Problem Set for Tutorial 6 — TDDD14/TDDD85

1 Context-Free Grammars

Exercise 1. Consider the CFG $G = (\{E, T, F\}, \{a, b, c, +, -, \cdot, /, (,)\}, P, E)$, where P comprises the productions

$$
\begin{aligned}
&E \to T \mid E + T \mid E - T \\
&T \to F \mid T \cdot F \mid T/F \\
&F \to a \mid b \mid c \mid (E)
\end{aligned}
$$

Find the derivation trees for the following strings.

1. $a \cdot b + c$
2. $a + a - b \cdot (a/b + b/c)$

Exercise 2. Find CFGs which generate the following languages.

1. All strings in $\{0, 1\}^*$ for which every 0 is followed by 1 immediately to the right.
2. All strings in $\{0, 1\}^*$ which are palindromes.
3. $\{0^n 1^n \mid n \geq 0\}$
4. All string in $\{a, b\}^*$ containing at least one a and one b, such that the number of a's preceding the first b is the same as the number of b's following the last a.

Exercise 3. Consider the CFG $G = (\{S, A, B\}, \{a, b\}, P, S)$, where P comprises the productions

$$
\begin{aligned}
&S \to aB \mid bA \\
&A \to a \mid aS \mid bAA \\
&B \to b \mid bS \mid aBB
\end{aligned}
$$

Show that G is ambiguous.

Exercise 4. Let G be a CFG consisting of the following productions (S is the start symbol):

$$
\begin{aligned}
&S \to AB \\
&A \to SA \mid BB \mid bB \\
&B \to b \mid aA \mid \epsilon
\end{aligned}
$$

Find an equivalent CFG with a single ϵ-production $S \to \epsilon$, and without unit productions.

Exercise 5. Find equivalent Chomsky normal-form CFGs for the two CFGs below (S is the start symbol in both cases).

1. $S \to \neg S \mid (S \supset S) \mid p \mid q$

2. $S \to A \mid ABA$
   $A \to aA \mid B \mid a$
   $B \to bB \mid b$

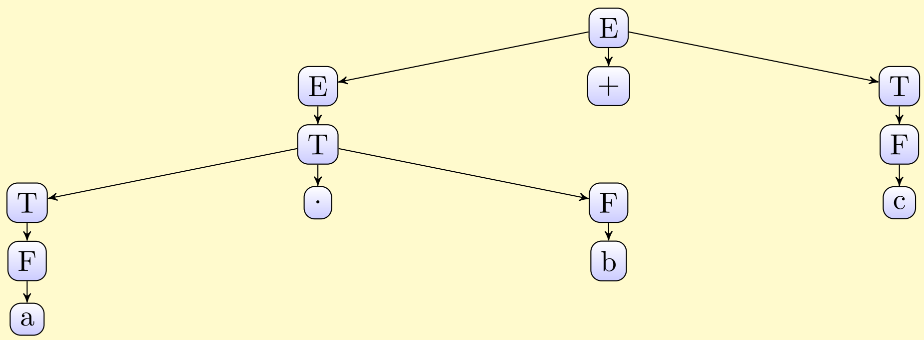
Solutions

Solution to Exercise 1. Our task is to find the derivation trees for the strings $a \cdot b + c$ and $a + a - b \cdot (a/b + b/c)$. We present a detailed solution for the first case and leave the second as an exercise.

A derivation for the string $a \cdot b + c$ is:

$$E \Rightarrow E + T \Rightarrow T + T \Rightarrow T \cdot F + F \Rightarrow F \cdot F + F \Rightarrow a \cdot F + F \Rightarrow a \cdot b + F \Rightarrow a \cdot b + c.$$

The corresponding derivation tree is:



Solution to Exercise 2. S is the start symbol in all grammars below.

1. $S \Rightarrow 1S \mid 01S \mid \epsilon$
2. $S \Rightarrow \epsilon \mid 0 \mid 1 \mid 0S0 \mid 1S1$
3. $S \Rightarrow 0S1 \mid \epsilon$
4. $S \rightarrow aSb \mid ab \mid bAa$
   $A \rightarrow \epsilon \mid aA \mid bA$

   Justification: Any string over $\{a, b\}$ can be generated from A. Productions $S \rightarrow ab \mid bAa$ generate the first b and the last a (and the number of a's preceding the first b is the same as the number of b's following the last a, it is 1 for the first production and 0 for the second). Production $S \rightarrow aSb$ adds one a preceding the first b and one b following the last a.

