

Exercises on Chomsky Normal Form and CYK parsing

1. Convert the following grammar to CNF

$S \rightarrow A(S)B \mid ""$

$A \rightarrow S \mid SB \mid x \mid ""$

$B \rightarrow SB \mid y$

This exercise is available in the “grammar tutoring system”

<http://laraserver3.epfl.ch:9000>

- Select exercise type “CNF Conversion” -> choose the problem “Exercise 1 of lecture exercise 12”
- Make sure you create a new start symbol S_1 and add the production $S_1 \rightarrow S \mid ""$ to your grammar before CNF conversion as the start symbol ‘ S ’ is nullable and also appears on the right hand side

Exercise 2

Which of the following properties of a grammar are preserved by the “Epsilon Production Elimination” algorithm

a. **Ambiguity** No, counter-example ?

If a grammar is “ambiguous” does it remain ambiguous after removing epsilon productions ?

b. **LL(1)** No, counter-example ?

c. **No Left Recursion** No, counter-example ?

We define left recursion as existence of productions of the form

– $N \rightarrow N \alpha$ (or)

– $N \rightarrow N_1 \alpha_1, N_1 \rightarrow N_2 \alpha_2, \dots, N_n \rightarrow N \alpha_n$

d. **Unambiguity** ? Yes, proof ?

Exercise 2(a) - Solution

- **Ambiguity** : If a grammar is “ambiguous” does it remain ambiguous after removing epsilon productions ?
- No, it need not. Eg. consider the following ambiguous grammar
 - $S \rightarrow aA \mid a$
 - $A \rightarrow b \mid \epsilon$
 - The grammar has two parse trees for the string: a
- **After removing epsilon productions we get**
 - $S \rightarrow aA \mid a$
 - $A \rightarrow b$
 - There is exactly one parse tree for every word generated by the grammar (which are a and ab)

Exercise 2(b) - Solution

- LL(1): If a grammar is LL(1) does it remain LL(1) after removing epsilon productions ?
- No, it need not. Eg. consider the following LL(1) grammar
 - $S \rightarrow a S b \mid \epsilon$
- After removing epsilon productions we get
 - $S' \rightarrow S \mid \epsilon$
 - $S \rightarrow a S b \mid a b$
 - Note: we have created a new start symbol S' as S was nullable and appeared on the right hand side
 - The grammar is not LL(1) as $\text{First}(a S b)$ and $\text{First}(a b)$ intersect

Exercise 2(c) - Solution

- No left recursion: If a grammar has no left recursion does it remain without left recursion after removing epsilon productions ?
- No, it need not. Eg. consider the following non-left recursive grammar
 - $S \rightarrow B S a \mid a$
 - $B \rightarrow \epsilon \mid b$
- After removing epsilon productions we get
 - $S \rightarrow B S a \mid S a \mid a$
 - $B \rightarrow \epsilon \mid b$
 - The production $S \rightarrow S a$ is a left recursive production
- Note: this also means that we have to eliminate epsilons before removing left recursion using the approach described in lecture 10

Exercise 2(d) - Solution

Unambiguity: If a grammar is unambiguous does it remain unambiguous after removing epsilon productions ?

- Yes. Proof: Let G' be obtained from G by eliminating epsilon productions

Let's prove the contra-positive form: if there are two left most derivations for a word w in G' then there will be two left most derivations for w in G

- Let $D1$ and $D2$ be two left most derivations for the word w in G'
- If all the productions used in $D1$ and $D2$ are also present in G then $D1$ and $D2$ are also valid in G , which will imply the claim.
- Therefore, say $D1$ or $D2$ use productions denoted using: $A_i \rightarrow \gamma_j$ that did not belong to G .
- For each $A_i \rightarrow \gamma_j$ (except when A_i is a start symbol), there exists $A_i \rightarrow \alpha_j$ belonging to G such that γ_j is obtained from α_j by removing nullable non-terminals from some positions (denoted by say \bar{X}_j).
- Hence, we can create new derivations $D1'$ and $D2'$ (that are valid in G) from $D1$ and $D2$ by replacing every use of the rule $A_i \rightarrow \gamma_j$ by $A_i \rightarrow \alpha_j$ and deriving empty strings (using productions of G) from non-terminals in positions \bar{X}_j

