CoCo - Assignment 2

Johannes Agerskov, Mads Friis Frand-Madsen, and Sifan Huang

Dated: February 21, 2021

2.6

Give CFGs that generate the following languages:

b)

$L = \{a, b\}^* \setminus \{a^n b^n \mid n \geq 0\} = \{a^n b^n \mid n \geq 0\}^{\complement}$.
Consider the following CFG:

$$\begin{aligned} S &\to X \mid Y \mid Z, \\ Z &\to aZ \mid Za \mid Zb \mid bZ \mid ba, \\ X &\to aX \mid aXb \mid a, \\ Y &\to Yb \mid aYb \mid b \end{aligned} \tag{0.1}$$

Here we see that the tree of all derivation contain one branch for each of the variables X, Y and Z. The X branch generates all strings of the form $a^i b^j$ for $i > j$. The Y branch derives all string of the form $a^i b^j$ for $i < j$. Finally the Z branch generate all string that contain the substring ba, i.e. which is not of the form $a^i b^j$ for any $i, j \in \mathbb{N}$. Thus we see that this context free grammar generates all string expect those of the form $a^n b^n$, $n \geq 0$.

d)

$L = \{x_1\#x_2\#...\#x_k \mid k \geq 1, \text{each } x_i \in \{a, b\}^*, \text{ and for some } i \text{ and } j,\ x_i = x_j^{\mathcal{R}}\}$.
Consider the following CFG:

$$\begin{aligned} S &\to XYZ, \\ Y &\to aYa \mid bYb \mid \#T\# \mid \varepsilon \\ X &\to T\# \mid \varepsilon, \\ Z &\to \#T \mid \varepsilon, \\ T &\to aT \mid bT \mid T\#T \mid \varepsilon. \end{aligned} \tag{0.2}$$

Notice that Y generates a palindromic string of even length (could also be ε) that then might be split in the middle by a $\#T\#$ ($\varepsilon\#T\#\varepsilon$ in the case of empty string). Now T can generate any number of strings in $\{a, b\}^*$ separated by any number of #s. Thus any string generated by this grammar will be $x_1\#x_2\#...\#x_i\#...\#x_j\#...\#x_k$ for some $k \geq 1$ and for some $i, j \geq 1$ where $|x_i| = |x_j|$ and x_ix_j is a palindromic string. Thus since $x_ix_j = (x_ix_j)^{\mathcal{R}} = x_j^{\mathcal{R}}x_i^{\mathcal{R}}$ we have $x_i = x_j^{\mathcal{R}}$. On the contrary is it obvious that we can generate every string in L by this CFG. Thus the CFG generates L.

2.14

Convert the following CFG into Chomsky normal form.

$$\begin{aligned} A &\to BAB \mid B \mid \varepsilon, \\ B &\to 00 \mid \varepsilon. \end{aligned}$$

We start by adding a new start variable and removing all ϵ-rules

$$\begin{aligned} S &\to A \mid \varepsilon, \\ A &\to BAB \mid BB \mid AB \mid BA \mid \underbrace{A}_{redundant} \mid B \mid \cancel{\varepsilon}, \\ B &\to 00 \mid \cancel{\varepsilon}. \end{aligned}$$

We then remove all unit rules

$$\begin{aligned} S &\to BAB \mid BB \mid AB \mid BA \mid 00 \mid \varepsilon, \\ A &\to BAB \mid BB \mid AB \mid BA \mid 00, \\ B &\to 00. \end{aligned}$$

Finally we patch up the grammar to get it to Chomsky normal form

$$\begin{aligned} S &\to T_1B \mid BB \mid AB \mid BA \mid T_2T_2 \mid \varepsilon, \\ A &\to T_1B \mid BB \mid AB \mid BA \mid T_2T_2, \\ B &\to T_2T_2, \\ T_1 &\to BA, \\ T_2 &\to 0 \end{aligned}$$

The CFG is then written in Chomsky normal form.

2.42

Use the pumping lemma for CFLs to show that the following languages are *not* context free.

a)

Let $L = \{0^n1^n0^n1^n \mid n \geq 0\}$,. We show that L is not context free.

