Lecture One

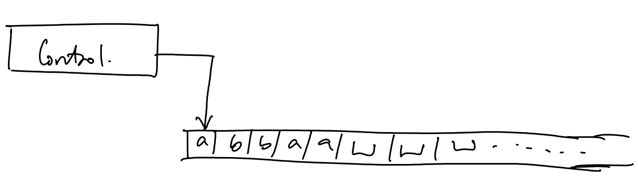
Wednesday, March 17, 2010
10:37 AM

- Finite automaton have limited memory.
- Pushdown automaton, memory only usable as last-in-first out stack.

- Turing Machines proposed by Alan Turing 1936.
  Main differences:
  1) A Turing machine can both write on the tape and read from it.
  2) The read-write head can move both left & right.
  3) The tape is infinite.
  4) The special states for rejecting and accepting take effect immediately.

Formal Defn — A Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where Q, Σ, Γ are all finite sets and.

1. Q is the set of states.
2. Σ is the input alphabet <u>not</u> containing the blank symbol. $\sqcup$.
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$, $\Sigma \subseteq \Gamma$.
4. $\delta: Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$ is the transition fn
5. $q_0 \in Q$ is the start state.
6. $q_{accept} \in Q$ is the accept state.
7. $q_{reject} \in Q$ is the reject state, $q_{accept} \neq q_{reject}$.



- Turing machine M receives input $w = w_1 w_2 \ldots w_n \in \Sigma^*$ on the leftmost n squares of the tape. The rest of the tape is blank. (i.e. filled with $\sqcup$)

- The head starts on the leftmost square of the tape.

- The first $\sqcup$ marks the end of input.

- Computation proceeds according to the rules of the transition function.
- If M ever tries to move head left of the left end, the tape remains at the same place, even though the transition fn indicates L.
- The computation halts on entering accept or reject states. If neither happens M goes on forever.

Configuration of the Turing machine :- uqv.

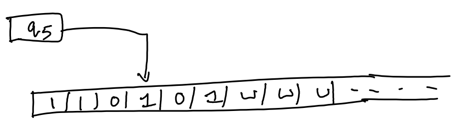
1) Current tape content is uv.

2) Current state is q.

3) Current head location is first symbol of v.

4) Tape contains only blank symbols after the last symbol of v.

$110\,q_5\,101$



- uaq_ibv yields uq_jacv if $\delta(q_i, b) = (q_j, c, L)$
- uaq_ibv yields $uacq_jv$ if $\delta(q_i, b) = (q_j, c, R)$
- q_ibv yields q_jcv if $\delta(q_i, b) = (q_j, c, L)$
- uaq_ib yields $uacq_j\sqcup$ if $\delta(q_i, b) = (q_j, c, R)$

- Start configuration q_0w.
- Accepting configuration the state is q_{accept}.
- Rejecting configuration the state is q_{reject}.

$\left.\right\}$ halting configurations

"M accepts w" if a sequence of configurations $C_1 C_2 \dots C_k$ exist where

1. C_1 is the start configuration q_0w.
2. each C_i yields C_{i+1}.
3. C_k is an accepting configuration.

Defn: Language recognized by M a.k.a the language of M, denoted $L(M)$ is the collection of strings accepted by M

Defn (Turing recognizable) :- Language L is Turing recognizable if some Turing machine recognizes it.

A.k.a. Recursively Enumerable Language.

Defn (Decider) :- A Turing machine that halts on all inputs (never 'loops').
A decider that recognizes L is said to decide L.

Defn (Turing-Decidable): Language L is Turing-decidable or simply decidable if some Turing machine decides it.

A.k.a. Recursive Language.

Example:- $A = \{0^{2^n} \mid n \geq 0\}$.

