

CoCo - Exam 2021

Johannes Agerskov

Dated: April 15, 2021

Some of the problem descriptions below have been copied from the exam sheet.

Question 1

Consider the following languages over the alphabet $\Sigma = \{a, b, c\}$:

$$L_1 = \{w \in \Sigma^* : w \text{ contains an odd number of occurrences of the letter } a\}$$

$$L_2 = \{a^m b^n c^{m+n} : m, n \in \mathbb{N}\}$$

$$L_3 = \{a^m c^{m+n} b^n : m, n \in \mathbb{N}\}$$

$$L_4 = \{a^m c^{m^2 n^2} b^n : m, n \in \mathbb{N}\}$$

Part 1.1

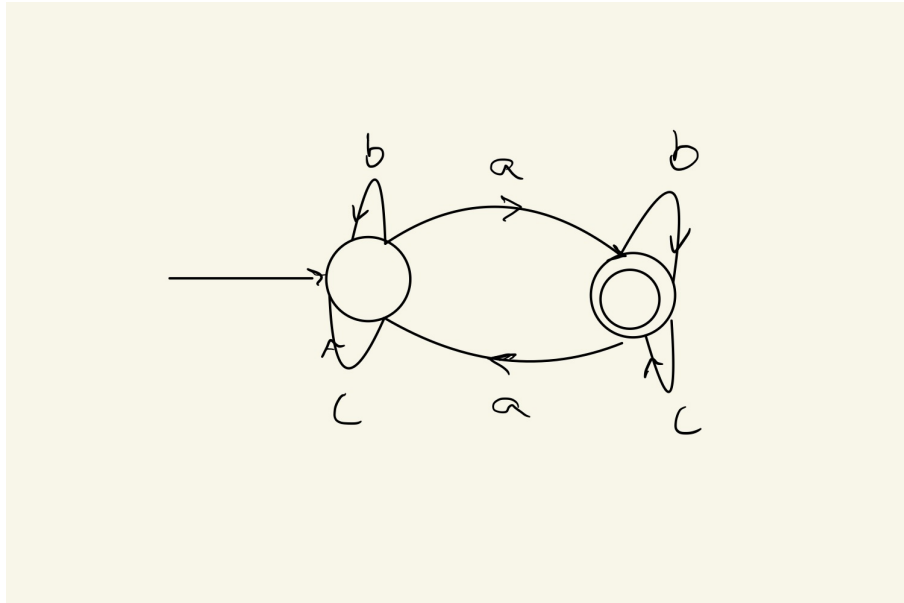
We show that L_1 is regular, and it thus follows that it also is context-free by Corollary 2.32 in M. Sipser.

Proof. The DFA in Figure 1 recognizes L_1 , it then follows from definition 1.16 that L_1 is regular. □

Part 1.2

We show that L_2 is not regular, but it is context-free.

Proof. That is L_2 is not regular can be seen by the following proof by contradiction: Assume L_2 is regular, and let p denote the pumping length as given by the pumping lemma for regular languages Theorem 1.70. Consider now the string $w = a^p b^p c^{2p} \in L_2$, which clearly satisfies $|w| > p$. Thus by the pumping lemma we can split $w = xyz$ with $|xy| \leq p$ and $|y| > 0$. Clearly we then have $xy = a^k$ for some $0 < k \leq p$, and therefore $y = a^m$ for some $0 < m \leq p$. But then clearly $xy^i z = a^{p+(i-1)m} b^p c^{2p}$ which for $i \neq 1$ is not in L_2 , contradicting the pumping lemma. Thus we conclude that L_2 is not regular.

Figure 1: DFA that recognizes L_1

That L_2 is context-free can be seen by the fact that it is generated by the CFG

$$\begin{aligned} S &\rightarrow aAc, \\ A &\rightarrow aAc \mid bBc, \\ B &\rightarrow bBc \mid \epsilon. \end{aligned}$$

Notice that it is not clear whether \mathbb{N} contains 0 or not. Since the convention in M. Sipser is $\mathbb{N} = \{1, 2, 3, \dots\}$ that is what is used here. But even if $\mathbb{N} = \{0, 1, 2, \dots\}$ L_2 is still context-free since then it is generated by CFG

$$\begin{aligned} S &\rightarrow aAc \mid \epsilon, \\ A &\rightarrow aAc \mid B, \\ B &\rightarrow bBc \mid \epsilon. \end{aligned}$$

□

Part 1.3

We show that L_3 is not regular, but it is context-free.

