X	Constituent of unknown or uncertain category



# A Hand-Parsed Sentence using the Penn Treebank Annotation

*Battle-tested industrial managers here always buck up nervous newcomers with the tale of the first of their countrymen to visit Mexico, a boatload of samurai warriors blown ashore 375 years ago.*

```
( (S
  (NP Battle-tested industrial managers
    here)
  always
  (VP buck
    up
    (NP nervous newcomers)
    (PP with
      (NP the tale
        (PP of
```



# A Hand-Parsed Sentence using the Penn Treebank Annotation

```

(NP (NP the
      (ADJP first
        (PP of
          (NP their countrymen)))
      (S (NP *)
        to
        (VP visit
          (NP Mexico))))
      ,
      (NP (NP a boatload
        (PP of
          (NP (NP samurai warriors)
            (VP-1 blown
              ashore
                (ADVP (NP 375 years)
                  ago))))))
        (VP-1 *pseudo-attach*))))))

```



# Unification-based Grammars

Grammatical features such as case modify the word morphology

Cases	Noun groups
Nominative	der kleine Ober
Genitive	des kleinen Obers
Dative	dem kleinen Ober
Accusative	den kleinen Ober

The rule

`np --> det, adj, n.`

outputs ungrammatical phrases as:

`?-np(L, []).`

`[der, kleinen, Ober]; %wrong`

`[der, kleinen, Obers]; %wrong`

`[dem, kleine, Obers] %wrong`

...



# Representing Features

A possible solution is to use arguments: `np(case:C)` where the `C` value is a member of list `[nom, gen, dat, acc]`

```
np(gend:G, num:N, case:C, pers:P, det:D)
np(gend:G, num:N, case:C, pers:P, det:D) -->
  det(gend:G, num:N, case:C, pers:P, det:D),
  adj(gend:G, num:N, case:C, pers:P, det:D),
  n(gend:G, num:N, case:C, pers:P).
```



# A Small Fragment of German

```
det(gend:masc, num:sg, case:nom, pers:3, det:def) --> [der].
det(gend:masc, num:sg, case:gen, pers:3, det:def) --> [des].
det(gend:masc, num:sg, case:dat, pers:3, det:def) --> [dem].
det(gend:masc, num:sg, case:acc, pers:3, det:def) --> [den].
adj(gend:masc, num:sg, case:nom, pers:3, det:def) --> [kleine].
adj(gend:masc, num:sg, case:gen, pers:3, det:def) -->
[kleinen].
adj(gend:masc, num:sg, case:dat, pers:3, det:def) -->
[kleinen].
adj(gend:masc, num:sg, case:acc, pers:3, det:def) -->
[kleinen].
n(gend:masc, num:sg, case:nom, pers:3) --> ['Ober'].
n(gend:masc, num:sg, case:gen, pers:3) --> ['Obers'].
n(gend:masc, num:sg, case:dat, pers:3) --> ['Ober'].
n(gend:masc, num:sg, case:acc, pers:3) --> ['Ober'].
```



# A Unification-based Formalism

Unification-based grammars use a notation close to that of DCGs

$$\begin{array}{c} NP \\ \left[ \begin{array}{l} gend : G \\ num : N \\ case : C \\ pers : P \\ det : D \end{array} \right] \end{array} \rightarrow \begin{array}{c} DET \\ \left[ \begin{array}{l} gend : G \\ num : N \\ case : C \\ pers : P \\ det : D \end{array} \right] \end{array} \begin{array}{c} ADJ \\ \left[ \begin{array}{l} gend : G \\ num : N \\ case : C \\ pers : P \\ det : D \end{array} \right] \end{array} \begin{array}{c} N \\ \left[ \begin{array}{l} gend : G \\ num : N \\ case : C \\ pers : P \end{array} \right] \end{array}$$