Proof. Assume that L is context free and let p be its pumping length as given by the pumping lemma. Consider then the string $w = 0^p1^p0^p1^p \in L$. According to the pumping lemma, we can split $w = uvxyz$ where $|vxy| \leq p$ and $|vy| > 0$. Now if $vxy = 0^m$ for some $0 < m \leq p$, then $uv^2xy^2z \notin L$ since it contains more 0s than 1s. If $vxy = 1^m$ for some $0 < m \leq p$, then $uv^2xy^2z \notin L$ since it contains more 1s than 0s. If $vxy = 0^n1^m$ for some $2 \leq n + m \leq p$ then vxy belong to one of the halfs of w and thus $uv^2xy^2z \notin L$ since it does not contain an equal amounts of 1s and 0s across the middle. If $vxy = 1^m0^n$ for some $2 \leq n + m \leq p$ then $uv^2xy^2z = 0^p1^k0^{k'}1^p \notin L$ since $k > p$ *or* $k' > p$ (v contains at least one 0 *or* y contains at least one 1). Thus a contradiction with the pumping lemma is unavoidable, and we conclude that L is not context free □

d)

Let $L = \{t_1\#t_2\#...\#t_k \mid k \geq 2, \text{each } t_i \in \{a, b\}^*,\ t_i = t_j \text{ for some } i, j\}$. We show that L is not context free.

Proof. Assume that L is context free and let p be its pumping length as given by the pumping lemma. Consider then the string $w = a^pb^p\#a^pb^p \in L$. Split the string $w = uvxyz$ according to the pumping lemma. Now either $vxy = b^j\#a^i$ for some $i + j \leq p - 1$ *or* $vxy = a^ib^j$ for some $i + j \leq p$. In the second case $vy = a^mb^n$ for some $0 < n + m \leq p$ and thus $uv^0xy^0z = a^{p-m}b^{p-n}\#a^pb^p \notin L$ *or* $uv^0xy^0z = a^pb^p\#a^{p-m}b^{p-n} \notin L$. If $vxy = b^j\#a^i$ then if $\# \in v$ or $\# \in y$ we have $\# \notin uv^0xy^0z$ and thus $uv^0xy^0z \notin L$ by the $k \geq 2$ condition in the definition of L. If $\# \in X$ then $v = b^l$ and $y = a^{l'}$ for $0 < l + l' \leq p$ and thus $uv^2xy^2z = a^pb^{p-l}\#a^{p-l'}b^p \notin L$. Thus a contradiction is unavoidable, and we conclude that L is not context free. □

2.54

Let $Y = \{w \mid w = t_1\#t_2\#...\#t_k, \text{for } k \geq 0,\ t_i \in 1^* \text{ and } t_i \neq t_j \text{ for } i \neq j\}$. We prove that Y is not a context free language.

Proof. Assume for contradiction that Y is a CFL with pumping length, as given by the pumping lemma, p. Let $w = 1^p\#1^{p+1}\#...\#1^{3p} \in Y$. By the pumping lemma it is known that we can split $w = uvxyz$ where $|vxy| \leq p$ and $|vy| > 0$. If vxy contains a # symbol (notice that it can contain at most one #), then if $\# \in v$ we have $v = 1^m\#1^n$ for some $n, m \leq p$ and then $v^3 = 1^m\#\underbrace{1^{n+m}}_{t_i}\#\underbrace{1^{n+m}}_{t_{i+1}}\#1^n$, which contains two sections $t_i = t_{i+1}$. Thus, any string containing v^3 is not in Y which contradicts the pumping lemma. The same applies for y. Therefore we must have $\# \in x$. But then $v = 1^m$ and $y = 1^n$ for some $0 < n + m \leq p - 1$ (since x contains at least a #). Thus

$$uv^0xy^0z = 1^p\#...\#1^{p+k-1}\#1^{p+k-m}\#1^{p+k+1-n}\#...\#1^{3p}$$
$$uv^2xy^2z = 1^p\#...\#1^{p+k-1}\#1^{p+k+m}\#1^{p+k+1+n}\#...\#1^{3p}$$