Proof. The proof essentially goes as that for L_2 .

That is L_3 is not regular can be seen by the following proof by contradiction: Assume L_3 is regular, and let p denote the pumping length as given by the pumping lemma for regular languages Theorem 1.70. Consider now the string $w = a^p b^{2p} c^p \in L_3$, which clearly satisfies $|w| > p$. Thus by the pumping lemma we can split $w = xyz$ with $|xy| \leq p$ and $|y| > 0$. Clearly we then have $xy = a^k$ for some $0 < k \leq p$, and therefore $y = a^m$ for some $0 < m \leq p$. But then clearly $xy^i z = a^{p+(i-1)m} b^{2p} c^p$ which for $i \neq 1$ is not in L_3 , contradicting the pumping lemma.

Thus we conclude that L_3 is not regular.

That L_3 is context-free can be seen by the fact that it is generated by the CFG

$$S \rightarrow aAc^2Cc,$$

$$A \rightarrow aAc \mid \epsilon,$$

$$C \rightarrow cCb \mid \epsilon.$$

if $\mathbb{N} = \{1, 2, 3, \dots\}$ and by CFG

$$S \rightarrow AC,$$

$$A \rightarrow aAc \mid \epsilon,$$

$$C \rightarrow cCb \mid \epsilon.$$

if $\mathbb{N} = \{0, 1, 2, 3, \dots\}$

□

Part 1.4

We show that L_4 is not context-free, and it then clearly follows from Corollary 2.32 that L_4 is not regular.

Proof. Assume that L_4 is context-free, and let p be the pumping length as given by the pumping lemma for context-free languages Theorem 2.34. Consider then the string $w = a^p c^{p^2+1} b$. By the pumping lemma we may split $w = uvxyz$ where $|vy| > 0$ and $|vxy| \leq p$. Thus either $vxy = a^k$ for some $0 < k \leq p$ in which case $vy = a^m$ for some $0 < m \leq p$, but then $uv^i xy^i z = a^{p+(i-1)m} c^{p^2+1} b$ which is not in L_4 for $i \neq 1$, contradicting the pumping lemma. Or $vxy = a^k c^l$ for some $0 < k + l \leq p$, but then $vy = a^s c^t$ for some $0 < k + l \leq p$ and $uv^i xy^i z = a^{p+(i-1)s} c^{p^2+1+(i-1)t} b$, but clearly there exist for any s, t such that $s > 0$ an $i \in \{0, 1, 2, \dots\}$ such that $(p + (i-1)s)^2 > p^2 + 1 + (i-1)t$, and thus for such an i , $uv^i xy^i z$ is not in L_4 contradicting the pumping lemma and if $s = 0$ we obviously have $uv^i xy^i z = a^{p+(i-1)s} c^{p^2+1+(i-1)t} b \notin L_4$ also contradicting the pumping lemma. Or we may have $vxy = c^k$ for some $0 < k \leq p$, in which case $vy = c^m$ for some $0 < m \leq p$, but then $uv^i xy^i z = a^p c^{p^2+1+(i-1)m} b$ which is not in L_4 for $i \neq 1$ contradicting the pumping lemma. Finally we may have $vxy = c^k b$ in which case the contradicting is obtained by the same method as for $vxy = a^k c^l$. Thus a contradiction is unavoidable, and we conclude that L_4 is not context-free. □

Part 1.5

We show now that L_4 belongs to L.

Proof. We assume that $\mathbb{N} = \{1, 2, 3, \dots\}$, but the proof works for $\mathbb{N} = \{0, 1, 2, \dots\}$ with small modifications. Consider the log-space TM, $M =$ "On input w

1. Scan the input, and compare neighboring letters (by storing and overwriting one letter on the worktape all the way). If ever substring ab , ca , ba or bc is found, *reject*.

2. Count the number of as , bs and cs with three counters, i, j, k respectively, on the worktape.
3. Check that $i^2 + j^2 = k$. If yes *accept*, if no, *reject*."

