Formal Languages and Automata Theory

End-Semester Test

Maximum marks: 60 | Time: April 12, 2022 | Duration: 11am – 01pm

---

1. Consider the following language over the alphabet $\{0,1,\#\}$.

$$L_1 = \left\{ x\,\#\,y \mid x,y \in \{0,1\}^*,\ x \neq y,\ |x| = |y| \right\}.$$

Here, $|w|$ denotes the length of the string w. Prove/Disprove: L_1 is context-free. **(10)**

1. Consider the following language over the alphabet $\{a,b,\#\}$.

$$L_1 = \left\{ x\,\#\,y \mid x,y \in \{a,b\}^*,\ x \neq y,\ |x| = |y| \right\}.$$

Here, $|w|$ denotes the length of the string w. Prove/Disprove: L_1 is context-free. **(10)**

1. Consider the following language over the alphabet $\{0,1,\#\}$.

$$L_1 = \left\{ x\,\#\,y \mid x,y \in \{0,1,\#\}^*,\ x \neq y,\ |x| = |y| \right\}.$$

Here, $|w|$ denotes the length of the string w. Prove/Disprove: L_1 is context-free. **(10)**

1. Consider the following language over the alphabet $\{a,b,\#\}$.

$$L_1 = \left\{ x\,\#\,y \mid x,y \in \{a,b,\#\}^*,\ x \neq y,\ |x| = |y| \right\}.$$

Here, $|w|$ denotes the length of the string w. Prove/Disprove: L_1 is context-free. **(10)**

Solution Consider the language

$$L_1 = \{x\,\#\,y \mid x \neq y, |x| = |y|\}.$$

Here, x,y are in $\{c,d\}^*$ or $\{c,d,\#\}^*$. This language is not context-free. We prove this by the pumping lemma.

Suppose that L_1 is context-free, and let k be a pumping-lemma constant for L_1. Consider the string

$$z = c^{k+k!}d^k\#c^kd^{k+k!} \in L_1.$$

The pumping lemma gives a decomposition of this string of the form $z = uvwxy$ such that $vx \neq \varepsilon$, $|vwx| \leqslant k$, and $z_i = uv^iwx^iy \in L_1$ for all $i \geqslant 0$. We consider several cases.

Case 1: vx contains #. Then, z_0 does not contain any #.

Case 2: v and x are both to the left of #, or both to the right of #. Then, for all $i \neq 1$, the two sides of # in z_i are of unequal lengths.

Case 3: v is to the left of # and x is to the right of #. If $|v| \neq |x|$, then again for $i \neq 1$, the two sides of # in z_i are of unequal lengths. So we must have $|v| = |x| \neq 0$. Since $|vwx| \leqslant k$, we must have v in the left block of d's, and x in the right block of c's. Since $1 \leqslant l = |v| = |x| \leqslant k$, we conclude that l is a divisor of $k!$. But then, $z_{1+k!/l} = c^{k+k!}d^{k+k!}\#c^{k+k!}d^{k+k!} \in L_1$.

---

2. Design a DPDA (deterministic pushdown automaton) to accept the language

$$L_2 = \left\{ a^mb^n \mid m,n \geqslant 0,\ \text{and}\ 2m - 3n = 5 \right\}.$$

Your DPDA should loop in only two distinguished states t and r. There must not be any other infinite loops or cycles. The DPDA enters these states after reading the entire input (including the end-of-input marker), and t is the only final state, whereas r is a non-final state. Show all the transitions clearly. **(10)**

2. Design a DPDA (deterministic pushdown automaton) to accept the language

$$L_2 = \left\{ a^m b^n \;\middle|\; m,n \geqslant 0, \text{ and } 3m - 2n = 5 \right\}.$$

Your DPDA should loop in only two distinguished states t and r. There must not be any other infinite loops or cycles. The DPDA enters these states after reading the entire input (including the end-of-input marker), and t is the only final state, whereas r is a non-final state. Show all the transitions clearly. **(10)**

Solution Consider the language

$$L_2 = \{a^m b^n \mid m,n \geqslant 0, \text{ and } um - vn = w\}.$$

Here, u, v, w are constant positive integers. In the start state s, the DPDA consumes the a's, and pushes u symbols like A to the stack for each a.

