Tutorial 3A: Turing Machines and Unrestricted Grammars

Problem 1 A linear bounded automaton (LBA) is exactly like a 1-tape TM, except that the input string $x \in \Sigma^*$ is enclosed in left and right end markers $\vdash$ and $\dashv$ which may not be overwritten. The machine is constrained to never move left of $\vdash$ or right of $\dashv$. It is allowed to read/write between these markers.

(a) Give a rigorous formal definition of deterministic linearly bounded automata, including a definition of configurations and acceptance.

(b) Let M be a LBA with state set Q of size k and tape alphabet Γ of size m. How many possible configurations are there on input x with $|x| = n$?

(c) Argue that it is possible to detect in finite time whether an LBA loops on a given input.

Solution (a) M is a deterministic linear bounded automata:

$$M = (Q, \Sigma, \Gamma, \vdash, \dashv, \delta, s, t, r)$$

where Q is the (finite) set of states,
Σ is the input alphabet,
Γ is the tape alphabet ($\Sigma \subset \Gamma$),
$\vdash$ is the left end marker,
$\dashv$ is the right end marker ($\vdash, \dashv \in \Gamma \setminus \Sigma$)
δ is the transition function,
s is the start state,
t is the accept state,
r is the reject state ($s, t, r \in Q$).

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$$

such that

$$\forall q \in Q, \quad \delta(q, \vdash) = (p, \vdash, R) \quad \text{for some } p \in Q$$
$$\forall q \in Q, \quad \delta(q, \dashv) = (p, \dashv, L) \quad \text{for some } p \in Q.$$

Let the input to the LBA be x, $|x| = n$.

A configuration of M is a three-tuple: $C \in Q \times \Gamma^n \times \{0, ..., n+1\}$

$$C = (q, z, m)$$

which indicates
- the current state is $q \in Q$,
- the current contents of the tape is $\vdash z \dashv$, $z \in \Gamma^n$,
- the head pointer points at position m, $m \in \{0, ..., n+1\}$.

We can define the relation $\xrightarrow[M]{*}$ between configurations as usual.

Then, M is said to accept x if $(s, x, 0) \xrightarrow[M]{*} (t, y, l)$ for some $y \in \Gamma^n, l \in \{0, ..., n+1\}$.

So,

$$L(M) = \left\{ x \mid (s, x, 0) \xrightarrow[M]{*} (t, y, l) \right\}$$

(b) A configuration is described by $Q \times \Gamma^n \times \{0, 1, ..., n+1\}$. So, the total number of possible configurations are $|Q| \times |\Gamma|^n \times (n+2) = k(n+2)m^n$.

(c) An LBA either halts in finite steps, or loops forever. Since the LBA is deterministic, if an LBA reaches the same configuration twice for some input, it must loop on that input.

Let $c = k(n+2)m^n$ denote the total number of possible configurations.

Let's run the LBA for c steps, then it will have gone through at most $c+1$ configurations in total. If it halted before completion of all these steps, we know that it halts on the given input. If it did not halt yet, we know that it **must** loop on this input, since it has seen a configuration twice by now. Thus we can determine in finite time whether an LBA loops on a given input. □

Problem 2 Design an NTM to accept the language

$$\{wxyxz \mid w, x, y, z \in \{0,1\}^*, |x| = 2026\}$$

Solution N on input x:

1. Move right. At each step, non-deterministically either continue moving right or proceed to Step 2.
2. Repeat 2026 times: mark the symbol under the head with $\widehat{\ }$, then move right. If the input ends before all 2026 marks are placed, reject.
3. Move right. At each step, non-deterministically either continue moving right or proceed to Step 4.
4. Repeat 2026 times: mark the symbol under the head with $'$, then move right. If the input ends before all 2026 marks are placed, reject.
5. Repeat 2026 times: Scan to the leftmost $\widehat{\ }$-marked symbol; call its value a. Scan to the leftmost $'$-marked symbol; call its value b.
   - If $a \neq b$, reject.
   - Otherwise, erase the $\widehat{\ }$ marker from the previous cell, and erase the $'$ marker from your cell.
6. Accept. □

Problem 3 A TM $\mathcal{M}$ has a two-way infinite tape. Initially all cells on the tape are blank. Only one cell is storing the symbol #. The head of $\mathcal{M}$ is pointing to a blank. The task of $\mathcal{M}$ is to locate the cell storing #.

