Assigment 3 - CoCo

Johannes Agerskov

Dated: February 28, 2021

4.29

Let $A = \{\langle R, S\rangle \mid R \text{ and } S \text{ are regular expressions and } L(R) \subseteq L(S)\}$. We give a TM that desides A. Consider the following TM, M.
M first controls if R and S are regular expressions, if not *reject*. If they are both regular expression, M constructs DFAs, D_1 and D_2, that recognize $L(R)$ and $L(S)^\complement$ (complement of $L(S)$). This is done by the usual conversion of regular expression to DFA, with the interchange of accept and non-accept states in the case of S. M now constructs a DFA, D, from D_1 and D_2 that recognizes $L(R) \cap L(S)^\complement$ by the same method as one would prove that regular languages are closed under intersection. Then M runs M_1 on D where M_1 is the TM from the proof of theorem 4.4 that desides if $L(D)$ is empty. If M_1 accepts, M *accepts*. If M_1 rejects, M *rejects*. Thus M accepts if $L(R) \cap L(S)^\complement = \emptyset$ which is equivalent to $L(R) \subseteq L(S)$ and reject if $L(R) \cap L(S)^\complement \neq \emptyset$ which is equivalent to $L(R) \not\subseteq L(S)$. Notice that D_1, D_2 and D can be constructed directly from the input. M_1 needs to be encoded as a subroutine on M. However, this requires adding finitely many states and tape letters to the TM.

4.13

Let $C_{\text{CFG}} = \{\langle G, k\rangle \mid G \text{ is a CFG and } L(G) \text{ contains exactly } k \text{ strings where } k \geq 0 \text{ or } k = \infty\}$. We give a TM, M, that desides C_{CFG}. In the following R is the TM from the proof of theorem 4.8 that given input $\langle G\rangle$ where G is some CFG, desides if $L(G) = \emptyset$. S is the TM from the proof of theorem 4.7 that given input $\langle G, w\rangle$ where G is a CFG and w is a string desides if G generates w. We also notice that if a CFG is in Chomsky normal form then every generation of a string of length n requires $2n-1$ steps (derivations). We are then going to utilize the pumping lemma for CFGs. Basically if G contains n variable and let b be the maximum number of symbols on the righthand side of any rule in G, we know that any string if $L(G)$ of length greater than or equal to $p := b^{n+1}$ can be pumped (for more details see the proof of theorem 2.34), and thus if any such string is in the CFG, it has inifinitely many string. Now let us describe M. For simplicity let M have two tapes, one for the computation and one for storing variables below such as i. M acts as follows. On input $\langle G, k\rangle$

1. Convert G to Chomsky normal form.

2. Set $j = 0$ and $i = 0$ on the second tape (in binary numbers).

3. Run R on G, if R accepts and $k = i$, M *accept*. If R reject and $j = p$ and $k = \infty$, M *accept*. If R accepts and $j = p$, M *rejects*.

4. Count number of strings of terminals that can be obtained from the start variable, and add this number to i on the second tape.

5. Add rules from the start variable that directly generate all possible string that can be reached in 2 derivation from the start variable and remove all rules from the start variable with strings of terminals on the righthand side..

6. Add 1 to j ($j = j + 1$) and return to 3.

The machine, M, then accept if either it has counted k string in the language and the language can generate no other string *or* is there *is* at least one string with length greater than or equal to p *and* $k = \infty$, and rejects otherwise.