Exercise 2(d) - Solution

- If A_i was the start symbol of G' and it does not belong to G then we remove the derivation $A_i \rightarrow S$ at the start of $D1$ or $D2$ and replace it with the start symbol S of G
- Will the resulting derivations $D1'$ and $D2'$ be distinct ?
- Consider the point where $D1$ and $D2$ diverge. Let
 - $D1: S \Rightarrow^* xB\alpha \Rightarrow x\beta_1\alpha \Rightarrow^* w$
 - $D2: S \Rightarrow^* xB\alpha \Rightarrow x\beta_2\alpha \Rightarrow^* w$
- If neither $B \rightarrow \beta_1$ or $B \rightarrow \beta_2$ are of the form $A_i \rightarrow \gamma_j$ then $D1'$ and $D2'$ will also diverge at the same point as we preserve $B \rightarrow \beta_1$ and $B \rightarrow \beta_2$ in $D1'$, $D2'$ respec.
- Case (i), say
 - $D1: S \Rightarrow^* xA_i\alpha \Rightarrow x\gamma_j\alpha \Rightarrow^* w$
 - $D2: S \Rightarrow^* xA_i\alpha \Rightarrow x\beta\alpha \Rightarrow^* w$, where $A_i \rightarrow \beta$ is a rule in G as well
- γ_j is replaced by α_j in $D1'$ and some non-terminals in α_j derive epsilon. However, no non-terminal in β derives epsilon in $D2$ as $D2$ is a derivation of G' that has no epsilon productions except for the start symbol. By construction, the start symbol of G' will never appear on the right hand side if it is nullable.
- Hence, $A_i \rightarrow \beta$ will be preserved in $D2'$ and no non-terminal in β derives epsilon in $D2'$. Hence, $D1'$ will differ from $D2'$ irrespective of whether $\alpha_j = \beta$ or not.

Exercise 2(d) - Solution

- Case (ii), say
 - D1: $S \Rightarrow^* xA_i\alpha \Rightarrow x\gamma_j\alpha \Rightarrow^* w$
 - D2: $S \Rightarrow^* xA_i\alpha \Rightarrow x\gamma_k\alpha \Rightarrow^* w$
- γ_j is replaced by α_j in D1' and γ_k by α_k in D2'. Since $\gamma_j \neq \gamma_k$ either α_j and α_k are different or non-terminals at different positions in α_j and α_k are reduced to empty strings in D1' and D2', respectively. Hence, D1' will differ from D2'

Exercise 3

Which of the following properties of a grammar are preserved by the “Unit Production Elimination” algorithm

- Ambiguity No, counter-example ?
- Left Recursion No
 - What about these rules: $B \rightarrow A \mid a$, $A \rightarrow B$?
- LL(1) Yes, proof ?
- Unambiguity Yes, proof ?

Exercise 3(a) - Solution

- **Ambiguity** : If a grammar is “ambiguous” does it remain ambiguous after removing unit productions ?
- No, it need not. Eg. consider the following ambiguous grammar
 - $S \rightarrow A \mid a$
 - $A \rightarrow a$
 - There exists two parse tree for a: $S \rightarrow A \rightarrow a$ and $S \rightarrow a$
- **After removing the unit production we get**
 - $S \rightarrow a$
 - There is exactly one parse tree for “a”

Exercise 3(b) - Solution

- Left recursion: If a grammar has left recursion will it have left recursion after removing unit productions ?
- No, it need not. Eg. consider the following left recursive grammar
 - $B \rightarrow A \mid a$
 - $A \rightarrow B$
- After removing the unit productions (using the graph based algorithm described in lecturecise 11) we get
 - $B \rightarrow a$
 - $A \rightarrow a$

Exercise 3(c) - Solution

- LL(1): If a grammar is LL(1) does it remain LL(1) after removing unit productions ?
- Yes. The following is a sketch of the proof (not a complete, rigorous proof).
- Let G' be obtained from G by eliminating unit productions. We need to show that all 3 properties of LL(1) grammar holds in G' if it holds in G
- Assume that only one unit production $A \rightarrow B$ (where $B \rightarrow \beta_1 \mid \dots \mid \beta_n$) is removed
- If there are many unit productions, say we remove them one after the other. Hence, it suffices to show that the property holds for this step.
- $A \rightarrow B$ will be replaced by $A \rightarrow \beta_1 \mid \dots \mid \beta_n$
 1. Every alternative of A in G' will have disjoint first sets. If there is an alternative whose first set intersects with β_i for some i then in G it would have intersected with $\text{first}(B)$ contradicting the fact that the input grammar is LL(1)
 2. A will have at most one nullable alternative in G' . If in G , A 's nullable alternative was B then there will exist exactly one β_i that is nullable in G (and hence in G') as G is in LL(1). If not, no β_i will be nullable in G (and hence in G').

