HW5 Solutions
Michał Dereziński
February 24, 2015

1 Exercise 4.3

(a) $S \rightarrow ASA | a \quad A \rightarrow a | b$

(b) $S \rightarrow ASA | aa | bb \quad A \rightarrow a | b$

(c) $S \rightarrow aCa | bDb \quad C \rightarrow ACA | a \quad D \rightarrow ADA | b \quad A \rightarrow a | b$

2 Exercise 4.10

(e) $S \rightarrow aSB | \Lambda \quad B \rightarrow b | bb | \Lambda$

(f) $S \rightarrow aSB | a \quad B \rightarrow b | bb | \Lambda$

3 Exercise 4.15

We use strong induction over the number $n$ of rules in a derivation for a given word.

**Inductive Hypothesis** $H(n)$: *Any word $w$, that has a derivation with $n$ rules in our CFG, has more a’s than b’s.*

**Base case:** $n = 1$. Only one rule allowed: $S \rightarrow a$. So $w = a$ and $H(1)$ holds.

**Inductive step:** Suppose that $H(k)$ holds for all $1 \leq k < n$. I will prove $H(n)$. Take a word $w$ with a derivation

$$S \Rightarrow w_1 \Rightarrow ... \Rightarrow w_n = w.$$
 Naturally, $w_1$ has to be one of $aS$, $bSS$, $SSb$, or $SbS$, where any non-terminal $S$ represents a subword of $w$ with a derivation of length at most $n-1$, so we can apply the inductive hypothesis to it. For example, if $w_1 = aS$, then there is a word $v$ such that $n_a(v) > n_b(v)$, where $w = av$, which means that

$$n_a(w) = n_a(v) + 1 > n_b(v) = n_b(w).$$

Similarly, if $w_1 = bSS$, then we have $w = bv_1v_2$ for some $v_1, v_2$, where $n_a(v_1) > n_b(v_1)$ and $n_a(v_2) > n_b(v_2)$, so

$$n_a(w) = n_a(v_1) + n_a(v_2) \geq (n_b(v_1)+1)+(n_b(v_2)+1) > n_b(v_1)+n_b(v_2)+1 = n_b(w).$$

The other two cases follow similarly. Thus, we conclude that $n_a(w) > n_b(w)$, and so $H(n)$ holds, concluding the induction.

4 Exercise 4.21

Definition 3.1 is a recursive definition of the class of regular languages over alphabet $\Sigma$, where every such language can be constructed from the basic building blocks, that are:

1. the empty language $\emptyset = \{\}$,
2. and, for every $a \in \Sigma$, a language with a single one-letter word: $\{a\}$.

We allow three operations for the constructions: $L_1 \cup L_2$, $L_1L_2$, and $L_1^*$. We perform structural induction to prove that for regular $L$, the following statement $P(L)$ is true: There is a CFG that generates $L$.

**Base cases.** Notice, that each of the basic languages can be generated by a CFG; for empty language we use a grammar with no productions, and we can get the language $\{a\}$ from a grammar with one production: $S \rightarrow a$. This concludes the base cases.

**Inductive Step.** We have to show that all three construction operations preserve property $P$, when we apply them to regular languages. Theorem 4.9 says that, in fact, $P$ is closed under those operations (a stronger claim), so we obtain that if $P(L_1)$ and $P(L_2)$, then also $P(L_1 \cup L_2)$, $P(L_1L_2)$ and $P(L_1^*)$. This concludes the structural induction.
5 Exercise 4.22

We construct an NFA $M$, where the states are variables (non-terminals) from the grammar plus one additional state that is also the only accepting state: $Z$. We make variable $S$ the start state. We create transitions as follows:

1. For every $A \rightarrow aB$, we add a transition $A \xrightarrow{a} B$.
2. For every $A \rightarrow a$, we add a transition $A \xrightarrow{a} Z$.
3. For every $A \rightarrow \Lambda$, we add a transition $A \xrightarrow{\Lambda} Z$.

$M$ accepts the same language as the grammar, because every path from start state to accept state corresponds to a derivation in the grammar (and vice versa).

6 Exercise 4.29 (b)

The grammar corresponds to the regular expression $(a + b)^*ab(ab + b)^+$. To obtain a regular grammar, it is useful to construct an NFA for the language.

A DFA for this is

and therefore, the regular grammar is

\[
\begin{align*}
S & \rightarrow aS \mid bS \mid aA \\
A & \rightarrow bB \\
B & \rightarrow aC \mid bD \\
C & \rightarrow bD \\
D & \rightarrow aC \mid bD \mid \Lambda.
\end{align*}
\]
7 Exercise 4.34

To show that the grammar is ambiguous, I must simply exhibit a string with two different parse trees.

Consider the string $x = ababa$. The two parse trees are:

```
S
  |   |
  b   S
  |
  a   a
```

8 Exercise 4.37

Consider the string $x = aababb$. The two parse trees are:

```
S
  |   |
  a   B
  |
  B   B
  |
  a   |
```

9 Exercise 4.54(c)

For full credit, you should show your work, including at least the following three stages.
Grammar after removing Λ-productions:

\[ S \rightarrow AaA \mid CA \mid BaB \mid A \mid aA \mid Aa \mid a \mid C \]
\[ A \rightarrow aaBa \mid CDA \mid aa \mid DC \mid CA \mid DA \mid D \mid A \mid CD \]
\[ B \rightarrow bB \mid bAB \mid bb \mid aS \mid a \]
\[ C \rightarrow Ca \mid bC \mid D \mid b \mid a \]
\[ D \rightarrow bD \mid b. \]

