Computational Linguistics II: Parsing

Derivations in CFGs; Complexity

CSGs

Frank Richter & Jan-Philipp Söhn

fr@sfs.uni-tuebingen.de, jp.soehn@uni-tuebingen.de

The Big Picture

hierarchy	grammar	machine	other
type 3	reg. grammar	D/NFA	reg. expressions
det. cf.	LR(k) grammar	DPDA	
type 2	CFG	PDA	
type 1	CSG	LBA	
type 0	unrestricted grammar	Turing machine	

Derivation Steps of Grammars

Definition

For every grammar G with $G = \langle N, T, P, S \rangle$ and $\Sigma = N \cup T$, for every $u, v \in \Sigma^*$,

if there is a rule $l \rightarrow r \in P$ with $u = w_1 l w_2$ and $v = w_1 r w_2$, where $w_1, w_2 \in \Sigma^*$ then

$u \Rightarrow_G^1 v$.

We say that u directly derives v in grammar G.

We write $\Rightarrow_G^*$ for the reflexive transitive closure of $\Rightarrow_G^1$ and omit the subscript G if the grammar is clear from the context.

Language Generated by a Grammar

Definition

For every grammar G with $G = \langle N, T, P, S \rangle$ the language $L(G)$ generated by G is

$$L(G) = \{x \in T^* | S \Rightarrow_G^* x\}.$$

More on Derivations (1)

If at each step in a derivation a production is applied to the leftmost nonterminal, then the derivation is said to be *leftmost*.

A derivation in which the rightmost nonterminal is replaced at each step is said to be *rightmost*.

An Example: Grammar (1)

(1) S $\rightarrow$ NP VP

(2) NP $\rightarrow$ Det N

(3abc) Det $\rightarrow$ der | die | dem

(4abc) N $\rightarrow$ mann | frau | fernglas

(5) N $\rightarrow$ N PP

(6) PP $\rightarrow$ mit NP

(7ab) VP $\rightarrow$ V NP | V NP PP

(8) V $\rightarrow$ sah

An Example: Derivation (2)

S $\Rightarrow^1$ NP VP $\Rightarrow^1$ Det N VP $\Rightarrow^1$ Det N V NP PP $\Rightarrow^1$

Det N V NP mit NP $\Rightarrow^1$ Det N V Det N mit NP $\Rightarrow^1$

Det N V Det N mit Det N $\Rightarrow^1$

der N V Det N mit Det N $\Rightarrow^1$

der mann V Det N mit Det N $\Rightarrow^1$

der mann sah Det N mit Det N $\Rightarrow^1$

der mann sah die N mit Det N $\Rightarrow^1$

der mann sah die frau mit Det N $\Rightarrow^1$

der mann sah die frau mit dem N $\Rightarrow^1$

der mann sah die frau mit dem fernglas

An Example: Leftmost Derivation (3)

S $\Rightarrow^1$ NP VP $\Rightarrow^1$ Det N VP $\Rightarrow^1$ der N VP $\Rightarrow^1$

der mann VP $\Rightarrow^1$ der mann V NP PP $\Rightarrow^1$

der mann sah NP PP $\Rightarrow^1$

der mann sah Det N PP $\Rightarrow^1$

der mann sah die N PP $\Rightarrow^1$

der mann sah die frau PP $\Rightarrow^1$

der mann sah die frau mit NP $\Rightarrow^1$

der mann sah die frau mit Det N $\Rightarrow^1$

der mann sah die frau mit dem N $\Rightarrow^1$

der mann sah die frau mit dem fernglas

An Example: Leftmost Derivation (3b)