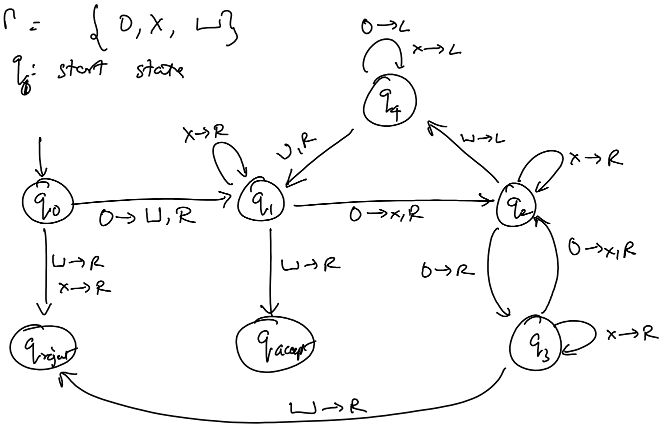
<u>Implimentation level description:-</u>

M = " On input w:

1. Sweep left to right across the tape, crossing off every other 0.
2. If in stage 1, the tape contained a single 0, accept.
3. If in stage 1, the tape contained more than a single 0 and the number of 0's was odd, reject.
4. Return the head to the left hand end of the tape.
5. Go to stage 1.

Formal description:-

- $Q = \{q_0, q_1, q_2, q_3, q_4, q_{accept}, q_{reject}\}$
- $\Sigma = \{0\}$
- $\Gamma = \{0, x, \sqcup\}$
- q_0: start state



1) $q_0 00 \rightarrow \sqcup q_1 0 \rightarrow \sqcup x q_2 \sqcup \rightarrow \sqcup q_4 x \rightarrow q_4 \sqcup x$
$\rightarrow \sqcup q_1 x \rightarrow \sqcup x q_1 \sqcup \rightarrow \sqcup x\, q_{accept} \sqcup$

2) $q_0 000 \rightarrow \sqcup q_1 00 \rightarrow \sqcup x q_2 0 \rightarrow \sqcup x 0 q_3 \sqcup \rightarrow \sqcup x 0 \sqcup q_{reject} \sqcup$

3) $q_0 0 \rightarrow \sqcup q_1 \sqcup \rightarrow \sqcup \sqcup q_{accept} \sqcup$.

Variants of Turing Machines:-

Variants of Turing Machines:-

1. "Stay put feature" $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}$

   Its equivalent.

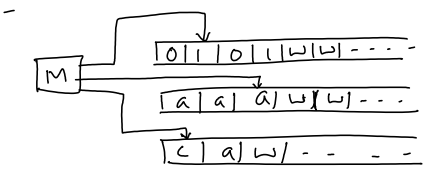
2. Multitape Turing Machine: $\delta: Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R\}^k$

   where k is the number of tapes

   $$\delta(q_i, a_1, \ldots, a_k) = (q_j, b_1, \ldots, b_k, L, R, \ldots, L)$$

Thm:- Every multitape machine has an equivalent single tape machine.

Proof:-



S = " On input $w = w_1 w_2 \ldots w_n$:

1. First S puts its tape into the format that represents all k-tapes of M.

   $\# \dot{w}_1 w_2 \ldots w_n \# \dot{\sqcup} \# \dot{\sqcup} \# \ldots \#$

2. To simulate a single move of multitape machine M,
   S makes a first pass from first # to (k+1)th # to determine the symbols under dots. Then makes second pass to update the tape according to M's transition function.

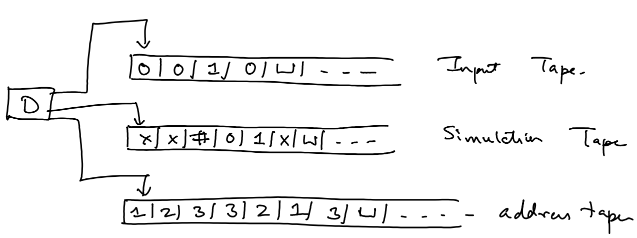
3. If at any point S wants to write dot over a #, then it writes blank symbol ($\sqcup$) there and shifts all the tape content from there one cell to the right. Then it continues simulation as before!!

Corollary:- A language is Turing recognizable if and only if some multitape Turing machine recognize it.

3. Non-Deterministic Turing Machine :-

Thm:- Every non-deterministic Turing Machine N has an equivalent deterministic Turing machine D.