Grammar after removing unit productions:

\[ S \rightarrow AaA \mid CA \mid BaB \mid aA \mid Aa \mid aAaA \mid CDA \]
\[ \mid aa \mid DC \mid DA \mid CD \mid Ca \mid bC \mid b \mid bD \]
\[ A \rightarrow aaBa \mid CDA \mid aa \mid DC \mid CA \mid DA \mid CD \mid Ca \mid bC \mid b \mid a \mid bD \]
\[ B \rightarrow bB \mid bAB \mid bb \mid aS \mid a \]
\[ C \rightarrow Ca \mid bC \mid b \mid a \mid bD \]
\[ D \rightarrow bD \mid b. \]

Final grammar after Chomskyization:

\[ S \rightarrow EA \mid CA \mid FB \mid XA \mid AX \mid a \mid HX \mid IA \]
\[ \mid XX \mid DC \mid DA \mid CD \mid CX \mid YC \mid b \mid YD \]
\[ A \rightarrow HX \mid IA \mid XX \mid DC \mid CA \mid DA \mid CD \mid CX \mid YC \mid b \mid a \mid YD \]
\[ B \rightarrow YB \mid JB \mid YY \mid XS \mid a \]
\[ C \rightarrow CX \mid YC \mid b \mid a \mid YD \]
\[ D \rightarrow YD \mid b \]

\[ E \rightarrow AX \quad F \rightarrow BX \quad G \rightarrow XX \quad H \rightarrow GB \]
\[ I \rightarrow CD \quad J \rightarrow YA \quad X \rightarrow a \quad Y \rightarrow b. \]

10 Exercise 4.16

Let us call the language generated by the grammar as \( L_G \). The proof consists of two parts. First, we have to show that any word \( w \in L \) has a derivation in the grammar (so, \( w \in L_G \)). Second, we will prove that any word \( w \in L_G \) (i.e. with a derivation) satisfies the property \( n_a(w) = n_b(w) \) (i.e. \( w \in L \)).
1. For \( n \geq 0 \), we prove inductive hypothesis \( H(n) \): Every word \( w \in L \) of length \( n \) has a derivation in the grammar.

**Base case.** Let \( n = 0 \). So, \( w = \Lambda \) and the derivation is \( S \Rightarrow \Lambda \).

**Inductive Step.** Suppose that \( H(k) \) holds for any \( 0 \leq k < n \) (strong induction, \( n \geq 1 \)). Take \( w \in L \), such that \( |w| = n \). Let \( x \) denote the first letter in \( w \). We have two cases:

(a) If \( x = a \), then clearly, if there is a derivation for \( w \), it has to start with the production \( S \rightarrow aSbS \). For this derivation to exist, the word has to be of the form \( w = av_i bv_i' \), where \( v_i, v_i' \in L_G \) and \( i \) represents the position of the separating letter \( b \) in \( w \) (so \( v_i = w[2..i-1] \) and \( v_i' = w[i+1..n] \)). Notice, that \( n_b(w) = n_a(w) \geq 1 \), so \( w \) must contain a letter \( b \). Now, we will show that there exists such letter \( b \) at some position \( i \) in \( w \), that the subword \( v_i \) (possibly empty) satisfies \( n_a(v_i) = n_b(v_i) \). For the sake of contradiction, suppose this is not true. It implies that the last letter in \( w \) also has to be \( a \), because otherwise using \( v_n \) would lead to contradiction. Therefore, \( v_n \) has to have more \( b \)'s than \( a \)'s to balance out \( w \). Denote

\[ d(i) = n_b(v_i) - n_a(v_i). \]

Like we said, \( d(n) > 0 \), and moreover \( d(2) = 0 \). Since, the value \( d(i) \) only changes by 1 between two consecutive input numbers, there has to be \( 2 \leq i \leq n \) such that \( d(i) = 0 \) and \( d(i+1) = 1 \). This means that there has to be a letter \( b \) at position \( i \) (since the function increases from \( i \) to \( i+1 \)), and so we get a contradiction.

We showed that there exists a partitioning \( w = av_i bv_i' \), such that \( v_i, v_i' \in L \). By inductive hypothesis, since \( |v_i|, |v_i'| < n \), we find that \( v_i, v_i' \in L_G \), so both derivations can be completed in our grammar, showing that \( w \in L_G \).

(b) If \( x = b \), then we repeat the above argument (with the roles of \( a \) and \( b \) reversed), for the production

\[ S \rightarrow bSaB, \]

obtaining, again, that \( w \in L_G \), which ends the induction.

2. We will prove the second part also by induction. For \( n \geq 0 \), we prove inductive hypothesis \( H(n) \): Every word \( w \) having a derivation with \( n \) productions is in \( L \).
**Base case.** Let $n = 1$. So, $w = \Lambda$, and obviously $w \in L$.

**Inductive Step.** Suppose that $H(k)$ holds for any $0 \leq k < n$ (strong induction, $n \geq 2$). Take a word $w$ with a derivation

$$S \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_n = w.$$ Naturally, $w_1$ has to be one of $aSbS$, $bSaS$. In either case, $w$ consists of two subwords $v_i, v'_i$, which, having shorter derivations must be in $L$ (by inductive hypothesis). Since, both productions introduce the same number of $a$’s and $b$’s, the entire word $w$ will remain balanced (concluding the induction):

$$n_a(w) = 1 + n_a(v_i) + n_a(v'_i) = 1 + n_b(v_i) + n_b(v'_i) = n_b(w).$$