Now we clearly have $p + k - m \geq p$ *or* $p + k + 1 + n \leq 2p$, since $m + n + 1 \leq p$. Therefore, we conclude that either, if $p + k - m \geq p$, we have $uv^0xy^0z \notin Y$ since the substring $\#1^{p+k-m}\#$ occurs twice, *or*, if $p + k + 1 + n \leq 2p$, we have $uv^2xy^2z \notin Y$ since the substring $\#1^{p+k+1+n}$ occurs twice. Thus we see that $\# \in vxy$ leads to a contradiction with the pumping lemma.
On the other hand if $vxy = 1^m$ for some $0 < m \leq p$ then $vy = 1^n$ for some $0 < n \leq m$. Thereby, either $uv^0xy^0z \notin Y$ or $uv^2xy^2z \notin Y$ since at least one of these contain two sections that are equal. Too see this, notice that $uv^0xy^0z = 1^p\#...\#1^{p+k-1}\#1^{p+k-n}\#1^{p+k+1}\#...\#1^{3p}$ and $uv^2xy^2z = 1^p\#...\#1^{p+k-1}\#1^{p+k+n}\#1^{p+k+1}\#...\#1^{3p}$ and since $n \leq p$ we clearly have $p+k+n \leq 3p$ or $p + k - n \geq p$.

Thus a contradiction is unavoidable and we conclude that Y is not context free. □

3.16

Show that the collection of decidable languages is closed under the following operations.

b) Concatenation

Proof. Given two deciable languages, L_1 and L_2, let M_1 and M_2 be Turing Machinces (TMs) that decides them. We construct a TM, M, that decides the concatenation, L_1L_2, of the two languages. For any string, w, let $w|_k$ denote the first k entries in w and $w|^k$ denote rest of w, such that $w = w|_kw|^k$ for any $0 \leq k \leq |w|$. Here we define $w|_0 = w|^{|w|} = \varepsilon$. We define M by the following algorithm: Given an input string w

1. Set $k = 0$.
2. Set k' such that $k' = k$.
3. Run M_1 on w_k.
4. If M_1 accepts, run M_2 on $w|^k$. If M_1 rejects $w|_k$ and $k < |w|$, set k such that $k = k' + 1$ and return to 2. If M_1 rejects $w|_k$ and $k = |w|$, M *rejects.*
5. If M_2 accepts $w|^k$, M *accepts.* If M_2 rejects $w|^k$ and $k < |w|$, set k such that $k = k' + 1$ and return to 2. If M_2 rejects $w|^k$ and $k = |w|$, M *rejects.*

Clearly, M is a decider, and it accepts a string w if and only if $w = xy$ for $x \in L_1$ and $y \in L_2$.

Given decidable language L_1 and L_2, let M_1 and M_2 be TMs that decides them. We construct an NTM, M that decides L_1L_2. We use that a language is decidable if and only if some

NTM decides it (Corollary 3.19). We define a NTM, M, to non-deterministically on the tape, split the input string in all possible ways. This can be done by letting the machine either put a # between its current head location and the one to right *or* leave it be. If it put down a #, it continues with the rest of the computation, and if it did not, it simply stays in the same state and moves right. Repeating all the way down the string until it meets a $\sqcup$, at which case it just rejects. the Turing machine then runs M_1 on the first component *i.e.* everything before # and M_2 on the second component *i.e.* everything after #. If M_1 moves its head right to a # we simply move the entire tape to the right of an including that # one slot right and produce a blank slot to the left of that # and if M_2 moves left onto a # it simply return to the slot to the right of the # simulating the beginning of the tape. M then *accepts* if for *one of all* the non-deterministic branches both M_1 *and* M_2 accept. Clearly M is a decider since each non-deterministic branch will be decided. □

c) Kleene star

Proof. Given a decidable language L, let M_1 be a TM that decides L. We construct a TM, M that decides L^*. We define first an auxiliary TM $M' = M$. We then define M (and M') by the following algorithm: Given an input string