$$\begin{aligned} \delta(s,a,\bot) &= (s,A^u\bot) \\ \delta(s,a,A) &= (s,A^{u+1}) \end{aligned}$$

If anything else appears in the input, the DPDA manages by moving to state p_1 using an ε-transition keeping its stack intact.

$$\delta(s,\varepsilon,*) = (p_1,*)$$

At state p_1, the DPDA consumes the b's from the input, and for each b, should be able to pop v A's from the stack. But only one pop is allowed for each transition, so we add temporary states $p_2, p_3, \ldots, p_v$.

$$\begin{aligned} \delta(p_1,b,A) &= (p_2,\varepsilon) \\ \delta(p_2,\varepsilon,A) &= (p_3,\varepsilon) \\ \delta(p_3,\varepsilon,A) &= (p_4,\varepsilon) \\ &\cdots \\ \delta(p_{v-1},\varepsilon,A) &= (p_v,\varepsilon) \\ \delta(p_v,\varepsilon,A) &= (p_1,\varepsilon) \end{aligned}$$

After all the b's are read, the end of input is exposed, and the DPDA discards w A's from the top, and eventually accepts by looping in state t.

$$\begin{aligned} \delta(p_1,\dashv,A) &= (t_1,\varepsilon) \\ \delta(t_1,\varepsilon,A) &= (t_2,\varepsilon) \\ \delta(t_2,\varepsilon,A) &= (t_3,\varepsilon) \\ &\cdots \\ \delta(t_{w-1},\varepsilon,A) &= (t_w,\varepsilon) \\ \delta(t_w,\varepsilon,\bot) &= (t,\bot) \\ \delta(t,\varepsilon,\bot) &= (t,\bot) \end{aligned}$$

Let us now see what happens if the input is not accepted. This may happen in the following cases.

1. An a is read in state p_1.

   $$\delta(p_1,a,*) = (r',*)$$

2. A is not on the top of the stack in some p_i.

   $$\begin{aligned} \delta(p_1,b,\bot) &= (r',\bot) \\ \delta(p_i,\varepsilon,\bot) &= (r',\bot) \text{ for } i = 2,3,\ldots,v \end{aligned}$$

3. Less than w A's are in the stack after all b's are read.

$$\begin{aligned} \delta(p_1, \dashv, \perp) &= (r, \perp) \\ \delta(t_i, \varepsilon, \perp) &= (r, \perp) \text{ for } i = 1, 2, \ldots, w-1 \end{aligned}$$

4. Excess A's remain in the stack.

$$\delta(t_w, \varepsilon, A) = (r, A)$$

In the state r', the DPDA reads the rest of the input and eventually goes to r.

$$\begin{aligned} \delta(r', a, *) &= (r', *) \\ \delta(r', b, *) &= (r', *) \\ \delta(r', \dashv, *) &= (r, *) \end{aligned}$$

2. Design a DPDA (deterministic pushdown automaton) to accept the language

$$L_2 = \left\{ a^m b^n \;\middle|\; m, n \geqslant 0, \text{ and } 2n - 3m = 5 \right\}.$$

Your DPDA should loop in only two distinguished states t and r. There must not be any other infinite loops or cycles. The DPDA enters these states after reading the entire input (including the end-of-input marker), and t is the only final state, whereas r is a non-final state. Show all the transitions clearly. **(10)**

2. Design a DPDA (deterministic pushdown automaton) to accept the language

$$L_2 = \left\{ a^m b^n \;\middle|\; m, n \geqslant 0, \text{ and } 3n - 2m = 5 \right\}.$$

Your DPDA should loop in only two distinguished states t and r. There must not be any other infinite loops or cycles. The DPDA enters these states after reading the entire input (including the end-of-input marker), and t is the only final state, whereas r is a non-final state. Show all the transitions clearly. **(10)**

Solution Consider the language

$$L_2 = \{a^m b^n \mid m, n \geqslant 0, \text{ and } un - vm = w\}.$$

Here, u, v, w are constant positive integers. The condition can be rewritten as

$$w + vm - un = 0.$$

The DPDA starts by pushing w symbols (like A) to the stack.