S $\stackrel{1}{\Rightarrow}$ NP VP $\stackrel{2}{\Rightarrow}$ Det N VP $\stackrel{3a}{\Rightarrow}$ der N VP $\stackrel{4a}{\Rightarrow}$

der mann VP $\stackrel{7b}{\Rightarrow}$ der mann V NP PP $\stackrel{8}{\Rightarrow}$

der mann sah NP PP $\stackrel{2}{\Rightarrow}$

der mann sah Det N PP $\stackrel{3b}{\Rightarrow}$

der mann sah die N PP $\stackrel{4b}{\Rightarrow}$

der mann sah die frau PP $\stackrel{6}{\Rightarrow}$

der mann sah die frau mit NP $\stackrel{2}{\Rightarrow}$

der mann sah die frau mit Det N $\stackrel{3c}{\Rightarrow}$

der mann sah die frau mit dem N $\stackrel{4c}{\Rightarrow}$

der mann sah die frau mit dem fernglas

An Example: Another Derivation (3)

S $\Rightarrow^1$ NP VP $\Rightarrow^1$ Det N VP $\Rightarrow^1$ der N VP $\Rightarrow^1$

der mann VP $\Rightarrow^1$ der mann V NP$\Rightarrow^1$

der mann sah NP $\Rightarrow^1$

der mann sah Det N$\Rightarrow^1$

der mann sah die N $\Rightarrow^1$

der mann sah die N PP $\Rightarrow^1$

der mann sah die frau PP $\Rightarrow^1$

der mann sah die frau mit NP $\Rightarrow^1$

der mann sah die frau mit Det N $\Rightarrow^1$

der mann sah die frau mit dem N $\Rightarrow^1$

der mann sah die frau mit dem fernglas

More on Derivations (2)

Derivation trees are useful summaries of the derivations of strings in a language. Corresponding to a particular derivation tree of a string w there is exactly one unique leftmost and one unique rightmost derivation.

More on Derivations (2)

Derivation trees are useful summaries of the derivations of strings in a language. Corresponding to a particular derivation tree of a string w there is exactly one unique leftmost and one unique rightmost derivation.

There are inherently ambiguous CF languages, i.e. languages L such that each CFG G generating L generates strings with more than one leftmost/rightmost derivation.

Parsing Versus Recognition

- **Recognition**:

Parsing Versus Recognition

- **Recognition**:
  - determine whether a sentence is part of the language

Parsing Versus Recognition

- **Recognition**:
  - determine whether a sentence is part of the language
  - output: yes / no

Parsing Versus Recognition

- **Recognition**:
  - determine whether a sentence is part of the language
  - output: yes / no
- **Parsing**:

Parsing Versus Recognition

- **Recognition**:
  - determine whether a sentence is part of the language
  - output: yes / no
- **Parsing**:
  - determine the structure of a sentence if it belongs to the language

Parsing Versus Recognition

- **Recognition**:
  - determine whether a sentence is part of the language
  - output: yes / no
- **Parsing**:
  - determine the structure of a sentence if it belongs to the language
  - output: syntactic tree

Parsing Versus Recognition

- **Recognition**:
  - determine whether a sentence is part of the language
  - output: yes / no
- **Parsing**:
  - determine the structure of a sentence if it belongs to the language
  - output: syntactic tree
  - needs “memory” of which rules were applied

Time Complexity (1)

- A parser's time complexity depends on the parsing algorithm as well as on the size of the grammar (mostly neglected) and the length of the input n.

Time Complexity (1)

- A parser's time complexity depends on the parsing algorithm as well as on the size of the grammar (mostly neglected) and the length of the input n.
- Complexity is described in terms of the **worst case** behavior of algorithms.

Time Complexity (1)

- A parser's time complexity depends on the parsing algorithm as well as on the size of the grammar (mostly neglected) and the length of the input n.
- Complexity is described in terms of the **worst case** behavior of algorithms.
- The complexity of an algorithm is **independent of the implementation**.

Time Complexity (1)

- A parser's time complexity depends on the parsing algorithm as well as on the size of the grammar (mostly neglected) and the length of the input n.
- Complexity is described in terms of the **worst case** behavior of algorithms.
- The complexity of an algorithm is **independent of the implementation**.
- The empirically tested efficiency is in most cases better than the complexity, but it depends not only on the hardware and the implementation but also on the grammar and on the input (ambiguity rate, length, grammaticality).