# Some Rules

$$S \rightarrow \begin{matrix} NP \\ \left[ \begin{array}{l} num : N \\ case : nom \\ pers : P \end{array} \right] \end{matrix} \begin{matrix} VP \\ \left[ \begin{array}{l} num : N \\ pers : P \end{array} \right] \end{matrix}$$

$$\begin{matrix} VP \\ \left[ \begin{array}{l} num : N \\ pers : P \end{array} \right] \end{matrix} \rightarrow \begin{matrix} V \\ \left[ \begin{array}{l} trans : i \\ num : N \\ pers : P \end{array} \right] \end{matrix}$$

$$\begin{matrix} VP \\ \left[ \begin{array}{l} num : N \\ pers : P \end{array} \right] \end{matrix} \rightarrow \begin{matrix} V \\ \left[ \begin{array}{l} trans : t \\ num : N \\ pers : P \end{array} \right] \end{matrix} \begin{matrix} NP \\ [case : acc] \end{matrix}$$





# Feature Structures are Graphs

Structures can be embedded

$$\left[ \begin{array}{l} f_1 : v_1 \\ f_2 : \left[ \begin{array}{l} f_3 : v_3 \\ f_4 : \left[ \begin{array}{l} f_5 : v_5 \\ f_6 : v_6 \end{array} \right] \end{array} \right] \end{array} \right]$$

*Pronoun*

→ *er*

$$\left[ \begin{array}{l} \text{agreement} : \left[ \begin{array}{l} \text{gender} : \text{masc} \\ \text{number} : \text{sg} \\ \text{pers} : 3 \end{array} \right] \\ \text{case} : \text{nom} \end{array} \right]$$

*Pronoun*

→ *ihn*

$$\left[ \begin{array}{l} \text{agreement} : \left[ \begin{array}{l} \text{gender} : \text{masc} \\ \text{number} : \text{sg} \\ \text{pers} : 3 \end{array} \right] \\ \text{case} : \text{acc} \end{array} \right]$$



# Feature Structures are Graphs



# Unification-based Formalism

The feature notation is based on the name, not on the position

$$\begin{bmatrix} \textit{gen} : \textit{fem} \\ \textit{num} : \textit{pl} \\ \textit{case} : \textit{acc} \end{bmatrix} \text{ and } \begin{bmatrix} \textit{num} : \textit{pl} \\ \textit{case} : \textit{acc} \\ \textit{gen} : \textit{fem} \end{bmatrix}$$

are equivalent

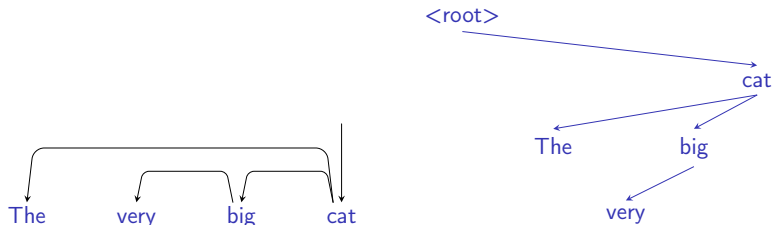
Unification is a generalization of Prolog unification

See the course book for the implementation



# Dependency Grammars

Dependency grammars (DG) describe the structure in term of links



Each word has a head or “régissant” except the root of the sentence.

A head has one or more modifiers or dependents:

*Cat* is the head of *big* and *the*; *big* is the head of *very*.

DG can be more versatile with a flexible word order language like German, Russian, or Latin.



# A Sentence Tree – Stemma



# Properties of Dependency Graphs

**Acyclic**



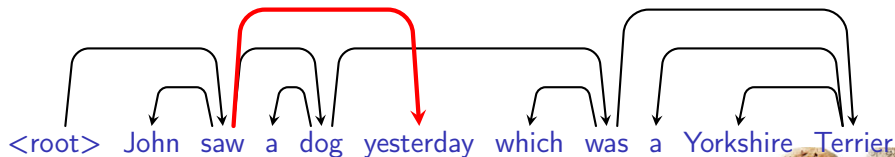
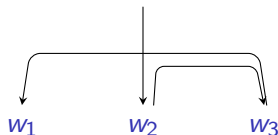
**Connected  
Projective**