TOO DIFICULT METHOD

Let $C_{\text{CFG}} = \{\langle G, k\rangle \mid G \text{ is a CFG and } L(G) \text{ contains exactly } k \text{ strings where } k \geq 0 \text{ or } k = \infty\}$. We give an NTM, M, that desides C_{CFG}. In the following R is the TM from the proof of theorem 4.8 that given input $\langle G\rangle$ where G is some CFG, desides if $L(G) = \emptyset$. S is the TM from the proof of theorem 4.7 that given input $\langle G, w\rangle$ where G is a CFG and w is a string desides if G generates w. We also notice that if a CFG is in Chomsky normal form then every generation of a string of length n requires $2n - 1$ steps (derivations). We are then going to utilize the pumping lemma for CFGs. Basically if G contains n variable and let b be the maximum number of symbols on the righthand side of any rule in G, we know that any string if $L(G)$ of length greater than or equal to $p := b^{n+1}$ can be pumped (for more details see the proof of theorem 2.34), and thus if any such string is in the CFG, it has inifinitely many string. Now let us describe M. M acts as follows. On input $\langle G, k\rangle$, M converts G to Chomsky normal form and non-deterministically perfom all derivations of $2p - 1$ steps or less. It then counts the numbers of strings of only terminal letters, say i, generated. Furthermore, it construct G' which is just G but with every rule involing $S \rightarrow$ removed and with the rules $S \rightarrow w_1|w_2|...|w_d$ added, where w_d are all strings that was generated with length p Thus G' can generate all the same strings as G that have lenght greater than or equal to p. M now runs R on G', and if R accepts and $i = k$ M *accepts* and if R rejects and $k = \infty$, M *accepts* and otherwise M *rejects*.

Non-determanism does not work, as it can not count number of string generated across branches.

Let $C_{\text{CFG}} = \{\langle G, k\rangle \mid G \text{ is a CFG and } L(G) \text{ contains exactly } k \text{ strings where } k \geq 0 \text{ or } k = \infty\}$. We give a TM, M, that desides C_{CFG}. In the following R is the TM from the proof of theorem 4.8 that given input $\langle G\rangle$ where G is some CFG, desides if $L(G) = \emptyset$. We also notice that if a

CFG is in Chomsky normal form then every derivation of a string of length n requires $2n-1$ steps (derivations). We are then going to utilize the pumping lemma for CFGs. Basically if G contains m variable and let b be the maximum number of symbols on the righthand side of any rule in G (two in the case of Chomsky normal form), we know that any string if $L(G)$ of length greater than or equal to $p := b^{m+1}$ can be pumped (for more details see the proof of theorem 2.34), and thus if any such string is in the CFG, it has inifinitely many string. Now let us describe M. M acts as follows. On input $\langle G, k\rangle$, M converts G to Chomsky normal form and perfom all derivations of $2p-1$ steps or less simultaneously. This is done by simply starting with the start varibale on the tape, and then use all possible rules, each rule separated on the tape by #s. Continue this process with all new variables generated. It then counts the numbers of strings of only terminal letters generated without a # in it, but with a # on each side, say there are i of them. Furthermore, it construct G' which is just G but with every rule involing, $S \to ...$, removed and with the rules $S \to w_1|w_2|...|w_d$ added, where w_d are all strings that was generated with length p. Thus G' can generate all the same strings as G that have lenght greater than or equal to p. M now runs R on G', and if R accepts and $i = k$ M *accepts* and if R rejects and $k = \infty$, M *accepts*. Otherwise M *rejects*.

5.4

We show that if $A \leq_m B$ and B is a regular language, then A need not be a regular language.

Proof. We proceed by constructing an example of the satement where A is non-regular and B is regular with $A \leq_m B$.
By definition of $A \leq_m B$, there exist a computable function $f : \Sigma^* \to \Sigma^*$ such that $f(A) \subseteq B$ and $f(A^\complement) \subseteq B^\complement$. By definition of computable there exist a TM, M_1, such that for each input w, M_1 halts with only $f(w)$ on its tape. Let B be the regular language 1^* and let A be the non-regular language $\{0^n1^n \mid n \geq 0\}$. We let the computable function be defined by

$$f(w) = \begin{cases} 1^n & \text{if } w = 0^n1^n \text{ for } n \geq 0 \\ 0 & \text{if } w \neq 0^n1^n \text{ for } n \geq 0 \end{cases} \tag{0.1}$$