Solution to Exercise 3. The string $aabbab$ has two distinct left derivations:
$S \Rightarrow aB \Rightarrow aaBB \Rightarrow aabSB \Rightarrow aabbAB \Rightarrow aabbaB \Rightarrow aabbab$
$S \Rightarrow aB \Rightarrow aaBB \Rightarrow aabB \Rightarrow aabbS \Rightarrow aabbaB \Rightarrow aabbab$

Solution to Exercise 4. We follow the method from the lecture. First we add productions to P in order to obtain the smallest $P_1 \supseteq P$ such that

(a) if $A \to \alpha B \beta$ and $B \to \epsilon$ are in P_1 then $A \to \alpha\beta$ is in P_1.

Any nonempty terminal string derived from S in G can be derived in (N, Σ, P_1, S) without using any ϵ-production. So we can remove the ϵ-productions from P_1, obtaining P_1'.

Now we add productions to P_1' in order to obtain the smallest $P_2 \supseteq P_1'$ such that

(b) if $A \to B$ and $B \to \gamma$ are in P_2 then $A \to \gamma$ is in P_2.

Any terminal string derived from S in (N, Σ, P_1', S) can be derived in (N, Σ, P_2, S) without using any unit production. Thus we can remove the unit productions from P_2, obtaining P_2'.

$G' = (N, \Sigma, P_2', S)$ is the result, $L(G') = L(G) - \{\epsilon\}$. (Notice that in [Kozen] the rules (a) and (b) are applied together. Doing this separately, as above, is also correct.)

For the given grammar new productions are added as follows. In order to remove ϵ-productions:

production	with	production	gives	production
$B \to \epsilon$		$S \to AB$		$S \to A$
		$A \to BB$		$A \to B$
		$A \to bB$		$A \to b$
		$A \to B$		$A \to \epsilon$
$A \to \epsilon$		$S \to AB$		$S \to B$
		$A \to SA$		$A \to S$
		$B \to aA$		$B \to a$
		$S \to A$		$S \to \epsilon$
$S \to \epsilon$		$A \to SA$		$A \to A$
		$A \to S$		$A \to \epsilon$
$B \to \epsilon$		$S \to B$		$S \to \epsilon$

The obtained set P_1' of productions is:

$$\begin{aligned}
&S \to AB \mid A \mid B \\
&A \to BB \mid B \mid bB \mid b \mid SA \mid S \\
&B \to aA \mid a \mid b
\end{aligned}$$

To get rid of unit productions:

production	with	production	gives	production
$S \to A$		$A \to BB$		$S \to BB$
		$A \to B$		$S \to B$
		$A \to bB$		$S \to bB$
		$A \to b$		$S \to b$
		$A \to SA$		$S \to SA$
		$A \to S$		$S \to S$
$S \to B$		$B \to aA$		$S \to aA$
		$B \to a$		$S \to a$
		$B \to b$		$S \to b$
$A \to B$		$B \to aA$		$A \to aA$
		$B \to a$		$A \to a$
		$B \to b$		$A \to b$
$A \to S$		$S \to AB$		$A \to AB$
		$S \to A$		$A \to A$
		$S \to B$		$A \to B$

The obtained set P_2' of productions is:

$$\begin{aligned} &S \to AB \mid BB \mid bB \mid b \mid SA \mid aA \mid a \\ &A \to AB \mid BB \mid bB \mid b \mid SA \mid aA \mid a \\ &B \to aA \mid a \mid b \end{aligned}$$

As we want to obtain a grammar equivalent to the initial one, the removed production $S \to \epsilon$ has to be added.

Solution to Exercise 5.

1. Introduce productions for each terminal symbol which does not occur on its own on the right hand side of some production, i.e.:

   $$\begin{aligned} &A \to \neg \\ &B \to (\\ &C \to \supset \\ &D \to) \end{aligned}$$

   Then replace all such terminal symbols in the original grammar with the corresponding nonterminal from the productions above.

   $$\begin{aligned} &S \to AS \mid BSCSD \mid p \mid q \\ &A \to \neg \\ &B \to (\\ &C \to \supset \end{aligned}$$

$D \rightarrow)$

The only production above which is not in Chomsky normal-form is $S \rightarrow BSCSD$. We can systematically rewrite this production into a set of productions in Chomsky normal-form as follows:

$S \rightarrow BSCSD$ is replaced by $S \rightarrow BE$ and $E \rightarrow SCSD$
$E \rightarrow SCSD$ is replaced by $E \rightarrow SF$ and $F \rightarrow CSD$
$F \rightarrow CSD$ is replaced by $R \rightarrow CG$ and $G \rightarrow SD$

Thus an equivalent Chomsky normal-form grammar is obtained:

$S \rightarrow AS \mid BE \mid p \mid q$
$E \rightarrow SF$
$F \rightarrow CG$
$G \rightarrow SD$
$A \rightarrow \neg$
$B \rightarrow ($
$C \rightarrow \supset$
$D \rightarrow)$

