Lecture L&M 5

Chomsky Normal Form, CYK Parsing

5.1 Noncontracting grammars

For each of the following grammars, construct an equivalent grammar G_L with a nonrecursive start symbol that is "essentially noncontracting", i.e. only the (new) start symbol may have a λ-rule.

(Extra:) Give a regular expression that describes the language accepted by each grammar.

A. Grammar G:

$$\begin{aligned} S &\to a\,S \mid b\,S \mid B \\ B &\to b\,b \mid C \mid \lambda \\ C &\to c\,C \mid \lambda \end{aligned}$$

First, we transform the grammar into an equivalent grammar in which the start symbol is no longer recursive.

$$\begin{aligned} S' &\to S \\ S &\to a\,S \mid b\,S \mid B \\ B &\to b\,b \mid C \mid \lambda \\ C &\to c\,C \mid \lambda \end{aligned}$$

The variables with λ-derivation according to the discussed algorithm:

iteration	*null*	*prev*
0	$\{B, C\}$	
1	$\{S, B, C\}$	$\{B, C\}$
2	$\{S', S, B, C\}$	$\{S, B, C\}$
3	$\{S', S, B, C\}$	$\{S', S, B, C\}$

The requested grammar now becomes:

$$\begin{aligned} S' &\to S \mid \lambda \\ S &\to a\,S \mid b\,S \mid B \mid a \mid b \\ B &\to b\,b \mid C \\ C &\to c\,C \mid c \end{aligned}$$

The corresponding regular expression is $(a \cup b)^*(bb \cup c^*)$.

B. Grammar G:

$$\begin{aligned} S &\to A\,B\,C \mid a\,B\,C \\ A &\to a\,A \mid B\,C \\ B &\to b\,B \mid \lambda \\ C &\to c\,C \mid \lambda \end{aligned}$$

The grammar already had a non-recursive start symbol.

$$\begin{aligned} S &\to A\,B\,C \mid a\,B\,C \\ A &\to a\,A \mid B\,C \\ B &\to b\,B \mid \lambda \\ C &\to c\,C \mid \lambda \end{aligned}$$

The variables with λ-derivation:

Iteration	*null*	*prev*
0	$\{B, C\}$	
1	$\{B, C, A\}$	$\{B, C\}$
2	$\{B, C, A, S\}$	$\{B, C, A\}$
3	$\{B, C, A, S\}$	$\{B, C, A, S\}$

The resulting noncontracting grammar:

$$
\begin{aligned}
S &\to A\,B\,C \mid A\,B \mid B\,C \mid A\,C \mid A \mid B \mid C \mid \lambda \mid a\,B\,C \mid a\,B \mid a\,C \mid a \\
A &\to a\,A \mid B\,C \mid B \mid C \mid a \\
B &\to b\,B \mid b \\
C &\to c\,C \mid c
\end{aligned}
$$

The corresponding regular expression is $a^*b^*c^*b^*c^*$.

5.2 Chain rules

For each of the following grammars, construct an equivalent grammar G_C without chain rules. (Note that the grammars have no λ-rules, but you may need to make the start symbol nonrecursive.)

C. Grammar G:

$$
\begin{aligned}
S &\to A\,S \mid A \\
A &\to a\,A \mid b\,B \mid C \\
B &\to b\,B \mid b \\
C &\to c\,C \mid B
\end{aligned}
$$

Adding non-recursive start symbol:

$$
\begin{aligned}
S' &\to S \\
S &\to A\,S \mid A \\
A &\to a\,A \mid b\,B \mid C \\
B &\to b\,B \mid b \\
C &\to c\,C \mid B
\end{aligned}
$$

The chain variables are computed iteratively again:

Iter	$chain(S')$	$chain(S)$	$chain(A)$	$chain(B)$	$chain(C)$
0	$\{S'\}$	$\{S\}$	$\{A\}$	$\{B\}$	$\{C\}$
1	$\{S', S\}$	$\{S, A\}$	$\{A, C\}$	$\{B\}$	$\{C, B\}$
2	$\{S', S, A\}$	$\{S, A, C\}$	$\{A, C, B\}$	$\{B\}$	$\{C, B\}$
3	$\{S', S, A, C\}$	$\{S, A, C, B\}$	$\{A, C, B\}$	$\{B\}$	$\{C, B\}$
4	$\{S', S, A, C, B\}$	$\{S, A, C, B\}$	$\{A, C, B\}$	$\{B\}$	$\{C, B\}$
4	$\{S', S, A, C, B\}$	$\{S, A, C, B\}$	$\{A, C, B\}$	$\{B\}$	$\{C, B\}$