$$\delta(s, \varepsilon, \perp) = (p, A^w \perp)$$

In the state p, v A's are pushed for each a consumed from the input.

$$\delta(p, a, A) = (p, A^{v+1})$$

If anything else appears in the input, the DPDA manages by moving to state q_1 using an ε-transition keeping its stack intact.

$$\delta(p, \varepsilon, *) = (q_1, *)$$

At state q_1, the DPDA consumes the b's from the input, and for each b, should be able to pop u A's from the stack. But only one pop is allowed for each transition, so we add temporary states $q_2, q_3, \ldots, q_v$.

$$\begin{aligned} \delta(q_1, b, A) &= (q_2, \varepsilon) \\ \delta(q_2, \varepsilon, A) &= (q_3, \varepsilon) \\ \delta(q_3, \varepsilon, A) &= (q_4, \varepsilon) \\ &\cdots \\ \delta(q_{u-1}, \varepsilon, A) &= (q_u, \varepsilon) \\ \delta(q_u, \varepsilon, A) &= (q_1, \varepsilon) \end{aligned}$$

After all the b's are read, both the end of input and the stack bottom marker should be exposed. This leads to acceptance.

$$\begin{aligned} \delta(q_1, \dashv, \perp) &= (t, \perp) \\ \delta(t, \varepsilon, \perp) &= (t, \perp) \end{aligned}$$

Let us now see what happens if the input is not accepted. This may happen in the following cases.

1. An a is read in state q_1.

   $$\delta(q_1, a, *) = (r', *)$$

2. A is not on the top of the stack in some q_i.

   $$\begin{aligned} \delta(q_1, b, \perp) &= (r', \perp) \\ \delta(q_i, \varepsilon, \perp) &= (r', \perp) \text{ for } i = 2, 3, \ldots, u \end{aligned}$$

3. Excess A's remain in the stack.

   $$\delta(q_1, \dashv, A) = (r, A)$$

In the state r', the DPDA reads the rest of the input and eventually goes to r.

$$\begin{aligned} \delta(r', a, *) &= (r', *) \\ \delta(r', b, *) &= (r', *) \\ \delta(r', \dashv, *) &= (r, *) \end{aligned}$$

---

3. For a language L over the alphabet $\{0, 1\}$, define the language

$$\text{half}(L) = \Big\{ x \;\Big|\; x \in \Sigma^*, \text{ and there exists } y \in \Sigma^* \text{ such that } |x| = |y| \text{ and } xy \in L \Big\}.$$

Prove/Disprove the following three statements.

(a) If L is context-free, then half(L) nust be context-free. **(8)**

(b) If L is recursively enumerable, then half(L) must be recursively enumerable. **(7)**

(c) If L is recursive, then half(L) must be recursive. **(5)**

Solution (a) *False* Take $L = \{0^n 1^n 0^m 1 0^{3m} 1 \mid n, m \geqslant 1\}$. You can design a CFG for L (do it), so L is context-free. If half(L) is context-free, than so also is half$(L) \cap \mathscr{L}(0^+1^+0^+1)$. But the latter language is $\{0^n 1^n 0^n 1 \mid n \geqslant 1\}$ which is not context-free (use a proof similar to that for the language $\{0^n 1^n 0^n \mid n \geqslant 1\}$).

(b) *True* Let M be a DTM for accepting L. We design an NTM N for half(L) (and then convert N to a DTM D using the usual procedure). N, given an input x, first deterministically computes the length l of x. N then non-deterministically appends a string $y \in \Sigma^*$ of length l to generate the string xy. Subsequently, N simulates M on xy, and accepts if M accepts. If $x \in$ half(L), then some guess for y lets the simulation accept.

(c) *True* The construction in this part is the same as in Part (b). Since L is recursive, we take M as a total TM. There are finitely many guesses ($|\Sigma|^l$ to be precise, where $l = |x|$) for y. Since M is total, each guess gives a string xy that can be accepted/rejected in finite time by the simulation of M. Therefore, N and the converted DTM D are total too.