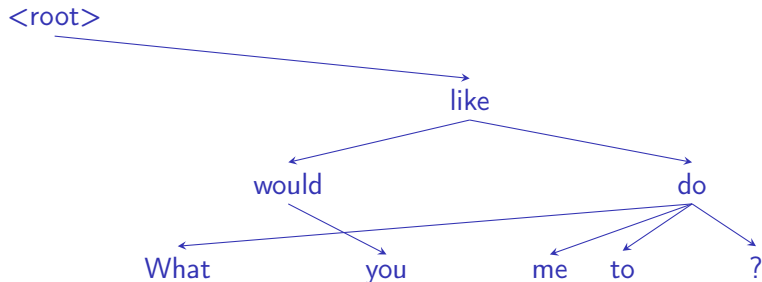
Each pair of words (Dep, Head), directly connected, is only separated by direct or indirect dependents of Dep or Head



# Nonprojective Graphs (McDonald and Pereira)



# Nonprojective Graphs (Järvinen and Tapanainen)





# Valence

Tesnière makes a distinction between essential and circumstantial complements

Essential – or core – complements are for instance subject and objects.

Circumstantial – or noncore – complements are the adjuncts

Valence corresponds to the verb saturation of its essential complements



# Valence Examples

Val.	Examples	Frames
0	<i>it's raining</i>	<i>raining</i> []
1	<i>he's sleeping</i>	<i>sleeping</i> [subject : he]
2	<i>she read this book</i>	<i>read</i> [subject : she object : book]
3	<i>Elke gave a book to Wolfgang</i>	<i>gave</i> [subject : Elke object : book iobject : Wolfgang]
4	<i>I moved the car from here to the street</i>	<i>moved</i> [subject : I object : car source : here destination : street]



# Subcategorization Frames

Valence is a model of verb construction. It can be extended to more specific patterns as in the *Oxford Advanced Learner's Dictionary* (OALD).

Verb	Complement structure	Example
<i>slept</i>	None (Intransitive)	<i>I slept</i>
<i>bring</i>	NP	<i>The waiter brought the meal</i>
<i>bring</i>	NP + to + NP	<i>The waiter brought the meal to the patron</i>
<i>depend</i>	on + NP	<i>It depends on the waiter</i>
<i>wait</i>	for + NP + to + VP	<i>I am waiting for the waiter to bring the meal</i>
<i>keep</i>	VP(ing)	<i>He kept working</i>
<i>know</i>	that + S	<i>The waiter knows that the patron loves fish</i>



# Subcategorization Frames in German

Verb	Complement structure	Example
<i>schlafen</i>	None (Intransitive)	<i>Ich habe geschlafen</i>
<i>bringen</i>	NP(Accusative)	<i>Der Ober hat eine Speise gebracht</i>
<i>bringen</i>	NP(Dative) NP(Accusative)	+ <i>Der Ober hat dem Kunde eine Speise gebracht</i>
<i>abhängen</i>	von + NP(Dative)	<i>Es hängt vom Ober ab</i>
<i>warten</i>	auf + S	<i>Er wartete auf dem Ober, die Speise zu bringen</i>
<i>fortsetzen</i>	NP	<i>Er hat die Arbeit fortgesetzt</i>
<i>wissen</i>	NP(Final verb)	<i>Der Ober weiß, das der Kunde Fisch liebt</i>



# Dependencies and Grammatical Functions

The dependency structure generally reflects the traditional syntactic representation

The links can be annotated with grammatical function labels.