2. First eliminate all unit productions. This yields

$S \rightarrow ABA \mid aA \mid a \mid bB \mid b$
$A \rightarrow aA \mid a \mid bB \mid b$
$B \rightarrow bB \mid b$

We then proceed as in the previous exercise: productions for terminal symbols are introduced where necessary and productions with right hand sides comprising three or more nonterminals are systematically rewritten into a set of productions in Chomsky normal-form. This results in the following grammar:

$S \rightarrow AE \mid CA \mid DB \mid a \mid b$
$A \rightarrow CA \mid DB \mid a \mid b$
$B \rightarrow DB \mid b$
$E \rightarrow BA$
$C \rightarrow a$
$D \rightarrow b$

2 Pushdown Automata

Exercise 6. Consider the following PDA M, where

States: $\{q_0, q_1, q_2\}$

Input alphabet: $\{a, b\}$

Stack alphabet: $\{a, \perp\}$

Initial state: q_0

Initial stack symbol: $\perp$

Final states: $\{q_2\}$

and the transition relation is

$$\begin{aligned}
\delta = \{ \quad & ((q_0, a, \perp), (q_0, a\perp)), \\
& ((q_0, a, a), (q_0, aa)), \\
& ((q_0, b, a), (q_1, \epsilon)), \\
& ((q_1, b, a), (q_1, \epsilon)), \\
& ((q_1, \epsilon, \perp), (q_2, \epsilon)) \quad \}
\end{aligned}$$

Find the configurations that describe the actions of the automaton when the following strings are used as input. For each string, also state whether M accepts it by 1) final state and 2) empty stack.

1. aa
2. $aabba$
3. $aaabbb$

Is M a deterministic pushdown automaton? In other words, is its next configuration relation a (partial) function?

Exercise 7. Construct a deterministic PDA accepting the language $\{a^i b^j \mid 0 \leq i < j\}$.

Exercise 8. For each of the following CFG's, construct a PDA which accepts the language generated by the CFG in question. (S is the start symbol, as usual.)

1. $S \Rightarrow aAA$
   $A \Rightarrow aS \mid bS \mid a$

2. $S \Rightarrow aA \mid aBB$
   $A \Rightarrow Ba \mid Sb$
   $B \Rightarrow bAS \mid \epsilon$

Solutions

Solution to Exercise 6. We need to find configurations and acceptance for the strings aa, $aabba$, and $aaabbb$. Also, we will determine if M is deterministic.

1. **String:** aa
   - $(q_0, aa, \bot) \to (q_0, a, \bot) \to (q_0, a\bot) \to (q_0, \epsilon, aa\bot)$.
   - The final configuration is thus $(q_0, \epsilon, aa\bot)$, and we do not accept since q_0 is not accept, and since the stack is not empty.
2. **String:** $aabba$
   - $(q_0, aabba, \bot) \to (q_0, abba, a\bot) \to (q_0, bba, aa\bot) \to (q_1, ba, a\bot) \to (q_1, a, \bot) \to (q_2, a, \bot)$.
   - The final configuration is $(q_2, a, \bot)$. But we do not accept the string since the input string is not exhausted. Formally, recall that in order to accept a string we require a final configuration of the form $(\_, \varepsilon, \_)$.
3. **String:** $aaabbb$
   - $(q_0, aaabbb, \bot) \to (q_0, aabbb, \bot) \to (q_0, abbb, aa\bot) \to (q_0, bbb, aaa\bot) \to (q_0, bb, aa\bot) \to (q_1, b, a\bot) \to (q_1, \varepsilon, \bot) \to (q_2, \varepsilon, \varepsilon)$.
   - The final configuration is $(q_2, \varepsilon, \bot)$ and we accept by final state or empty stack.
4. **Determinism Check:** Yes, it is deterministic, if $((p, x, y), (q, s)) \in \delta$ and $((p, x, y), (q', s')) \in \delta$ then $(q, s) = (q', s')$. Note that if we are in state q_1 and read a b then we loop if the head of the stack is a, and we are only allowed to proceed to q_2 with an ε-transition if the head of the stack is $\bot$.

Solution to Exercise 7. Our automaton is $M = (\{q_0, q_1, q_2\}, \{a, b\}, \{a, \bot\}, \delta, q_0, \bot, \{q_2\})$, where δ is defined below. It accepts $\{a^i b^j \mid 0 \le i < j\}$ by final state. The role of the states is described by

state	consumed input	stack	
q_0	a^i	$a^i\bot$	where $i \ge 0$
q_1	$a^i b^j$	$a^{i-j}\bot$	where $i \ge j > 0$
q_2	$a^i b^i b^k$	$\bot$	where $i \ge 0$, $k > 0$

$$
\begin{array}{lll}
\delta = \{ & ((q_0, a, \bot), (q_0, a\bot)), & ((q_1, b, a), (q_1, \epsilon)), \\
& ((q_0, b, \bot), (q_2, \bot)), & ((q_1, b, \bot), (q_2, \bot)), \\
& ((q_0, a, a), (q_0, aa)), & ((q_2, b, \bot), (q_2, \bot)), \\
& ((q_0, b, a), (q_1, \epsilon)) & \qquad\qquad \}
\end{array}
$$

Solution to Exercise 8. The idea is to use the stack to simulate a leftmost derivation of the grammar. If the PDA has read w from the input, the stack contains $\gamma\bot$ and the state is q_1 then $S \Rightarrow^* w\gamma$.