3. For a language L over the alphabet $\{a, b\}$, define the language

$$\text{half}(L) = \Big\{ x \;\Big|\; x \in \Sigma^*, \text{ and there exists } y \in \Sigma^* \text{ such that } |x| = |y| \text{ and } xy \in L \Big\}.$$

Prove/Disprove the following three statements.

(a) If L is context-free, then half(L) nust be context-free. **(8)**

(b) If L is recursively enumerable, then half(L) must be recursively enumerable. **(7)**

(c) If L is recursive, then half(L) must be recursive. **(5)**

Solution (a) Replace 0 by a and 1 by b in the previous variant.

3. For a language L over the alphabet $\{0,1\}$, define the language

$$\text{half}(L) = \Big\{ x \mid x \in \Sigma^*, \text{ and there exists } y \in \Sigma^* \text{ such that } |x| = |y| \text{ and } yx \in L \Big\}.$$

Prove/Disprove the following three statements.

(a) If L is context-free, then $\text{half}(L)$ nust be context-free. **(8)**

(b) If L is recursively enumerable, then $\text{half}(L)$ must be recursively enumerable. **(7)**

(c) If L is recursive, then $\text{half}(L)$ must be recursive. **(5)**

Solution (a) Take $L = \{10^{3m}10^m1^n0^n \mid m,n \geqslant 1\}$. Then, $\text{half}(L) \cap \mathscr{L}(10^+1^+0^+) = \{10^n1^n0^n \mid n \geqslant 1\}$.

3. For a language L over the alphabet $\{a,b\}$, define the language

$$\text{half}(L) = \Big\{ x \mid x \in \Sigma^*, \text{ and there exists } y \in \Sigma^* \text{ such that } |x| = |y| \text{ and } yx \in L \Big\}.$$

Prove/Disprove the following three statements.

(a) If L is context-free, then $\text{half}(L)$ nust be context-free. **(8)**

(b) If L is recursively enumerable, then $\text{half}(L)$ must be recursively enumerable. **(7)**

(c) If L is recursive, then $\text{half}(L)$ must be recursive. **(5)**

Solution (a) Replace 0 by a and 1 by b in the previous variant.

---

4. Consider the following language over the alphabet $\Sigma = \{a,b,c\}$.

$$L_4 = \Big\{ a^n w c^n \;\Big|\; w \in \Sigma^*,\ n \geqslant 0,\ \text{ and } \#a(w) = n \Big\}$$

Here, $\#a(w)$ denotes the number of a's in the string w. Design an unrestricted grammar for L_4. Explain the roles played by the non-terminal symbols of your grammar. **(8)**

Solution We use a derivation of the form

$$S \to^* a^nT(Uc)^n \to^* a^nTU^nc^n \to^* a^nT(aV)^nc^n \to a^nV(aV)^nc^n.$$

Each V generates a string over $\{b,c\}$. Thus, we use the following productions.

$$\begin{array}{rcl} S & \to & aSUc \mid T \\ cU & \to & Uc \\ aU & \to & Ua \\ VU & \to & UV \\ TU & \to & aV \\ T & \to & V \\ V & \to & bV \mid cV \mid \varepsilon \end{array}$$

4. Consider the following language over the alphabet $\Sigma = \{a,b,c\}$.

$$L_4 = \Big\{ a^n w c^n \;\Big|\; w \in \Sigma^*,\ n \geqslant 0,\ \text{ and } \#b(w) = n \Big\}$$

Here, $\#b(w)$ denotes the number of b's in the string w. Design an unrestricted grammar for L_4. Explain the roles played by the non-terminal symbols of your grammar. **(8)**

4. Consider the following language over the alphabet $\Sigma = \{a, b, c\}$.

$$L_4 = \left\{ a^n w c^n \;\middle|\; w \in \Sigma^*,\ n \geqslant 0,\ \text{and}\ \#c(w) = n \right\}$$

Here, $\#c(w)$ denotes the number of c's in the string w. Design an unrestricted grammar for L_4. Explain the roles played by the non-terminal symbols of your grammar. **(8)**

4. Consider the following language over the alphabet $\Sigma = \{a, b, c\}$.

$$L_4 = \left\{ a^n w b^n \;\middle|\; w \in \Sigma^*,\ n \geqslant 0,\ \text{and}\ \#c(w) = n \right\}$$

Here, $\#c(w)$ denotes the number of c's in the string w. Design an unrestricted grammar for L_4. Explain the roles played by the non-terminal symbols of your grammar. **(8)**

Solution Make appropriate changes in the grammar of the first variant for the other variants.