In a simple sentence, it corresponds to the subject and the object

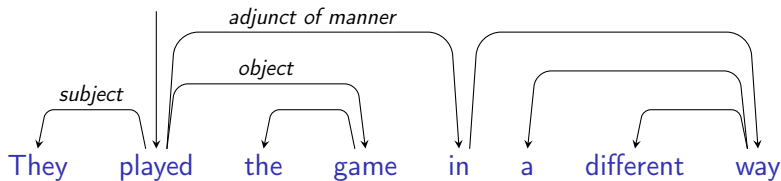


Probably a more natural description to tie syntax to semantics



# Dependencies and Functions (II)

Adjuncts form another class of functions that modify the verb  
They include prepositional phrases whose head is set arbitrarily to the front preposition  
Adjuncts include adverbs that modify a verb



# Dependency Parse Tree



Word pos.	Word	Direction	Head	Head position	Function
1	<i>Bring</i>	*		Root	Main verb
2	<i>the</i>	>	<i>meal</i>	3	Determiner
3	<i>meal</i>	<	<i>Bring</i>	1	Object
4	<i>to</i>	<	<i>Bring</i>	1	Location
5	<i>the</i>	>	<i>table</i>	6	Determiner
6	<i>table</i>	<	<i>to</i>	4	Prepositional complement



# Representing Dependencies

$$D = \{ \langle \text{Head}(1), \text{Rel}(1) \rangle, \langle \text{Head}(2), \text{Rel}(2) \rangle, \dots, \langle \text{Head}(n), \text{Rel}(n) \rangle \},$$

The representation of *Bring the meal to the table*:

$$D = \{ \langle 0, \text{root} \rangle, \langle 3, \text{det} \rangle, \langle 1, \text{object} \rangle, \langle 1, \text{loc} \rangle, \langle 6, \text{det} \rangle, \langle 4, \text{pcomp} \rangle \},$$





# Annotation: MALT XML

```
<sentence id="24">
<word id="1" form="Dessutom" postag="ab" head="2"
  deprel="ADV"/>
<word id="2" form="höjs" postag="vb.prs.sfo" head="0"
  deprel=""/>
<word id="3" form="åldergränsen" postag="nn.utr.sin.def.nom"
  head="2" deprel="SUB"/>
<word id="4" form="till" postag="pp" head="2" deprel="ADV"/>
<word id="5" form="18" postag="rg.nom" head="6" deprel="DET"/>
<word id="6" form="år" postag="nn.neu.plu.ind.nom" head="4"
  deprel="PR"/>
<word id="7" form="." postag="mad" head="2" deprel="IP"/>
</sentence>
```



TMALT XML is an extended annotation

# Annotation: CoNLL

The CoNLL shared tasks organize evaluations of machine-learning systems for natural language processing.

They define formats to share data between participants.

1	Dessutom	—	AB	AB	—	2	+A	—	—
2	höjs	—	VV	VV	—	0	ROOT	—	—
3	åldergränsen	—	NN	NN	—	2	SS	—	—
4	till	—	PR	PR	—	2	OA	—	—
5	18	—	RO	RO	—	6	DT	—	—
6	år	—	NN	NN	—	4	PA	—	—
7	.	—	IP	IP	—	2	IP	—	—



# Annotation: CoNLL

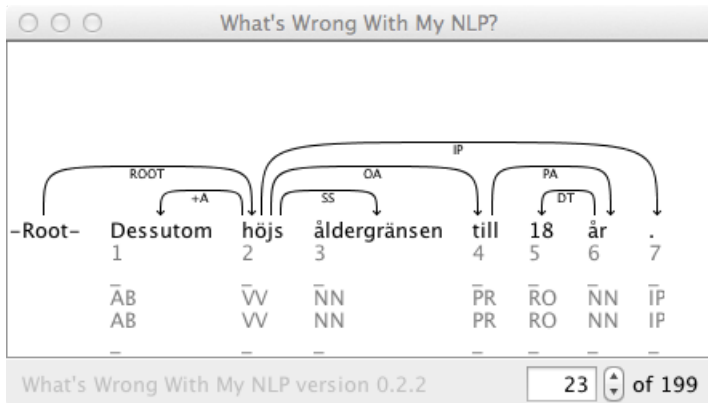
#	Name	Description
1	ID	Token index, starting at 1 for each sentence.
2	FORM	Word form or punctuation.
3	LEMMA	Lemma or stem.
4	CPOSTAG	Part-of-speech tag.
5	POSTAG	Fine-grained part-of-speech tag.
6	FEATS	Unordered set of morphological features separated by a vertical bar ( ).
7	HEAD	Head of the current token, which is either a value of ID or zero (0) if this is the root.
8	DEPREL	Dependency relation to the HEAD.
9	PHEAD	Projective head of current token, which is either a value of ID or zero (0). The dependency structure resulting from the PHEAD column is guaranteed to be projective, when available in the corpus.
10	PDEPREL	Dependency relation to the PHEAD.