So the grammar G_C is (after removing duplicate right-hand sides):

$$
\begin{aligned}
S' &\to A\,S \mid a\,A \mid b\,B \mid b \mid c\,C \\
S &\to A\,S \mid a\,A \mid b\,B \mid b \mid c\,C \\
A &\to a\,A \mid b\,B \mid c\,C \mid b \\
B &\to b\,B \mid b \\
C &\to c\,C \mid b\,B \mid b
\end{aligned}
$$

D. *(extra)* Grammar G:

$$\begin{aligned} S &\to A\,B \mid C \\ A &\to a\,A \mid B \\ B &\to b\,B \mid C \\ C &\to c\,C \mid a \mid A \end{aligned}$$

The grammar already has a non-recursive start symbol.

$$\begin{aligned} S &\to A\,B \mid C \\ A &\to a\,A \mid B \\ B &\to b\,B \mid C \\ C &\to c\,C \mid a \mid A \end{aligned}$$

The chain variables are:

$$\begin{aligned} chain(S) &= \{S, C, A, B\} \\ chain(A) &= \{A, B, C\} \\ chain(B) &= \{B, C, A\} \\ chain(C) &= \{C, A, B\} \end{aligned}$$

So the grammar G_C is:

$$\begin{aligned} S &\to A\,B \mid a\,A \mid b\,B \mid cC \mid a \\ A &\to a\,A \mid b\,B \mid c\,C \mid a \\ B &\to b\,B \mid a\,A \mid c\,C \mid a \\ C &\to c\,C \mid a \mid a\,A \mid b\,B \end{aligned}$$

It is easy to see that this is equivalent to

$$\begin{aligned} S &\to A\,A \mid a\,A \mid b\,A \mid c\,A \mid a \\ A &\to a\,A \mid b\,A \mid c\,A \mid a \end{aligned}$$

5.3 Useless nonterminals

For each of the following grammars, construct an equivalent grammar G_T without nonterminating symbols, and then an equivalent grammar G_U without unreachable (and without nonterminating) symbols. Use the algorithms given in the lecture (and in the book).

E. Grammar G:

$$\begin{aligned} S &\to A\,A \mid C\,D \mid b\,B \\ A &\to a\,A \mid a \\ B &\to b\,B \mid b\,C \\ C &\to c\,B \\ D &\to d\,D \mid d \end{aligned}$$

Given the grammar

$$\begin{aligned} S &\to A\,A \mid C\,D \mid b\,B \\ A &\to a\,A \mid a \\ B &\to b\,B \mid b\,C \\ C &\to c\,B \\ D &\to d\,D \mid d \end{aligned}$$

The terminating variables can be determined using the strategy from the book.

Iteration	*term*	*prev*
0	$\{A, D\}$	
1	$\{A, D, S\}$	$\{A, D\}$
2	$\{A, D, S\}$	$\{A, D, S\}$

So B and C are non-terminating. This results in

$$\begin{aligned} S &\to A\,A \\ A &\to a\,A \mid a \\ D &\to d\,D \mid d \end{aligned}$$

Now the following applies

Iteration	*reach*	*prev*
0	$\{S\}$	
1	$\{S, A\}$	$\{S\}$
2	$\{S, A\}$	$\{S, A\}$

So D is not reachable; we get

$$\begin{aligned} S &\to A\,A \\ A &\to a\,A \mid a \end{aligned}$$

F. *(extra)* Grammar G:

$$\begin{aligned} S &\to A\,C\,H \mid B\,B \\ A &\to a\,A \mid a\,F \\ B &\to C\,F\,H \mid b \\ C &\to a\,C \mid D\,H \\ D &\to a\,D \mid B\,D \mid C\,a \\ F &\to b\,B \mid b \\ H &\to d\,H \mid d \end{aligned}$$