Evidently this M decides L_4 . Furthermore, it is log-space, as the first step requires only one slot on the worktape, step 2 requires only tree counters (in binary) which take up only logarithmic space. And finally we may multiply $i \cdot i$ by adding i , i times, which can be done by having an extra counter keeping track of how many times we have added i . Of course we also need the obvious separator symbols, which can clearly be included in log space. Thus we see that M run in logarithmic space, and we conclude that L_4 is in L. \square

Question 2

For any string w we let $rev(w)$ denote the reverse of w .

Part 2.1

We consider in the following the language

$$REV_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM such that at some point during the execution of } M \text{ on } w \\ \text{the tape of } M \text{ consists of } rev(w) \text{ followed by only blank symbols} \}$$

Part 2.2

We show that REV_{TM} is Turing recognizable.

Proof. The following TM, recognizes REV_{TM} , $M' =$ "On input $\langle M, w \rangle$

1. Simulate M on w
2. While M is running:
 3. keep a counter, i , of how many computation steps of M have been simulated.
 4. In each computation step of M , compare the first $\min(i, |w|)$ entries of the tape to $rev(w)$, if at some point the tape content matches $rev(w)$ *accept*. If M halts without $rev(w)$ appearing on the tape, *reject*.

Clearly M' accept if and only if there is some point where the tape content of M running on w matches $rev(w)$. Notice that the counter i ensures that the checking in each step can be done in finite time, and also that the tape content of M clearly cannot be anything but blank beyond i , since M have not have time to alter these tape cells yet. \square

Part 2.3

We show that REV_{TM} is not decidable.

Proof. This follows by simply noting that we can reduce A_{TM} to REV_{TM} by the following reduction: For any $\langle M, w \rangle$ construct $\langle \tilde{M}, w \rangle$, where \tilde{M} is equivalent to M but it starts by writing a special character, which is not in the input alphabet, on the tape, say \aleph , which does not interfere with the computation, and it erases the tape (including \aleph) and writes $\text{rev}(w)$ on the tape before accepting. Then clearly $\langle \tilde{M}, w \rangle \in \text{REV}_{\text{TM}}$ if and only if $\langle M, w \rangle \in A_{\text{TM}}$. Furthermore, this reduction is clearly done by a computable function. Thus we conclude that $A_{\text{TM}} \leq \text{REV}_{\text{TM}}$ and it follows by Theorem 4.11 and Corollary 5.23 that REV_{TM} is undecidable. \square

Part 2.4

Let $f : \Sigma^* \rightarrow \Sigma^*$ be a computable function. We show that there is a TM M with the property that $f(\langle M \rangle)$ is a description of a TM M' such that for each $w \in \Sigma^*$,

1. M halts on w if and only if M' halts on w , and
2. if M' halts on w with a string s_w on its tape then M halts on w with the string $s_w s_w$ on its tape.

Proof. Consider the following TM, $M =$ "On input w

1. Obtain own description via recursion theorem (Thm 6.3).
2. Compute $f(M)$, and let G be the TM such that $\langle G \rangle = f(M)$.
3. Simulate G on w . If G halts, erase everything on the tape except the tape content of G and duplicate the tape content. *Accept* if G accepted and *reject* if G rejected."

Clearly, if $f(M)$ is a description of the TM M' we see that M halts if and only if M' halts, since M simulates M' in its description. Also we see that by design, M will have exactly $s_w s_w$ on the tape when halting on w , if M' have s_w on the tape when halting on w . Thus M is exactly a description of the desired TM. \square

Question 3

For a directed graph $G = (V, E)$ and a subset V' of V , the induced subgraph $G[V']$ is the subgraph of G with vertex set V' and edge set consisting of those edges of E with both endpoints in V' .

When we refer to a cycle in the following, we refer to a directed graph consisting of distinct vertices v_1, v_2, \dots, v_m and edges (v_i, v_{i+1}) for $i = 1, 2, \dots, m-1$, and an edge (v_m, v_1) ; m is the size of the cycle. Let $s \geq 2$ be a given integer and consider the following decision problem:

Let $s \geq 2$ be a given integer and consider the following decision problem:

$$s - CYCLE = \{ \langle G \rangle \mid G = (V, E) \text{ is a directed graph with a cycle of size } s \text{ as subgraph} \}$$

Part 3.1

We show that $s - CYCLE$ belongs to P.