# Visualizing Dependencies

Using *What's Wrong With My NLP*

(<https://code.google.com/p/whatswrong/>):



# Function Annotation Tagset (Järvinen and Tapanainen 1997)

Name	Description	Example
<b>Main functions</b>		
main	Main element	<i>He doesn't <b>know</b> whether to send a gift</i>
qtag	Question tag	<i>Let's play another game, <b>shall</b> we?</i>
<b>Intranuclear links</b>		
v-ch	Verb chain	<i>It <b>may have been being</b> examined</i>
pcomp	Prepositional complement	<i>They played the game <b>in a different way</b></i>
phr	Verb particle	<i>He asked me who would look <b>after</b> the baby</i>



# Function Annotation Tagset (Järvinen and Tapanainen 1997)

## Verb complementation

subj	Subject	
obj	Object	<i>I gave him <u>my address</u></i>
comp	Subject complement.	<i>It has become <b>marginal</b></i>
dat	Indirect object	<i>Pauline gave it <u>to Tom</u></i>
oc	Object complement	<i>His friends call him <b>Ted</b></i>
copred	Copredicative	<i>We took a swim <b>naked</b></i>
voc	Vocative	<i>Play it again, <b>Sam</b></i>

## Determinative functions

qn	Quantifier	<i>I want <b>more</b> money</i>
det	Determiner	<i><b>Other</b> members will join...</i>
neg	Negator	<i>It is <b>not</b> coffee that I like, but <b>tea</b></i>



# Function Annotation Tagset (Järvinen and Tapanainen 1997)

## Modifiers

attr	Attributive nominal	<i><u>Knowing</u> no French, I couldn't express my thanks</i>
mod	Other postmodifiers	<i>The baby, <u>Frances Bean</u>, was. . .</i> <i>The people <u>on the bus</u> were singing</i>
ad	Attributive adverbial	<i>She is <u>more</u> popular</i>

## Junctives

cc	Coordination	<i><u>Two or</u> more cars. . .</i>
----	--------------	-------------------------------------



# Dependency vs. Constituency

Constituency (most textbooks) is a declining formalism

It cannot properly handle many languages: Swedish, Russian, Czech, Arabic, etc.

Dependency parsing can handle all these languages as well as English, German, French, etc.

Dependency parsing has improved considerably over the last 4 years: see CoNLL 2006 and 2007.

CoNLL 2008 and 2009 extend it to semantic parsing

However, constituency and dependency are (weakly) compatible provided that we restrict us to projective dependency graphs





# From Constituency to Dependency

It is possible to convert constituent trees into dependency graphs  
We need to identify a headword in all the PS rules, here with a star:

s --> np, vp\*.

vp --> verb\*, np.

np --> det, noun\*.

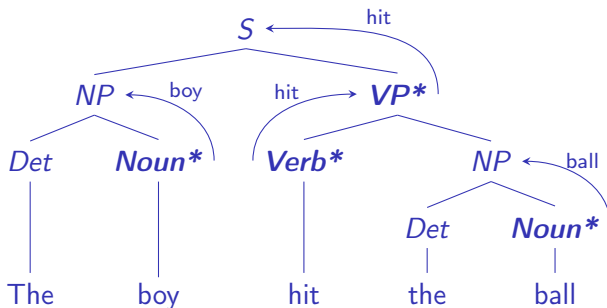
Parsers by Magerman and Collins used this to convert the Penn Treebank constituent annotation for their dependency parsers

When projective, dependency structures are loosely compatible with constituent grammars.



# From Constituency to Dependency (II)

A constituent tree with head-marked rules:



The resulting dependency graph:

