CSCE-433 Formal Languages & Automata
CSCE-627 Theory of Computability

Spring 2022

Instructor: Dr. Jianer Chen
Office: PETR 428
Phone: 845-4259
Email: `chen@cse.tamu.edu`
Office Hours: MWF 10:30–11:30am

Senior Grader: Avdhi Shah
Office: N/A
Phone: tba
Email: `avdhi.shah@tamu.edu`
Office Hours: tba

Solutions to Assignment #5

1. Give implementation-level descriptions of Turing machines that decide the following languages over the alphabet $\{0, 1\}$.

(a) (CSCE 433 students only) $L_{433} = \{w \mid w$ contains twice as many 0's as 1's$\}$.

(b) (CSCE 627 students only) $L_{627} = \{w \mid w$ does not contain twice as many 0's as 1's$\}$.

Solutions.

(a) We can use the following Turing machine M_{433} to accept the language L_{433}: M repeatedly scans the tape, and in each scan, it erases two 0's and one 1. The transition function δ is given as follows.

(1) $\delta(q_s, 0/1) = (q_-, 0/1, \downarrow)$;
(2) $\delta(q_-, 0) = (q_0, \#, \rightarrow)$;
(3) $\delta(q_-, 1) = (q_1, \#, \rightarrow)$;
(4) $\delta(q_-, \#) = (q_-, \#, \rightarrow)$;
(5) $\delta(q_-, \Box) = (q_{acc}, \Box, \downarrow)$;
(6) $\delta(q_0, 0) = (q_{00}, \#, \rightarrow)$;
(7) $\delta(q_0, 1) = (q_{01}, \#, \rightarrow)$;
(8) $\delta(q_0, \#) = (q_0, \#, \rightarrow)$;
(9) $\delta(q_1, 0) = (q_{01}, \#, \rightarrow)$;
(10) $\delta(q_1, 1) = (q_1, 1, \rightarrow)$;
(11) $\delta(q_1, \#) = (q_1, \#, \rightarrow)$;
(12) $\delta(q_{00}, 1) = (q_b, \#, \leftarrow)$;
(13) $\delta(q_{00}, 0) = (q_{00}, 0, \rightarrow)$;
(14) $\delta(q_{00}, \#) = (q_{00}, \#, \rightarrow)$;
(15) $\delta(q_{01}, 0) = (q_b, \#, \leftarrow)$;
(16) $\delta(q_{01}, 1) = (q_{01}, 1, \rightarrow)$;
(17) $\delta(q_{01}, \#) = (q_{01}, \#, \rightarrow)$;
(18) $\delta(q_b, 0/1) = (q_b, 0/1, \leftarrow)$;
(19) $\delta(q_b, \#) = (q_b, \#, \leftarrow)$;
(20) $\delta(q_b, \Box) = (q_-, \Box, \rightarrow)$.

Thus, the Turing machine is $M_{433} = (Q, \Sigma, \delta, q_s, q_{acc}, q_{rej})$, where $Q = \{q_s, q_-, q_0, q_1, q_{00}, q_{01}, q_b\}$, and $\Sigma = \{0, 1, \#, \Box\}$ ($\Box$ is the blank symbol). Note that we use q_s instead of q_0 as the start state for notational convenience. The state q_- is for the case where the number of erased 0's is twice the number of erased 1's. The state q_0 (respectively, q_1, q_{00}, and q_{01}) is for the case where a new 0 (respectively, a new 1, two new 0's, and a new 0 and a new 1) has been erased. Note that the machine M "erases" a 0 or a 1 by overwriting it with the symbol #. Once two 0's and one 1 are erased, the machine M goes to state q_b (lines (12) and (15)), which moves the head back to the beginning of the input and restarts with the state q_- (lines (18)-(20)). In particular, if in the state q_-, we see the end of the input, i.e., the right $\Box$ after the input, then we have erased all 0's and the 1's, and the number of erased 0's is just twice that of erased 1's, so the machine M accepts (line (5)). Note that for all states q' and symbols a' such that $\delta(q', a')$ is not given in the above list, we implicitly define $\delta(q', a') = (q_{rej}, a', \downarrow)$. In particular, we have

$$\delta(q_0, \Box) = \delta(q_1, \Box) = \delta(q_{00}, \Box) = \delta(q_{01}, \Box) = (q_{rej}, a', \downarrow),$$

which shows the cases when we reach the end $\Box$ of the input with all 0's and 1's erased, but the number of erased 0's is not exactly twich the number of erased 1's – so we should reject.

(b) Please read the discutions and explanations in the solution to (a) above. Note that L_{627} is just the complement of L_{433}. Thus, we can simply swap the accept state q_{acc} and the reject state q_{rej} of the Turing machine M_{433} in (a), which will give a Turing machine M_{627} that accepts L_{627} (note that M_{433} is deterministic). The Turing machine for the language L_{627} is $M_{627} = (Q, \Sigma, \delta, q_s, q_{acc}, q_{rej})$, where $Q = \{q_s, q_-, q_0, q_1, q_{00}, q_{01}, q_b\}$, and $\Sigma = \{0, 1, \#, \Box\}$. The transition function δ is given as follows.

(1) $\delta(q_s, 0/1) = (q_-, 0/1, \downarrow)$;
(2) $\delta(q_-, 0) = (q_0, \#, \rightarrow)$;
(3) $\delta(q_-, 1) = (q_1, \#, \rightarrow)$;
(4) $\delta(q_-, \#) = (q_-, \#, \rightarrow)$;
(5) $\delta(q_0, 0) = (q_{00}, \#, \rightarrow)$;
(6) $\delta(q_0, 1) = (q_{01}, \#, \rightarrow)$;
(7) $\delta(q_0, \#) = (q_0, \#, \rightarrow)$;
(8) $\delta(q_0, \Box) = (q_{acc}, \Box, \downarrow)$;
(9) $\delta(q_1, 0) = (q_{01}, \#, \rightarrow)$;
(10) $\delta(q_1, 1) = (q_1, 1, \rightarrow)$;
(11) $\delta(q_1, \#) = (q_1, \#, \rightarrow)$;
(12) $\delta(q_1, \Box) = (q_{acc}, \Box, \downarrow)$;
(13) $\delta(q_{00}, 0) = (q_{00}, 0, \rightarrow)$;
(14) $\delta(q_{00}, 1) = (q_b, \#, \leftarrow)$;
(15) $\delta(q_{00}, \#) = (q_{00}, \#, \rightarrow)$;
(16) $\delta(q_{00}, \Box) = (q_{acc}, \Box, \downarrow)$;
(17) $\delta(q_{01}, 0) = (q_b, \#, \leftarrow)$;
(18) $\delta(q_{01}, 1) = (q_{01}, 1, \rightarrow)$;
(19) $\delta(q_{01}, \#) = (q_{01}, \#, \rightarrow)$;
(20) $\delta(q_{01}, \Box) = (q_{acc}, \Box, \downarrow)$;
(21) $\delta(q_b, 0/1) = (q_b, 0/1, \leftarrow)$;
(22) $\delta(q_b, \#) = (q_b, \#, \leftarrow)$;
(23) $\delta(q_b, \Box) = (q_-, \Box, \rightarrow)$;
(24) $\delta(q_-, \Box) = (q_{rej}, \Box, \rightarrow)$.

Note that the machine M_{627} rejects only when it has erased all 0's and 1's in the input, but is in the state q_- (line (24)), which is the case that in the input the number of 0's is exactly twice that of 1's. $\Box$

2. Show that the collection of (Turing-)decidable languages is closed under the operations of (a) complementation, and (b) intersection. Use the solution for Problem 3.15(a) in the textbook (page 191) as a guide for the level of details needed in your solutions.

Proof.

(a) To show that the collection of decidable languages is closed under complementation, let L be a decidable language. Thus, L is accepted by a deterministic Turing machine $M = (Q, \Sigma, \delta, q_0, q_{acc}, q_{rej})$ that halts on all inputs. Since the Turing machine M is deterministic, for every input x, if $x \in L$, then the (unique) computational path of M on input x runs and stops at its accepting state q_{acc}, while if $x \notin L$, then the (unique) computational path of M on input x runs and stops at its rejecting state q_{rej}. Now we swap the accepting state q_{acc} and the rejecting state q_{rej} in M, we get a new Turing machine $M' = (Q, \Sigma, \delta, q_0, q'_{acc}, q'_{rej})$, where $q'_{acc} = q_{rej}$ and $q'_{rej} = q_{acc}$ (i.e., M' has the same state set Q, the same alphabet Σ, the same transition function δ, and the same start state q_0. However, M' uses q_{rej} as its accepting state and q_{acc} as its rejecting state). Note that the Turing machine M' is also deterministic. On any input x, if $x \in L$, then the (unique) computational path of M' on input x runs and stops at its rejecting state $q'_{rej} = q_{acc}$, while if $x \notin L$, then the (unique) computational path of M' on input x runs and stops at its accepting state $q'_{acc} = q_{rej}$. Thus, the Turing machine M' accepts exactly the language $\overline{L}$ that is the complement of L. Since the Turing machine M' halts on all inputs, the language $\overline{L}$ is decidable. This completes the proof that the complement $\overline{L}$ of a decidable language L is also decidable, i.e., the collection of decidable languages is closed under complementation.

(b) To show that the collection of decidable languages is closed under intersection, let L_1 and L_2 be two decidable languages. Thus, L_1 and L_2 are accepted by deterministic Turing machines M_1 and M_2, respectively, where both M_1 and M_2 halt on all inputs. Now consider the following Turing machine $M_\cap$:

$M_\cap$ on input x
(1) run M_1 on x, if M_1 rejects x, then reject;
(2) run M_2 on x, if M_2 rejects x, then reject;
(3) accept x.

By our assumption, the Turing machines M_1 and M_2 halt on all inputs. Thus, if the algorithm reaches step (3), then both Turing machines M_1 and M_2 accept the input x (in steps (1) and (2), respectively), i.e., $x \in L_1 \cap L_2$. On the other hand, if $x \notin L_1 \cap L_2$, then either M_1 or M_2 would reject x so the Turing machine $M_\cap$ would reject x in either step (1) or step (2). In conclusion, the Turing machine $M_\cap$ accepts the language $L_1 \cap L_2$. Finally, the Turing machine $M_\cap$ halts on all inputs since the Turing machines M_1 and M_2 halt on all inputs. This proves that the language $L_1 \cap L_2$ is decidable. As a consequence, the collection of decidable languages is closed under intersection. $\square$

3. Show that the collection of Turing-recognizable languages is closed under the operation of intersection. Use the solution for Problem 3.16(a) in the textbook (page 191) as a guide for the level of details needed in your solutions.

Proof. To show that the collection of Turing-recognizable languages is closed under intersection, let L_1 and L_2 be two Turing-recognizable languages, which are recognized by two deterministic Turing machines M_1 and M_2, respectively, where for each $i = 1, 2$, if $x \in L_i$, then the Turing machine M_i on input x will stop at its accepting state, while if $x \notin L_i$, then the Turing machine M_i on input x will either stop at its rejecting state or loop without stopping. Now consider the following Turing machine $M_\cap$:

$M_\cap$ on input x
(1) run M_1 on x, if M_1 rejects x, then reject;
(2) run M_2 on x, if M_2 rejects x, then reject;
(3) accept x.

We show that the Turing machine $M_\cap$ recognizes the language $L_1 \cap L_2$. Let $x \in L_1 \cap L_2$, then $x \in L_1$ and $x \in L_2$. Thus, both Turing machines M_1 and M_2 on the input x accept x (i.e., halt at their corresponding accepting states). As a consequence, on the input $x \in L_1 \cap L_2$, step (1) of the Turing machine $M_\cap$ will not run into a dead loop, but eventually find out that the Turing machine M_1 accepts x. Thus, step (1) of $M_\cap$ will not reject x but eventually move to step (2). Similarly, step (2) of $M_\cap$ will not reject x but eventually move to step (3), which will accept x. This shows that for an input $x \in L_1 \cap L_2$, the Turing machine $M_\cap$ will accept x and halt.

Now consider an input $x \notin L_1 \cap L_2$. We have either $x \notin L_1$ or $x \notin L_2$ (or both). If $x \notin L_1$, then in step (1) of $M_\cap$, either (i) M_1 halts and rejects x – so $M_\cap$ rejects x, or (ii) M_1 on x runs into a dead loop – then $M_\cap$ on x also runs into a dead loop without stopping. Similarly, if $x \notin L_2$, then step (2) of $M_\cap$, thus the Turing machine $M_\cap$, will either reject x or run into a dead loop. Summarizing the discussion, for $x \notin L_1 \cap L_2$, the Turing machine $M_\cap$ on x will either reject x or run into a dead loop.

Combining the above discussions, we conclude that the Turing machine $M_\cap$ recognizes the language $L_1 \cap L_2$, i.e., the language $L_1 \cap L_2$ is Turing-recognizable. This completes the proof that the collection of Turing-recognizable languages is closed under intersection. $\square$

Remark. Comparing Questions 2 and 3, you might wonder why we were not asked to prove that the collection of Turing-recognizable languages is closed under complementation, as we did for the collection of decidable languages. The reason is that the collection of Turing-recognizable languages is *not* closed under complementation. Students are invited to think why a proof similar to that for Question 2(a) on decidable languages will not work for Turing-recognizable languages.

Question 5 in this homework set will also give a hint that the collection of Turing-recognizable languages is not closed under complementation: it claims that if the collection of Turing-recognizable languages is closed under complementation, then every Turing-recognizable language is decidable.

4. Prove that the following languages are decidable.

(a) (CSCE 433 students only) $L_{433} = \{\langle A \rangle \mid A \text{ is a DFA and } L(A) = \Sigma^*\}$.

(b) (CSCE 627 students only) $L_{627} = \{\langle G \rangle \mid G \text{ is a CFG that generates } \epsilon\}$.

Proof.

(a) Note that for a DFA A, $L(A) = \Sigma^*$ if and only if the complement $\overline{L(A)}$ of $L(A)$ is the empty language $\emptyset$. Thus, we only need to construct a DFA $\overline{A}$ that accepts the complement of $L(A)$ then test if $\overline{A}$ accepts the empty language $\emptyset$. As we studied in class, the construction of the DFA $\overline{A}$ to accept $\overline{L(A)}$ is quite easy: we simply swap the final states and the non-final states in A (note that A is deterministic). Finally, as given in the textbook, checking whether the DFA $\overline{A}$ accepts the empty language $\emptyset$ can be implemented by checking whether there is path in the DFA $\overline{A}$ from the start state to a finite state, which can be done using, for example, Depth-First Search on the state diagram of $\overline{A}$, starting from the start state of $\overline{A}$. The algorithm (i.e., Turing machine) M_{433} is given as follows:

M_{433} on input $\langle A \rangle$
1. construct $\langle \overline{A} \rangle$, where $\overline{A}$ is the DFA that accepts $\overline{L(A)}$. This can be done by swapping the final states and the non-final states in the DFA A;
2. construct the state diagram $G(\overline{A})$ for the DFA $\overline{A}$;
3. **if** (there is a path from the start state to a final state in $G(\overline{A})$)
   then reject **else** accept.

As explained above, this Turing machine M_{433} correctly accepts the language L_{433}. Moreover, there is no place to make the Turing machine M_{433} run into dead loop, i.e., the Turing machine M_{433} halts on all inputs. Thus, the language L_{433} accepted by this Turing machine M_{433} that always halts is decidable.

(b) One way to prove this is to first convert the given CFG G into an equivalent CFG G' in Chomsky Normal Form, where we assume that the start variable of G' is S'. The algorithm for converting a CFG into an equivalent CFG in Chomsky Normal From is given in the textbook (Theorem 2.9, page 109), which was also discussed in detail in our class. Note that the CFG G' in Chomsky Normal Form generates ϵ if and only if it has a production rule $S' \to \epsilon$ (where S' is the start variable of G'), which can be easily checked. This then completes the proof that the language L_{627} is decidable.

We can also give a direct proof that is based on an algorithm that repeatedly eliminates production rules of the form $X \to \epsilon$, where X is not the start variable. The algorithm uses the same ideas to eliminate production rules of the form $X \to \epsilon$ for non-start variables X, as used in the proof of Theorem 2.9 in the textbook. The algorithm (i.e., the Turing machine) M_{627} is given as follows.

M_{627} on input $\langle G \rangle$, where G is a CFG with a start variable S
1. let P be the set of all production rules for G;
2. **while** (there is a production rule $X \to \epsilon$ in P, where $X \neq S$)
   delete $X \to \epsilon$ in P;
   for (each production rule of the form $Y \to \alpha X \beta$)
   if ($Y \to \alpha\beta$ is not in P) **then** add $Y \to \alpha\beta$ to P;
3. **if** ($S \to \epsilon$ is in P)
   then accept **else** reject.

The correctness of the Turing machine M_{627} can be proved using induction on the number of production rules of the form $X \to \epsilon$ in the set P, which is omitted here. To see that the Turing machine M_{627} halts on all inputs, note that we neither introduce new symbols nor add production rules to P that are longer (i.e., containing more symbols on its right side) than the production rules in the original CFG G. Thus, for a given CFG G, the number of production rules that can be added is a finite number, which implies that the **while**-loop in step 2 will eventually terminate and move to step 3, which will stop the Turing machine M_{627}. This completes the proof that the language L_{627} is accepted by the Turing machine M_{627} that halts on all inputs. As a consequence, the language L_{627} is decidable. $\square$

5. Prove: let L be a language such that both L and the complement $\overline{L}$ of L are Turing-recognizable, then L is decidable. The level of details of your proof should be similar to that for Questions 2-3 above.

Proof. Since both L and $\overline{L}$ are Turing-recognizable, there are two deterministic Turing machines M and $\overline{M}$ such that for any $x \in L$, the Turing machine M on input x runs in a finite number of steps then halts and accepts x (on the other hand, the Turing machine $\overline{M}$ on x may halt and reject x or run into a dead loop), while for any $y \in \overline{L}$, i.e., $y \notin L$, the Turing machine $\overline{M}$ on input y runs in a finite number of steps then halts and accepts y (again, the Turing machine M on y may halt and reject y or run into a dead loop). Now consider the following Turing machine M':

M' on input x
1. $k = 1$;
2. loop
2.1 run M on x for k steps, if M accepts x in no more than k steps, then accept;
2.2 run $\overline{M}$ on x for k steps, if $\overline{M}$ accepts x in no more than k steps, then reject;
2.3 $k = k + 1$.

Note that for each fixed k, steps 2.1 and 2.2 run the Turing machines M and $\overline{M}$, respectively, at most k steps. Thus, for each fixed k, the execution of steps 2.1 and 2.2 can never run into a dead loop.

We show that the Turing machine M' accepts the language L and halts on all inputs. Let x be any input to M'. If $x \in L$, then the Turing machine M accepts x in a finite number k_1 of steps. Thus, when step 2 of the Turing machine M' reaches the number $k = k_1$, step 2.1 of the Turing machine M' will find out that M accepts x in no more than k_1 steps, so M' accepts x (and halts) at step 2.1. On the other hand, if $x \notin L$, i.e., if $x \in \overline{L}$, then the Turing machine $\overline{M}$ accepts x in a finite number k_0 of steps. Thus, when step 2 of the Turing machine M' reaches the number $k = k_0$, step 2.1 of the Turing machine M' will find out that $\overline{M}$ accepts x in no more than k_0 steps, so M' rejects x (and halts) at step 2.2. Therefore, for any input x, the Turing machine M' on input x will always halt, accepts x if $x \in L$ and rejects x if $x \notin L$. Therefore, the language L is accepted by the Turing machine M' that halts on all inputs. This proves that the language L is decidable. $\square$

Remark. This remark echoes the remark in Question 3. The result of Question 5 shows that if the collection of Turing-recognizable languages is closed under complementation, then every Turing-recognizable language would be also decidable. This, as we have seen in class, is not true. For example, the HALTING problem is Turing-recognizable but not decidable.