Proof. Consider the following TM M "On input $\langle G \rangle$

1. For each collection of s vertices in G , v_1, \dots, v_s :
2. Check that there are edges in G : (v_i, v_{i+1}) for $i = 1, \dots, m-1$ and an edge (v_s, v_1) . If yes, *accept*.
3. *reject*."

Clearly, the loop in step one have at most n^s iterations, where n is the number of vertices in G , as there are less than n^s ways of choosing s vertices out of n . And it is clear that step 2 is polynomial in time as there are at most n^2 edges in G . Thus, M runs in polynomial time. It is also clear by design, that M accepts $\langle G \rangle$ if and only if, G has a cycle of size s as a subgraph. Therefore $s - CYCLE$ is in P. □

Part 3.2

For $s \geq 2$ consider now

$$\begin{aligned} s - CYCLE - HITTING - SET = \{ \langle G, k \rangle \mid G = (V, E) \text{ is a directed graph, } k \in \mathbb{N}, \\ \text{and there is a subset } V' \subseteq V \text{ of size } k \text{ such that } G[V \setminus V'] \\ \text{contains no cycle of size } s \text{ as subgraph} \} \end{aligned}$$

We show that $s - CYCLE - HITTING - SET$ is in NP.

Proof. We construct a polynomial time verifier, V , that verifies $s - CYCLE - HITTING - SET$. V takes as a certificate a subset V' of vertices of G . V "On input $(\langle G, k \rangle, c)$

1. Check that $|c| = k$, if not *reject*.
2. Construct $G' = G[V \setminus V']$.
3. Run M , from Part 3.1, on $\langle G' \rangle$, if M accepts, *reject*, if M rejects, *accept*.

Clearly V runs in polynomial time, as each step if polynomial in time (step 3 because of the result in 3.1). Furthermore, if $\langle G, k \rangle$ is in $s - CYCLE - HITTING - SET$ there exist a c such that V accepts $(\langle G, k \rangle, c)$, and if $\langle G, k \rangle$ is not in $s - CYCLE - HITTING - SET$ such a c can clearly not exist. Thus we conclude that V is a polynomial time verifier for $s - CYCLE - HITTING - SET$ and therefore it belongs to NP. □

Part 3.3

We show that $s - CYCLE - HITTING - SET$ is NP-complete for every $s \geq 2$.

Proof. By the result of Part 3.2 we need only show that $s - CYCLE - HITTING - SET$ is NP-hard. This is done by polynomial time reducing from $VERTEX-COVER$ which is NP-complete by Theorem 7.44, it then follows from Theorem 7.36 that $s - CYCLE - HITTING - SET$ is NP-hard (actually the theorem states that then $s - CYCLE - HITTING - SET$ is NP-complete, since it is NP).

Consider the following reduction: Given $\langle G, k \rangle$ where G is an undirected graph, construct $\langle G', k \rangle$ where G' is the directed graph with all edges in G , (u, v) replaced by a (directed) cycle of size s $(v_1, v_2), \dots, (v_m, u), (u, v_m + 1), \dots, (v_{s-2}, v)$. Clearly if $\langle G, k \rangle$ is in $VERTEX-COVER$, then there is a k -node vertex cover of G and that exact vertex-cover is a subset of vertices V' of G' such that $G'[V \setminus V']$ does not contain a cycle of size s , since such a cycle had to come from an edge of G but all edges touched at least one vertex in V' . Thus, if $\langle G, k \rangle \in VERTEX-COVER$ we have $\langle G', k \rangle$ is in $s - CYCLE - HITTING - SET$. On the other hand if $\langle G, k \rangle$ is not in $VERTEX-COVER$, then for any subset of vertices, V' of G of size k , there is an edge (u, v) such that $u, v \notin V'$.