Proof:-



1. Initially tape contains input w, tape 2 and 3 are empty.
2. Copy tape 1 to tape 2.
3. Use tape 2 to simulate N with input w on the branch of

3. Use tape 2 to simulate N with input w on the branch of non-deterministic computation given by tape 3. If no more symbols are left on tape 3 or this address is invalid, goto Stage 4. If a rejecting configuration is encountered, goto stage 4. If an accepting configuration is encountered, accept the input.

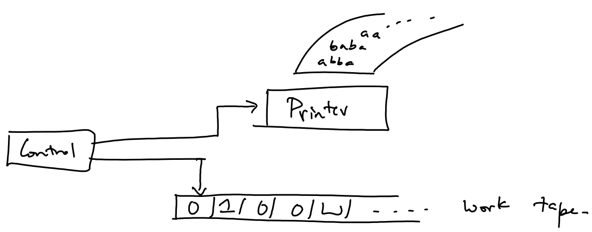
4. Replace string on tape 3 with lexicographically next string. Goto Stage 2.
D.

Corollary:- A language is Turing recognizable iff some non-deterministic Turing machine recognizes it.

- We can modify the construction so that if N halts on all branches of its computation D halts.
- Call a non-deterministic Turing machine "decider" if all branches halt on all inputs.

Corollary:- A language is decidable iff some nondeterministic Turing machine decides it.

Enumerator:-



- Enumerator E starts with blank input tape.
- It may not halt and may print infinite list of strings
- The language enumerated by E is the collection of all strings eventually printed by it.
- E may generate the strings of language in any order possibly with repetitions.

Thm:- A language is Turing-recognizable iff some Enumerator enumerates it.

Proof:- ($\Leftarrow$). Let E be enumerator for language A. Turing machine M recognizing A:

M = " On input w:
1. Run E. Every time E prints a string, compare it with w.
2. If w ever appears, accept."

($\Rightarrow$) Let M recognize A. Enumerator for A:-

E = " Ignore the input.
1. Repeat the following for $i = 1, 2, 3 \ldots$
a. Run M for i steps on each input $s_1, s_2, \ldots, s_i$.
b. If any computations accept, print out the corresponding s_j."

Church-Turing Thesis:-

- Informally speaking an algorithm is a collection of simple instructions for carrying out some tasks.

Intuitive Notion of Algorithms	equals	Turing machine Algorithms

The Church-Turing Thesis

- Alonzo Church in 1936 used a notational system called the λ-calculus to define algorithms.
- Turing did it with his machines, also in 1936.
- The two defn were shown to be equivalent
- Many other defn are also made later but are all shown equivalent.
- Church-Turing thesis states that "any reasonable definition is equivalent".

"Reasonable definition" for example requires that only finite amount of work is done in single step.

Hilbert's Tenth Problem:-

- In 1900, David Hilbert proposed 23 problems for the coming century. at International Congress of Mathematics in Paris.
- The tenth problem was to find "a process according to which it can be determined by a finite number of operations if a given polynomial has an integral root."
- Hilbert assumed that such an algorithm exists.
- In 1970, Yuri Matijasevič, building on the work of Martin Davis, Hilary Putnam, and Julia Robinson showed that <u>No</u> algorithm exists for this task.
- For single variate polynomials, an algorithm can be given.

Describing Turing Machines:-

1) Formal description.

2) Implementation description: The way Turing machine head moves and the way it stores data on its tape etc., Ignore details of states or transition function.

3) High-level description: Describe the algorithm, ignoring implementation details. (Default unless otherwise specified).

- For an object O, its string representation $\langle O \rangle$.
- For multiple objects $O_1, O_2, \ldots, O_k$, single string $\langle O_1, O_2, \ldots, O_k \rangle$.
- Any encoding is fine
- If the input is $\langle O \rangle$, the Turing machine first implicitly tests whether the input properly encodes an object of the desired form and rejects otherwise. This step need not be written in the algorithm explicitly.

Example. $A = \{ \langle G \rangle \mid G \text{ is a connected undirected graph} \}$

M = " On input $\langle G \rangle$, the encoding of a graph G:

1. Select the first node of G and mark it.
2. Repeat the following stage until no new nodes are marked:
   a. For each node in G, mark it if it is attached by an edge to a node that is already marked.
3. Scan all the nodes of G to check if they are all marked. If they are accept, otherwise reject."