---

5. Consider the language

$$L_5 = \left\{ M \;\middle|\; M \text{ is (the encoding of) a deterministic Turing machine that loops on at most 2022 input strings} \right\}.$$

Prove/Disprove:

(a) L_5 is recursively enumerable. **(6)**

(b) $\overline{L_5}$ (that is, the complement of L_5) is recursively enumerable. **(6)**

5. Consider the language

$$L_5 = \left\{ M \;\middle|\; M \text{ is (the encoding of) a deterministic Turing machine that loops on at least 2022 input strings} \right\}.$$

Prove/Disprove:

(a) L_5 is recursively enumerable. **(6)**

(b) $\overline{L_5}$ (that is, the complement of L_5) is recursively enumerable. **(6)**

5. Consider the language

$$L_5 = \left\{ M \;\middle|\; M \text{ is (the encoding of) a deterministic Turing machine that loops on less than 2022 input strings} \right\}.$$

Prove/Disprove:

(a) L_5 is recursively enumerable. **(6)**

(b) $\overline{L_5}$ (that is, the complement of L_5) is recursively enumerable. **(6)**

5. Consider the language

$$L_5 = \left\{ M \;\middle|\; M \text{ is (the encoding of) a deterministic Turing machine that loops on more than 2022 input strings} \right\}.$$

Prove/Disprove:

(a) L_5 is recursively enumerable. **(6)**

(b) $\overline{L_5}$ (that is, the complement of L_5) is recursively enumerable. **(6)**

Solution Let k be a constant positive integer. Consider the following four languages.

$$\begin{aligned} LE_k &= \{M \mid M \text{ loops on at most } k \text{ inputs}\} \\ GE_k &= \{M \mid M \text{ loops on at least } k \text{ inputs}\} \\ LT_k &= \{M \mid M \text{ loops on less than } k \text{ inputs}\} \\ GT_k &= \{M \mid M \text{ loops on more than } k \text{ inputs}\} \end{aligned}$$

All these languages are non-RE and non-co-RE.

First, note that $LE_k = LT_{k+1}$, $LT_k = LE_{k-1}$, $GE_k = GT_{k-1}$, and $GT_k = GE_{k+1}$. Moreover, $\overline{LE_k} = GT_k$, $\overline{LT_k} = GE_k$, $\overline{GE_k} = LT_k$, and $\overline{GT_k} = LE_k$. In view of these, the following two reductions suffice for all the four parts.

$\overline{\text{HP}} \leqslant LE_k$ and $\overline{\text{HP}} \leqslant LT_k$

N, on input y, does the following.

1. Simulate M on x for $|y|$ steps.
2. If the limited-time simulation does not halt, halt.
3. If the limited-time simulation halts, loop.

Let k' be the number of inputs, on which N loops.

If M does not halt on x, then M does not halt on x in any finite number of steps, so N halts on all inputs. Therefore $k' = 0$, and so $k' \leqslant k$ and $k' < k$.

If M halts on x in s steps, then for all inputs y of length $\geqslant s$, N loops. In particular, N loops on infinitely many inputs in this case, that is, $k' = \infty$, and we have $k' > k$ and $k' \geqslant k$.

$\overline{\text{HP}} \leqslant GE_k$ and $\overline{\text{HP}} \leqslant GT_k$

N, on input y, does the following.

1. Simulate M on x.
2. Halt.

Let k' be the number of inputs, on which N loops.

If M does not halt on x, then N loops on all inputs, that is, $k' = \infty$, and we have $k' \geqslant k$ and $k' > k$.

If M halts on x, then N halts on all inputs, that is, $k' = 0$, and we have $k' < k$ and $k' \leqslant k$.