Given the grammar

$$\begin{aligned} S &\to A\,C\,H \mid B\,B \\ A &\to a\,A \mid a\,F \\ B &\to C\,F\,H \mid b \\ C &\to a\,C \mid D\,H \\ D &\to a\,D \mid B\,D \mid C\,a \\ F &\to b\,B \mid b \\ H &\to d\,H \mid d \end{aligned}$$

The terminating variables:

Iteration	*term*	*prev*
0	$\{B, F, H\}$	
1	$\{B, F, H, S, A\}$	$\{B, F, H\}$
2	$\{B, F, H, S, A\}$	$\{B, F, H, S, A\}$

So C and D are non-terminating. This results in

$$\begin{aligned} S &\to B\,B \\ A &\to a\,A \mid a\,F \\ B &\to b \\ F &\to b\,B \mid b \\ H &\to d\,H \mid d \end{aligned}$$

Now the following applies

Iteration	*reach*	*prev*
0	$\{S\}$	
1	$\{S, B\}$	$\{S\}$
2	$\{S, B\}$	$\{S, B\}$

So A, F and H are not reachable. This results in:

$$\begin{aligned} S &\to B\,B \\ B &\to b \end{aligned}$$

5.4 Finalisation of CNF rules

For each of the following grammars, construct an equivalent one in Chomsky normal form (CNF). (Note that the grammars already have a nonrecursive start symbol, are essentially noncontracting, and contain neither chain rules nor useless symbols.)

G. Grammar G:

$$\begin{aligned} S &\to aA \mid ABa \\ A &\to AA \mid a \\ B &\to AbB \mid bb \end{aligned}$$

Given the grammar

$$\begin{aligned} S &\to aA \mid ABa \\ A &\to AA \mid a \\ B &\to AbB \mid bb \end{aligned}$$

The Chomsky normal form is as follows

$$\begin{aligned} S &\to A'A \mid AT_1 \\ A' &\to a \\ T_1 &\to BA' \\ A &\to AA \mid a \\ B &\to AT_2 \mid B'B' \\ T_2 &\to B'B \\ B' &\to b \end{aligned}$$

H. Grammar G:

$$\begin{aligned} S &\rightarrow aAbB \mid ABC \mid a \\ A &\rightarrow aA \mid a \\ B &\rightarrow bBcC \mid b \\ C &\rightarrow abc \end{aligned}$$

Given the grammar

$$\begin{aligned} S &\rightarrow aAbB \mid ABC \mid a \\ A &\rightarrow aA \mid a \\ B &\rightarrow bBcC \mid b \\ C &\rightarrow abc \end{aligned}$$

The Chomsky normal form is as follows

$$\begin{aligned} S &\rightarrow A'T_1 \mid AT_3 \mid a \\ A' &\rightarrow a \\ T_1 &\rightarrow AT_2 \\ T_2 &\rightarrow B'B \\ B' &\rightarrow b \\ T_3 &\rightarrow BC \\ A &\rightarrow A'A \mid a \\ B &\rightarrow B'T_4 \mid b \\ T_4 &\rightarrow BT_5 \\ T_5 &\rightarrow C'C \\ C' &\rightarrow c \\ C &\rightarrow A'T_6 \\ T_6 &\rightarrow B'C' \end{aligned}$$

Note: The right-hand sides of the grammar in Exercise G have at most three symbols. The main difference in Exercise H is that you have to deal with right-hand sides of more than three symbols. During the tutorial, after doing G, we recommend that you start with H, but as soon as you feel that you have understood the principle, move on to the next exercise.

5.5 The CYK parsing algorithm

I. Given the context-free (CNF) grammar $G = (V, \Sigma, P, S)$ with $V = \{S, A, B, C, D, E\}$, $\Sigma = \{a, b, c\}$ and P consisting of the rules

$$\begin{aligned} S &\rightarrow CA \mid BD \\ A &\rightarrow a \\ B &\rightarrow b \mid c \\ C &\rightarrow AE \\ D &\rightarrow EB \mid b \\ E &\rightarrow BD \mid CA \end{aligned}$$

(a) Construct the X_{ij}-matrix for the words $acba$ and $baca$ according to the CYK-algorithm.

(b) Use the X_{ij}-matrix to determine whether $acba$ and $baca$ belong to $\mathcal{L}(G)$, and if yes, what the corresponding derivation tree is.

(c) *(extra)* Would you call the CYK-algorithm a "top-down" or a "bottom-up" parsing algorithm, or neither? Motivate your answer.