Propose a strategy for doing this,
(a) if $\mathcal{M}$ is a DTM,
(b) if $\mathcal{M}$ is an NTM.

Solution (a) $\mathcal{M}$ on input:

1. If the current cell has #, report that we have located it.
2. Mark the current cell with an X.
3. While the current cell contains X, go left.
4. If the current cell has #, report that we have located it.
5. Mark the current cell with an X.
6. While the current cell contains X, go right.
7. Go to Step 1.

(b) $\mathcal{M}$ on input: Repeat:

1. If the current cell has #, report that we have located it.
2. Non deterministically, either go left or right.

Let us assume that the # is located at a distance n from the initial starting head pointer.

We can observe, that $\mathcal{M}$ in (a) will require $O(n^2)$ steps to find the symbol. On the other hand, $\mathcal{M}$ in (b) being a nondeterministic machine, can find the symbol in $O(n)$ steps. □

Problem 4 A Jump Turing Machine (JTM) $J = (Q, \Sigma, \Gamma, \delta, \vdash, \sqcup, s, t, r)$ is like a standard one-tape Turing Machinge (TM) with the only exception that each transition of J is of the form $\delta(p, A) = (q, B, m)$, where $p, q \in Q, A, B \in \Gamma, m \in \mathbb{Z}$.

This means that if the finite control of J is in the state p and the head of J scans the tape symbol A, then the state changes to q, the content of the tape cell is changed from A to B, and the head jumps by m (integer) cells relative to the current position.

If $m = 0$, the head stays at the current cell. If $m > 0$, the head makes a right jump. If $m < 0$, the head makes a left jump with the understanding that if the head is at position i on the tape, and $|m| > i$, then the head goes to the leftmost cell (which stores the left end-marker $\vdash$). Also assume that if A is $\vdash$, then $m \geq 0$.

Prove that a JTM is equivalent to a TM.

Solution Firstly, any TM can be simulated by a JTM. This is trivial, as any transition $\delta(p, A) = (q, B, L) \equiv \delta'(p, A) = (q, B, -1)$ and $\delta(p, A) = (q, B, R) \equiv \delta'(p, A) = (q, B, +1)$.

For the other direction, let J be a JTM with transition function δ. Note that δ is a finite object, i.e. it has a finite number of transition, each transition can make a finite sized jump.

So we can create a machine $M = (Q, \Sigma, \Gamma, \delta', \vdash, \sqcup, s, t, r)$ which simulates J.

For each transition $\delta(p, A) = (q, B, +m), m > 0$, M simulates it as
1. Write B to the tape.
2. Use a sequence of $m - 1$ intermediate states, scan m steps to the right.

If $m = 0$,
1. Write B to the tape, go to an intermediate state, going right on the input tape.
2. Come back to the original state by going left on the input tape.

The $m < 0$ case is symmetric as $m > 0$. □

Problem 5 Show that the class of recursively enumerable sets is closed under union and intersection.

Solution Let $L_1 = L(M_1)$ and $L_2 = L(M_2)$ be two r.e. sets. Then, define $M_\cup$ and $M_\cap$:

$M_\cup$ on input x: (assume having two tapes)
1. Copy x to the second tape.
2. Parallelly run M_1 and M_2 on both tapes, one at a time.
3. If any one of them accepts, accept. If both reject, reject.

$M_\cap$ on input x:
1. Run M_1 on x. If it rejects, then reject.
2. Run M_2 on x. If it accepts, then accept. If it rejects, then reject.