1. If $w = \varepsilon$, M *accepts.*
2. Set $k = 1$.
3. Set k' such that $k' = k$.
4. Run M_1 on $w|_k$.
5. If M_1 accepts $w|_k$, M marks the part of the string $w|_k$ and run M with the unmarked part of w, $w|^k$, as input string, if this M rejects $w|^k$ and $k < |w|$, set $k = k' + 1$ and return to 3. If M rejects $w|^k$ and $k = |w|$, M *rejects.* If M_1 rejects $w|_k$ and $k < |w|$, set k such that $k = k' + 1$ and return to 3. If M_1 rejects $w|_k$ and $k = |w|$, M *rejects.*

Notice that this definition of M refers to itself, however, we are guaranteed that this recursion is finite by the observation that every recursive step shortens the remaining string by at least one, and we we end up with the empty string M accepts if it has not rejected before that point. Thus it is clear that M is a decider.

Given a decidable language L, let M_1 be a TM that decides L. We construct a TM, M that decides L^*. We use that a language is decidable if and only if some NTM decides it (Corollary 3.19). We define a NTM, M, to non-deterministically on the tape, split the input string in all possible ordered splittings *i.e.* all possible ways of splitting the string w in $w_1\#w_2\#...\#w_n$ such that $w_1w_2...w_n = w$, where $n \in \mathbb{N}$. This can be done by letting the machine either put down a # to right of its current head location and then move two right such it is located to the left of #, *or* simply move one right. It then repeats this non-deterministic process. It stops a non-deterministic branch and continues the calculation, as described below, when it meets a $\sqcup$. After having stopped a branch. It then runs M_1 on each split component,*i.e.* substring w_i that is squeezed between two #-symbols. If M_1 moves its head right to a # we simply move the tape to the right of an including that # one slot right and produce a blank slot on the left of that #. M then accepts if *one of all* of the M_1s accept for *some* ordered splitting of w. Clearly M is a decider since each non-deterministic branch will be decided. $\square$

c) Complementation

Proof. Let L be a decidable language and M_1 a TM that decides it. We show that the complement of L also is a decidable language. For an input string simply define the TM, M that runs M_1 on w. M *rejects* if M_1 accepts, and M *accepts* if M_1 rejects. Clearly M accepts an input string, w, if and only if $w \notin L$. Furthermore, M is a decider since M_1 is. $\square$

d) Intersection

Proof. Let L_1 and L_2 be decidable languages, and M_1 and M_2 be TMs that decides them. We then define the TM, M by: Given an input string w, M runs M_1 on w, if M_1 rejects, M *rejects*. If M_1 accepts, M runs M_2 on w. If M_2 rejects, M *rejects*, if M_2 accepts, M *accepts*. Clearly M accepts a string if and only if $w \in L_1$ and $w \in L_2$, *i.e.* $w \in L_1 \cap L_2$. Furthermore, M is a decider since M_1 and M_2 is. $\square$

3.18

Consider the doubly infinite tape Turing machine. We show that this machine is equivalent to an ordinary TM. One direction is clear. Given the doubly infinite tape machine, we can simulate the ordinary TM by simply introducing a special symbol, $\aleph$ to the left of the input string, and include the transition function $\delta(q, \aleph) = (q, \aleph, R)$ for any state, q. Such that moving to this tape location always sends you back to the left most input address.

The other direction is more interesting. To Simulate the doubly infinite tape machine DM, by an ordinary TM, M, we include on the ordinary machine, a special character, $\aleph$, and put it in the left most position of the tape. Now whenever the head of M is above $\aleph$, M moves the entire tape except $\aleph$ one slot to the right leaving a blank slot after $\aleph$ which it then moves its head to. Thus the machine can now move infinitely to the left, by simply pushing the tape to the right. Clearly any doubly infinite tape machine can be simulated in this way. If DM accepts an input string, M with the same transition functions and states including this extra tape letter and transition functions for this letter, will perfom the same computations and thus accept the string. On the contrary, if DM does not accept w then M will not accept, since if can only enter the accept state if DM does so.