1. $M = (\{q_0, q_1, q_2\}, \{a, b\}, \{a, b, S, A, \bot\}, \delta, q_0, \bot, \{q_2\})$, where

$$\begin{aligned}
\delta = \{\ & ((q_0, \epsilon, \bot), (q_1, S\bot)),\\
& ((q_1, \epsilon, S), (q_1, aAA),\\
& ((q_1, \epsilon, A), (q_1, aS)),\\
& ((q_1, \epsilon, A), (q_1, bS)),\\
& ((q_1, \epsilon, A), (q_1, a)),\\
& ((q_1, a, a), (q_1, \epsilon)),\\
& ((q_1, b, b), (q_1, \epsilon)),\\
& ((q_1, \epsilon, \bot), (q_2, \epsilon)) \quad \}.
\end{aligned}$$

2. $M = (\{q_0, q_1, q_2\}, \{a, b\}, \{a, b, S, A, B, \bot\}, \delta, q_0, \bot, \{q_2\})$, where

$$\begin{aligned}
\delta = \{\ & ((q_0, \epsilon, \bot), (q_1, S\bot), && ((q_1, a, a), (q_1, \epsilon)),\\
& ((q_1, \epsilon, S), (q_1, aA)), && ((q_1, b, b), (q_1, \epsilon)),\\
& ((q_1, \epsilon, S), (q_1, aBB)), && ((q_1, \epsilon, \bot), (q_2, \epsilon)),\\
& ((q_1, \epsilon, A), (q_1, Ba)),\\
& ((q_1, \epsilon, A), (q_1, Sb)),\\
& ((q_1, \epsilon, B), (q_1, bAS)),\\
& ((q_1, \epsilon, B), (q_1, \epsilon)) \quad \}
\end{aligned}$$

Both automata accept by final state.

3 Advanced and Exam Like Exercises

Exercise 9. Consider the language P consisting of all properly balanced parentheses. That is, strings over (and) where each left parenthesis (has a matching right parenthesis). For example, the strings ((()())()) and ()() are in P but the string)(() is not. Prove that P is context-free by providing a context-free grammar for P. Your grammar should be unambiguous (motivate why).

Exercise 10. Consider the PDA $\langle Q, \Sigma, \Gamma, \delta, s, \bot, F \rangle$ where

- $Q = \{q_0\}$,
- $\Sigma = \{a, b\}$,
- $\Gamma = \{a, \bot\}$,
- $\delta = \{((q_0, a, \bot), (q_0, a\bot)), ((q_0, a, a), (q_0, aa)), ((q_0, b, a), (q_0, \varepsilon))\}$,
- $s = q_0$ is the start state,
- $\bot$ is the initial stack symbol, and
- $F = \{q_0\}$ is the set of final states.

The machine accepts by final state.

1. Which of the following strings over $\{a, b\}$ are accepted by the PDA?
   (a) aa.
   (b) ba.
   (c) $aabb$.
   (d) $abba$.
   (e) $abab$.
2. What is the language recognized by the PDA?

Solutions

Solution to Exercise 9. We prove that the language is context-free by constructing a context-free grammar which generates the language. A context-free grammar G for the language P is then given by:

$$S \to SS \mid (S) \mid \varepsilon$$

Here, we have made the assumption that $\varepsilon \in P$ since it, technically, fullfils the balanced parenthesis property. However, the grammar is *not* unambiguous since ε can be derived via, for example, $S \Rightarrow SS \Rightarrow \varepsilon S \Rightarrow \varepsilon\varepsilon$ or $S \Rightarrow \varepsilon$. Since the exercise specified that the grammar should be unambiguous we need to fix this, for example via the following grammar.

$$S \to (S)S \mid \varepsilon$$

This grammar is unambiguous. The rule $S \to (S)S$ ensures that every left parenthesis has a corresponding right parenthesis, and the derivation of the empty string is unique since the empty string can only be derived directly from S without further production.

Solution to Exercise 10.

1. aa, $aabb$, and $abab$.
2. All strings over $\{a, b\}$ where each occurence of a b has a matching a somewhere to the left in the string.