On the contrary if $\langle G', k \rangle$ is in $s - CYCLE - HITTING - SET$, then there is a subset of vertices V' of G' of size k such that $G'[V \setminus V']$ contains no cycle of size s as a subgraph. But then V' must contain a vertex in each of the constructed cycles between the vertices of G or equivalently for each vertex v' in V' , v' sits in a cycle of length s between two vertices of G (by construction) and there is a vertex v in G such that v is the first vertex of G , which is reached by following that cycle. Let then \tilde{V} be the set consisting of vertices of G , that are reached first from the vertices of V' by following their corresponding cycles. Clearly because $v \in \tilde{V}$ sits in the same cycles as $v' \in V'$, we see that $G'[V \setminus \tilde{V}]$ has no cycle of size s as a subgraph. But then any edge in G must touch a vertex in \tilde{V} . Since if an edge (u, v) did not touch any $v \in \tilde{V}$, the corresponding cycle coming from that edge in the construction above would be a cycle of size s in $G'[V \setminus \tilde{V}]$.

Therefore we conclude that $\langle G, k \rangle$ is in $VERTEX-COVER$ if and only if $\langle G', k \rangle$ is in $s - CYCLE - HITTING - SET$. We also note that the construction $\langle G, k \rangle \mapsto \langle G', k \rangle$ clearly is computable in polynomial time, as we need only include $s - 2$ extra vertices, remove one edge, and add s extra edges, for each edge in G . Therefore we conclude that

$$VERTEX-COVER \leq_P s - CYCLE - HITTING - SET$$

and it follows from Theorem 7.36 that $s - CYCLE - HITTING - SET$ is NP-complete. \square

Question 4

Recall that a string x is a substring of a string y if there are (possibly empty) strings w, r such that $y = wxr$. For example, every string is a substring of itself, and the string ba is a substring of $abba$, but the string aa is not a substring of $abba$. Consider the following two sets:

$\text{NFA} - \text{NO} - \text{GO} = \{(a, A) : a \text{ is a letter, and } A \text{ is an NFA such that no string accepted by } A \text{ contains } a \text{ as a substring}\}$

$\text{NFA} - \text{NO} - \text{GO} - \text{NFA} = \{(B, A) : B \text{ and } A \text{ are NFAs over the same alphabet such that no string accepted by } A \text{ contains any string accepted by } B \text{ as a substring}\}$

Part 4.1

We show that $\text{NFA} - \text{NO} - \text{GO}$ is NL-complete.

Proof. First we show that $\text{NFA} - \text{NO} - \text{GO}$ is in NL. We do this by showing that $\text{NFA} - \text{NO} - \text{GO}$ is in coNL, and it then follows by Theorem 8.27 (NL=coNL) that $\text{NFA} - \text{NO} - \text{GO}$ is in NL. Notice first that if an NFA A accept a string with the letter a in it, then it accepts a string of length at most $2p$ with the letter a in it, where p is number of states of the NFA. This can be seen by the pigeon hole principle. Assume that A accepts w , a is a substring of w , and $|w| > 2p$, then clearly A must loop at least twice on w . Therefore the substring a is read while A is in one of loops, or before or after the loops. In any case, we may remove all parts of the string w that does not contain a and that makes A loop. After this removal we are left with a string w' that contains a and which makes A loop at most once, thus it has length $|w'| \leq 2p$. Now consider the following logarithmic space NTM $M =$ "On input (a, A)

1. Check that a is a letter, if not *accept*.
2. Check that A is an NFA, if not *accept*.
3. Set $p :=$ the number of states of A .
4. Set $i := 1$ and $j = 0$.
5. While $i \leq 2p$
6. Non-deterministically store a letter, b , on the worktape (overwrite if one is already stored).
7. If $b = a$, set $j := 1$.
8. Simulate A on b and store its current state, if it accepts and $j = 1$, *accept*.
9. If $i = 2p$ and A does not accept, *reject*.
10. $i := i + 1$

Evidently, by the above discussion, M accepts the language

$$\overline{\text{NFA} - \text{NO} - \text{GO}} = \{(a, A) \mid a \text{ is not a letter, or } A \text{ is not an NFA, or } A \text{ accept a string with } a \text{ as a substring.}\}$$

Furthermore, M clearly runs in at most logarithmic space since it stores only counters of size up to $2p$ where p is number of states of A , and simulates an NFA, which requires no space. Thus we conclude that $\overline{\text{NFA} - \text{NO} - \text{GO}} \in \text{NL}$, or equivalently that $\text{NFA} - \text{NO} - \text{GO} \in \text{coNL} = \text{NL}$. \square

Part 4.2