Then, $L(M_\cup) = L_1 \cup L_2$, $L(M_\cap) = L_1 \cap L_2$. Thus, r.e. sets are closed under union and intersection. $\square$

Tutorial 3B

Problem 1 Design Turing Machines (TMs) as well as unrestricted grammars for the following languages:

(a) $L_1 = \{w \in \{a,b,c\}^* \mid \#a(w) = \#b(w) = \#c(w)\}$
(b) $L_2 = \{ww \mid w \in \{a,b\}^*\}$
(c) $L_3 = \{a^i b^j c^k d^l \mid i = k, j = l\}$
(d) $L_a = \{a^n w c^n \mid w \in \{a,b,c\}^*, n \geq 0, \#a(w) = n\}$
(e) $L_b = \{a^n w c^n \mid w \in \{a,b,c\}^*, n \geq 0, \#b(w) = n\}$
(f) $L_c = \{a^n w c^n \mid w \in \{a,b,c\}^*, n \geq 0, \#c(w) = n\}$

Solution Only unrestricted grammers for some of the languages are shown here.

(a)
$$S \to ABCS \mid \varepsilon$$
$$AB \to BA$$
$$BA \to AB$$
$$BC \to CB$$
$$CB \to BC$$
$$AC \to CA$$
$$CA \to AC$$
$$A \to a$$
$$B \to b$$
$$C \to c$$

(b)
$$S \to aAS \mid bBS \mid T$$
$$Aa \to aA$$
$$Ab \to bA$$
$$Ba \to aB$$
$$Bb \to bB$$
$$AT \to Ta$$
$$BT \to Tb$$
$$T \to \varepsilon$$

(c)
$$S \to UV$$
$$U \to aUc \mid T$$
$$V \to BVd \mid \varepsilon$$
$$cB \to Bc$$
$$TB \to bT$$
$$T \to \varepsilon$$

(d)
$$S \to aSAc \mid VR$$
$$cA \to Ac$$
$$RA \to aVR$$
$$V \to bV \mid cV \mid \varepsilon$$
$$R \to \varepsilon$$

□

Problem 2 Consider the unrestricted grammar over the singleton alphabet $\Sigma = \{a\}$, having the start symbol S, and with the following productions:

$$S \to AS \mid aT$$
$$Aa \to aaaA$$
$$AT \to T$$
$$T \to \varepsilon$$

What is the language generated by this unrestricted grammar? Justify.

Solution Some derivations:

$$S \to aT \to a$$
$$S \to AS \to AaT \to aaaAT \to aaaT \to aaa$$

We can observe, that the language generated by this unrestricted grammar is $\{a^{3^n} \mid n \geq 0\}$. □

Problem 3 Prove that, a language L is recursive if and only if there is an enumeration machine enumerating the strings of L in a non-decreasing order of length (strings of the same length may be arranged in

lexicographic order). For example, strings of $\{0,1\}^*$ would be arranged as $0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, ...$

Solution Let $L = L(M)$ be recursive, and M be a total Turing Machine. Then, we can create E (enumeration machine) as follows:

E runs:
1. Run M on all strings in Σ^* in a non-decreasing order of length (lexicographic order). If M accepts x, then enumerate x.

Thus, E enumerates the strings of L in a non decreasing order of length.

For the other direction, let E be an enumeration machine which enumerates the strings of L in a non-decreasing order of length. Then we can create a total TM M:

M on input x:
1. Run E.
2. If it enumerates x, accept.
3. If it enumerates y, $|y| > |x|$, reject.

Thus, M is a total TM which accepts the strings enumerated by E. □

Problem 4 Prove that any grammar, defined over non terminals N and terminals Σ, can be converted to an equivalent grammar with rules of the form $\alpha A \gamma \to \alpha \beta \gamma$, for $A \in N$ and $\alpha, \beta, \gamma \in (\Sigma \cup N)^*$.

Solution Left as an exercise to the reader.