Exercise 3(c) – Solution [Cont.]

3. By the same argument, every non-terminal that is nullable in G' is also nullable in G .
4. We now show that the follow set of every non-terminal in G' can only become smaller than its corresponding follow set in G .
5. Adding the production $A \rightarrow \beta_1 \mid \dots \mid \beta_n$ can only affect the follow set of every non-terminal N in β_i s.t. $\beta_i = \alpha N \gamma$ and either γ is empty or is nullable.
6. In this case, $\text{follow}(A) \subseteq \text{follow}(N)$ in G' . However, in G , $\text{follow}(A) \subseteq \text{follow}(B) \subseteq \text{follow}(N)$ as $A \rightarrow B$ and $B \rightarrow \beta_i$ belong to G and γ is nullable in G' implies it is also nullable in G . Hence, $\text{follow}(N)$ cannot increase for any such N .
7. Removing the production $A \rightarrow B$ can only reduce the size of $\text{follow}(B)$ in G' compared to its size in G .
8. Any reduction in the follow set of the non-terminals cannot violate the LL(1) property. Hence, G' is also in LL(1)

Exercise 3(d) - Solution

Unambiguity: If a grammar is unambiguous does it remain unambiguous after removing unit productions ?

- Yes. Sketch of the proof: Let G' be obtained from G by eliminating epsilon productions
- Let's prove the contra-positive form: if there are two left most derivations for a word w in G' then there will be two left most derivations for w in G
 - As in previous proof assume that we are removing only one unit production $A \rightarrow B$ where $B \rightarrow \beta_1 \mid \dots \mid \beta_n$. $A \rightarrow B$ will be replaced by $A \rightarrow \beta_1 \mid \dots \mid \beta_n$
 - Let $D1$ and $D2$ be two left most derivations for the word w in G'
 - If all the productions used in $D1$ and $D2$ are present in G then $D1$ and $D2$ are also valid in G , which will imply the claim.
 - Therefore, say $D1$ or $D2$ use productions denoted using: $A \rightarrow \beta_i$ that did not belong to G .
 - For each $A \rightarrow \beta_i$, there exists a derivation $A \rightarrow B \rightarrow \beta_i$ belonging to G .
 - Hence, we can create new derivations $D1'$ and $D2'$ (that are valid in G) from $D1$ and $D2$ by replacing every use of the rule $A \rightarrow \beta_i$ by the derivation $A \rightarrow B \rightarrow \beta_i$
 - Are $D1'$ and $D2'$ distinct ?

Exercise 3(d) - Solution [Cont.]

- Consider the point where D1 and D2 diverge. Let
 - D1: $S \Rightarrow^* xC\alpha \Rightarrow x\gamma_1\alpha \Rightarrow^* w$
 - D2: $S \Rightarrow^* xC\alpha \Rightarrow x\gamma_2\alpha \Rightarrow^* w$
- If neither of $C \rightarrow \gamma_1$ and $C \rightarrow \gamma_2$ are of the form $A \rightarrow \beta_i$ then D1' and D2' will also diverge at the same point as we do not change such productions
- Case (i), say
 - D1: $S \Rightarrow^* xA\alpha \Rightarrow x\beta_i\alpha \Rightarrow^* w$
 - D2: $S \Rightarrow^* xA\alpha \Rightarrow x\gamma_2\alpha \Rightarrow^* w$
- In this case D1' will be $S \Rightarrow^* xA\alpha \Rightarrow xB\alpha \Rightarrow x\beta_i\alpha \Rightarrow^* w$ and D2' will have $S \Rightarrow^* xA\alpha \Rightarrow x\gamma_2\alpha$. Moreover, γ_2 is not B as $A \rightarrow B$ does not belong to G' . Hence, D1' will differ from D2'
- Case (ii), say
 - D1: $S \Rightarrow^* xA_i\alpha \Rightarrow x\beta_i\alpha \Rightarrow^* w$
 - D2: $S \Rightarrow^* xA_i\alpha \Rightarrow x\beta_j\alpha \Rightarrow^* w$
- In this case D1': $S \Rightarrow^* xA\alpha \Rightarrow xB\alpha \Rightarrow x\beta_i\alpha \Rightarrow^* w$ and D2': $S \Rightarrow^* xA\alpha \Rightarrow xB\alpha \Rightarrow x\beta_j\alpha \Rightarrow^* w$. Hence, D1' will differ from D2'

Exercise 4

Given a grammar G in CNF, how many steps does it require to derive a string of size n .