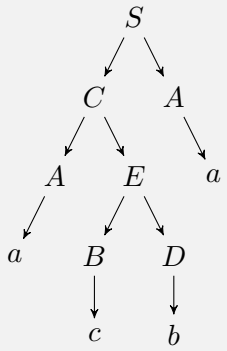
a. The X_{ij}-matrices for *acba* and *baca* according to CYK:

	1	2	3	4
1	$\{A\}$	$-$	$\{C\}$	$\{S, E\}$
2		$\{B\}$	$\{S, E\}$	$-$
3			$\{B, D\}$	$-$
4				$\{A\}$

	1	2	3	4
1	$\{B, D\}$	$-$	$-$	$-$
2		$\{A\}$	$-$	$-$
3			$\{B\}$	$-$
4				$\{A\}$

So $baca$ is not recognized!

b. Because S is on the top right of the matrix for *acba*, this word belongs to the language $\mathcal{L}(G)$. This is not the case for *baca*. The derivation tree for *acba* can be read directly from the matrix, using the corresponding rules:



c. The decomposition algorithm follows a bottom-up approach. Indeed, we start with the decomposition of all substrings of length 1, next all substrings of length 2, etc., until finally, we arrive at the string itself.

J. *(extra)* Given the context-free grammar $G = (V, \Sigma, P, S)$ with $V = \{S, A, B, C, D\}$, $\Sigma = \{a, b, c\}$ and P consisting of the rules

$$\begin{aligned} S &\to A\,C \mid D\,B \\ A &\to a \\ B &\to b \mid c \\ C &\to S\,A \\ D &\to B\,S \mid b \end{aligned}$$

(You can ignore the recursion in the start symbol S for this exercise.)

(a) Construct for the words $abca$ and $acab$ the X_{ij}-matrix.

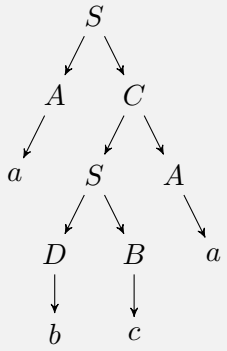
(b) Use the X_{ij}-matrix to determine whether $abca$ and $acab$ belong to $\mathcal{L}(G)$, and if yes, what the corresponding derivation tree is.

a. The X_{ij}-matrices for *abca* and *acab* according to CYK:

	1	2	3	4
1	$\{A\}$	$-$	$-$	$\{S\}$
2		$\{B, D\}$	$\{S\}$	$\{C\}$
3			$\{B\}$	$-$
4				$\{A\}$

	1	2	3	4
1	$\{A\}$	$-$	$-$	$-$
2		$\{B\}$	$-$	$-$
3			$\{A\}$	$-$
4				$\{B, D\}$

b. Because S is on the top right of the matrix for *abca*, this word belongs to the language $\mathcal{L}(G)$. This is not the case for *acab*. The derivation tree for *abca* can be read directly from the matrix, using the corresponding rules:



K. *(extra)* Give the upper diagonal matrix produced by the CYK algorithm when run with the Chomsky normal form grammar of G below and the input strings *abbb* and *aabbb*.

$$S \rightarrow aSb \mid ab$$

For convenience, the (CNF) grammar of G is given here:

$$\begin{aligned} S &\rightarrow AT \mid AB \\ T &\rightarrow XB \\ X &\rightarrow AT \mid AB \\ A &\rightarrow a \\ B &\rightarrow b \end{aligned}$$

The X_{ij}-matrices for $abbb$ and $aabbb$:

	1	2	3	4
1	$\{A\}$	$\{S, X\}$	$\{T\}$	$-$
2		$\{B\}$	$-$	$-$
3			$\{B\}$	$-$
4				$\{B\}$

	1	2	3	4	5
1	$\{A\}$	$-$	$-$	$\{S, X\}$	$\{T\}$
2		$\{A\}$	$\{S, X\}$	$\{T\}$	$-$
3			$\{B\}$	$-$	$-$
4				$\{B\}$	$-$
5					$\{B\}$

So neither of these words are accepted. Looking at the original grammar and language of G, we can see this directly: the language was $\{a^i b^i \mid i \geq 1\}$, and neither $abbb$ nor $aabbb$ are elements of this.