This is clearly a computable function as it is computed by the TM that scans it input to verify that it is of the form 0^n1^n for some $n \geq 0$, if so, it deletes all 0s and moves the 1s to the beginning of the tape before halting. If the input is not of the form 0^n1^n for some $n \geq 0$, it simply deletes the entire tape and put down a 0 on the leftmost tape position before halting. Notice that $f : \{0,1\}^* \to \{0,1\}^*$ with $f(A) \subseteq B$ and $f(A^\complement) \subset B^\complement$. Thus, we have constructet a *non-regular* A and *regular* B with $A \leq_m B$. □

It might be noted that the reason for mapping reducibility of a language, A, to a regular language, B, does not induce regularity of A, is to be found in the definition of mapping reducibility. We require only the mapping to be computable, which means that A can be

reduced to B only via a (decider) TM. Thus we might during this reduction utilize that TMs are more powerful than DFAs. In other word, computing with a TM and a DFA in series, is not equivalent to having a single DFA.

5.28

We consider the problem of wether a Turing machine ever writes a blank symbol over a nonblank symbol during its course of computation on any input string. Formulated as a laguage we might state the problem as

$L = \{\langle B, w\rangle \,|\, B$ is a TM, w is a string, and there exist for some $l \geq 0$ a sequence of configurations $C_1 C_2, ..., C_l$ of B on input string w such that C_1 is the start configuration, C_j yields C_{j+1} for all $j < l,$ and $C_i = ...aq_ic....$ and $C_{i+1} = ...a \sqcup q_{i+1}...$ or $C_{i+1} = ...q_{i+1}a \sqcup ...$for some $i < l\}$

We now show that L is undecidable. We show this by reducibility. We show that A_{TM} is reducible to L.

Proof. Assume that we have a decider, D, for L. We then construc a desider, M, for A_{TM}. First we notice that given any TM G we can modify it to never write a blank symbol, by just including a special letter in its tape alphabet, $\aleph$, and adding all the transition functions $\delta(q, \aleph) = (q', a, L/R)$ whenever G contains the transition function $\delta(q, \sqcup) = (q', a, L/R)$ (and $\delta(q, \aleph) = (q', \aleph, L/R)$ for any $\delta(q, \sqcup) = (q', \sqcup, L/R)$) and then replace all transition functions $\delta(q, a) = (q', \sqcup, L/R)$ with $\delta(q, a) = (q', \aleph, L/R)$, where q, q' are any pair of states, a is any nonblank tape letter, and L/R is L or R.
It is also clear given G' that we can further modify this to G'' that accepts the same language, but erases the entire tape by overwriting it with blanks before accepting, by just adding a new state with corresponding transition functions that erase the tape, and goes to the accept state when the tape is empty.

We describe M as follows: On input $\langle G, w\rangle$

1. Check that $\langle G, w\rangle$ is a correct encoding of a TM and a string. If not *reject.*
2. Construct G'', as described above, from G.
3. Run S on $\langle G'', w\rangle$. If S accepts, M *accepts.* If S rejects, M *rejects.*

It is then clear that if S accepts $\langle G'', w\rangle$ then G'' writes a blank over a nonblank, which it can do, if it accepts w, which is again only possible if G' and thus G accept w. On the other hand if S rejects, it is clear that G'' and thus G' and G does not accept w. Thus M decides A_{TM}. □

5.18b

We use Rice's theorem to prove that the following language is undecidable:

$$P = \{\langle M \rangle \mid M \text{ is a TM and } 1011 \in L(M)\}$$

Proof. We note that P is a laguage consisting of TM descriptions. We first show that P is nontrivial, i.e. that there exist a TM, S, such that $\langle S \rangle \in P$. Let S be the TM, that accepts all strings in $\{0,1\}^*$, then $1011 \in L(S)$, this TM is easily constructed by letting, every transition function map the start state to the accept state. We then show that for $\langle M \rangle \in P$, P is a property of $L(M)$. To see this, let M_1 and M_2 be TMs with $L(M_2) = L(M_1)$, then clearly $1011 \in L(M_1)$ if and only if $1011 \in L(M_2)$ so $\langle M_1 \rangle \in P$ if and only if $\langle M_2 \rangle \in P$. Thus we have shown that for any $\langle M \rangle \in P$, P is a nontrivial property of the language of M. It then follows from Rice's theorem that P is undecidable. □