Time Complexity (2)

Different orders of complexity:

Time Complexity (2)

Different orders of complexity:

- *linear complexity* $O(n)$: each word takes a constant time to process

Time Complexity (2)

Different orders of complexity:

- *linear complexity* $O(n)$: each word takes a constant time to process
- *quadratic complexity* $O(n^2)$: processing time is proportional to the square of the input length

Time Complexity (2)

Different orders of complexity:

- *linear complexity* $O(n)$: each word takes a constant time to process
- *quadratic complexity* $O(n^2)$: processing time is proportional to the square of the input length
- *cubic complexity* $O(n^3)$

Time Complexity (2)

Different orders of complexity:

- *linear complexity* $O(n)$: each word takes a constant time to process
- *quadratic complexity* $O(n^2)$: processing time is proportional to the square of the input length
- *cubic complexity* $O(n^3)$
- *polynomial complexity* $O(n^p)$: proportional to the pth power of n

Time Complexity (2)

Different orders of complexity:

- *linear complexity* $O(n)$: each word takes a constant time to process
- *quadratic complexity* $O(n^2)$: processing time is proportional to the square of the input length
- *cubic complexity* $O(n^3)$
- *polynomial complexity* $O(n^p)$: proportional to the pth power of n
- *exponential complexity* $O(p^n)$: proportional to the nth power of factor p

Time Complexity – Example

Comparison of polynomial and exponential complexities for p = 2 and for p=10:

n	poly.	expo.	poly.	expo.
1	2	1		
2	4	4		
3	9	8		
4	16	16		
5	25	32		
6	36	64		
7	49	128		
8	64	256		
9	81	512		
10	100	1024		

Time Complexity – Example

Comparison of polynomial and exponential complexities for p = 2 and for p=10:

n	poly.	expo.	poly.	expo.
1	2	1	1	10
2	4	4	1 024	100
3	9	8	59 049	1 000
4	16	16	1 048 576	10 000
5	25	32	9 765 625	100 000
6	36	64	60 466 176	1 000 000
7	49	128	282 475 200	10 000 000
8	64	256		
9	81	512		
10	100	1024		

Context Sensitive Grammars (1)

Let us be slightly more precise than before about our notion of grammars:

Definition
A quadruple $\langle N, T, P, S \rangle$ is a *grammar* iff

N and T finite sets, $N \cap T = \emptyset$,
$P \subseteq (N \cup T)^+ \times (N \cup T)^*$, P a finite set,
and $S \in N$.

Remark:
We write $A \rightarrow B$ for the elements of P, $A \in (N \cup T)^+$ and $B \in (N \cup T)^*$.

Context Sensitive Grammars (2)

Definition

A grammar $\langle N, T, P, S \rangle$ is *context-sensitive* iff every production in P is of the form $x_1 A x_2 \rightarrow x_1 y x_2$, with $x_1, x_2 \in \Sigma^*$, $y \in \Sigma^+$, $A \in N$ and the possible exception of $C \rightarrow \epsilon$ in case C does not occur on the righthand side of a rule in P.

Definition

A grammar $\langle N, T, P, S \rangle$ is *monotonic* iff for every production $l \rightarrow r \in P$, $|l| \leq |r|$.

Context Sensitive Grammars (3)

We will not prove the following important theorem:

Theorem

(I) For every monotonic grammar G_M there is a context-sensitive grammar G_S such that $L(G_M) = L(G_S)$.

(II) For every context-sensitive grammar G_S there is a monotonic grammar G_M such that $L(G_S) = L(G_M)$.

Remark:
The languages generated by monotonic and context-sensitive grammars are generally referred to as context-sensitive languages. Their grammars are called Type 1 grammars.