Exercise 4 -Solution

Intuition

Consider a derivation of a string of length n obtained as follows:

1. Derive a string of exactly n nonterminals from the start symbol, then
 2. Expand each nonterminal out to a single terminal.
- To obtain 'n' non-terminals from the start symbol, we need to apply productions of the form $S \rightarrow AB$ as that is the only way to generate non-terminals. **How many times do we have to apply such productions ?**
 - Application of one such production will increase the number of nonterminals by 1, since you replace one nonterminal with two nonterminals.
 - Since we start with one nonterminal, we need to repeat this $n-1$ times.
 - We need n more steps to convert nonterminals to terminals
 - Therefore, total number of steps = $2n - 1$
 - **Let's try to prove this bound formally**

Exercise 4 –Solution [Cont.]

- Denote the number of steps required to derive a string w from a non-terminal N as $NS(N, w)$
- If $|w| = 1$, we need exactly 1 step. $NS(N, w) = 1$ if $|w| = 1$
- If $|w| > 1$, we first apply a production $N \rightarrow N_1 N_2$ which will derive w .
- Say $N \Rightarrow^* N_1 N_2 \Rightarrow^* x N_2 \Rightarrow^* xy = w$,
 - where $|x| = q$ (say) and $|y| = |w| - q$
- Hence, $NS(N, w) = NS(N_1, x) + NS(N_2, y) + 1$
- We need a closed form for $NS(N, w)$ that depends only on $|w|$. Say $NS(N, w) = f(|w|)$
- We have, $f(|w|) = \begin{cases} 1 & \text{if } |w| = 1 \\ f(q) + f(|w| - q) + 1 & \text{if } |w| > 1, 1 \leq q < |w| \end{cases}$
- Where $2|w| - 1$ is the solution for the above recurrence

Exercise 5

Assume a grammar in CNF has n non-terminals. Show that if the grammar can generate a word with a derivation having at least 2^n steps, then the recognized language should be infinite

(see the pdf file uploaded in the lara wiki along with the slides)

Exercise 6

Show the CYK parsing table for the string “aabbab”
for the grammar

$S \rightarrow AB \mid BA \mid SS \mid AC \mid BD$

$A \rightarrow a$

$B \rightarrow b$

$C \rightarrow SB$

$D \rightarrow SA$

What should be done to construct a parse tree for the
string

Exercise 6 -Solution

a a b b a b
0 1 2 3 4 5

	0	1	2	3	4	5
0	A	-	-	S	D	S
1		A	S	C	S	C
2			B	-	-	-
3				B	S	C
4					A	S
5						B

- For generating parse trees, modify the parse table d as below
- Make every entry (i,j) , $i < j$, of the table a set of triples (N, s, p) where N accepts the sub-string from index i to j via a production of the form $p: N \rightarrow N_1 N_2$ and N_1 accepts the substring from index i to $(i+s-1)$ and N_2 accepts the substring from index $(i+s)$ to j

Exercise 6 –Solution [Cont.]

$S \rightarrow AB$ (p1) | BA (p2) | SS (p3) | AC (p4) | BD (p5)

$A \rightarrow a$ (p6)

$B \rightarrow b$ (p7)

$C \rightarrow SB$ (p8)

$D \rightarrow SA$ (p9)

	0	1	2	3	4	5
0	A	-	-	(S,1,p4)	(D,4,p9)	(S,4,p3)
1		A	(S,1,p1)	(C,2,p8)	(S,2,p3)	(C,4,p8)
2			B	-	-	-
3				B	(S,1,p2)	(C,2,p8)
4					A	(S,1,p1)
5						B

See next slide for an algorithm for generating one parse tree given a table of the above form

Exercise 6 –solution [Cont.]

Algorithm for generating one parse tree starting from a nonterminal N for a sub-string (i,j)

ParseTree(N, i, j)

- If $i = j$, if N is in parseTable(i,j) return Leaf($N, w(i,j)$) else report parse error
- Otherwise, pick an entry (N, s, p) from parseTable(i,j)
- If no such entry exist report that the sub-string cannot be parsed and return
- Let $p: N \rightarrow N_1 N_2$
- leftChild = ParseTree($N_1, i, i+s-1$)
- rightChild = ParseTree($N_2, i+s, j$)
- Return Node($N, leftChild, rightChild$)