Exam 2019

Q 2.1

Let

$$SAMETAPE_{TM} = \{\langle M_1, M_2, w_1, w_2 \rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } \exists t_1, t_2 \in \mathbb{N}_0 \text{ such that the tape content of } M_1 \text{ after } t_1 \text{ computations on input } w_1 \text{ match the tape content of } M_2 \text{ after } t_2 \text{ computations on input } t_2\}$$

we show that $SAMETAPE_{TM}$ is Turring recognizable. Notice first that theorem 3.13 says that every multitape TM has an equivalent single tape TM. Furhtermore, theorem 3.16 say similarly that every non-deterministic TM has an equivalent TM. Now it *does not* follows from these two result alone that every multitape non-deterministic TM has an equivalent TM. However, it *does* follows by combining their proofs. First the proof of theorem 3.16 uses a three tape (input tape, simulation tape and adress tape) TM, that simulates the NTM. Now in order to simular an m-tape NTM we use an $m + 2$-tape TM (input tape, m simulation tapes, and the adress tape). Now clearly by following the steps of the proof of theorem 3.16 we construct a four tape TM that simulates a multitape. Notice we need only one input tape, since this tape is never altered anyway. Thus we conclude that every multitape TM has an equivalent TM. We refer to the book (M. Sipser, *Introduction to the theory of computation, 3rd International edition*) for the remaining details of the two individual proofs.
Now to show that $SAMETAPE_{TM}$ is Turing recognizable, we construct a two tape NTM that recognizes it. On input $\langle M_1, M_2, w_1, w_2 \rangle$ the two-tape NTM, M, puts w_1 and w_2 on its two tapes, it then iteratively non-deterministically simulates a single computation *or* does nothing,

on each tape. In this way the non-deterministic branches after n non-deterministic computations contain two tapes: the tape of M_1 on input w_1 after i computations and the tape of M_2 on input w_2 after j computations on the second where $i, j \leq n$. Inbetween each step M checks on each non-deterministic branch if the tape content of the two tapes mathces, if yes, it *accepts* if not it keeps going. It is clear that M accepts if and only if $\langle M_1, M_2, w_1, w_2 \rangle \in SAMETAPE_{TM}$.

Q 2.2

We no show that $\overline{SAMETAPE_{TM}}$ is not Turing recognizable. This follows if we can show that $SAMETAPE_{TM}$ is not decidable, since we have by theorem 4.22 that a language is decidable if and only if it is Turing-recognizable and co-Turing-recognizable. Now to show that $SAMETAPE_{TM}$ is not decidable we show that A_{TM} is reducible to $SAMETAPE_{TM}$. To see this assume that we have a decider, S, for $SAMETAPE_{TM}$, and let R be a TM that does nothing when its head is on a blank tape entry or on the special letter ℵ expect moving right and left over and over. consider TM, M defined as follows. On input $\langle M_1, w \rangle$

1. Check if $\langle M_1, w \rangle$ is and encoding of a TM and a string, if not reject.

2. Modify M_1 to M_1' such that M_1' have a speciel letter ℵ in its tape alphabet, and M_1' empties its tape and put down an ℵ in the first slot, before accepting.

3. Run S on $\langle M, R, w, ℵ \rangle$, if S accept, M *accepts*. If S rejects M *rejects*.

from this construction it is clear that M accept $\langle M_1, w \rangle$ if and only if $\langle M_1, w \rangle \in A_{TM}$. Thus we conclude that no decider S for $SAMETAPE_{TM}$ exists, and thus $SAMETAPE_{TM}$ is not co-Turing-recognizable as desired.