CSC 320 - Lecture 12

#turing-machines #TM #multitape #nondeterministic #decider #deterministic

---

Turing Machines - Deterministic vs Non-Deterministic

Deterministic Function. Always returns same result for same input values. Transition function, for a well-defined input, is uniquely defined.

Clearly defined what happens next!

Nondeterministic Function. May return different results for different calls for same input values. Transition function returns a set of possible outcomes.

Turing Machine

We have seen one tape, left-ended, and unlimited to the right. And read/write head.

Variant of the Turing Machine

A **TM with a read/write/stop head** is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ where...

- Q, Σ, Γ are finite sets
  - Q set of states
  - Σ input alphabet not containing blank symbols $\sqcup$
  - Γ tape alphabet where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
- $\delta = Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$ transition function
- $q_0 \in Q$ start state
- $q_{accept} \in Q$ accept state
- q_{reject} reject state with $q_{reject} \neq q_{accept}$

Note. Read/Write head not forced to move left (L) or right (R) but can stay (S) at current position.

Question. Do I give my TM more power with S? (Recognize more languages)? No.

Equivalence of Both TM Models

Definitions of TM with a read/write/stop head and TM equivalent: Transition that does not move head can be simulated by two transitions.

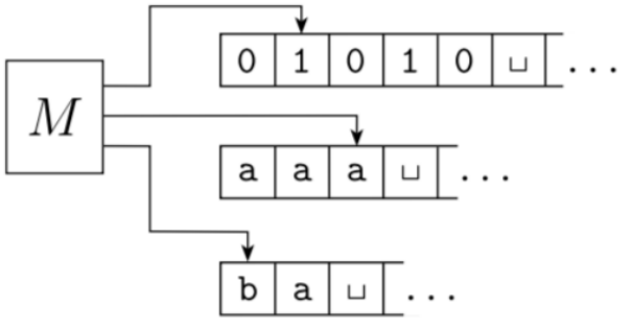
- One moving left, following by one moving right.

Variants of Turing Machines

Single-Tape Turing Machines (Deterministic), Multitape Turing Machines (Deterministic), Nondeterministic Turing Machines, Enumerators.

Multitape Turing Machines

Like original TM but has multiple tapes each with read/write head. And initial configuration input w on tape 1, all other tapes blank.



The transition function of a **Multitape Turing Machine** M with k **tapes** is defined as follows...

- $\delta = Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$

$\delta(q, a_1, \ldots, q_k) = (p, b_1, \ldots, b_k, L, R, \ldots, L)$ means...

- If M is in state q and heads 1 through k point at cells with content $a_1, \ldots, a_k$ then...
  - M changes to state p
  - replaces $a_1, \ldots, a_k$ with $b_1, \ldots, b_k$, and
  - move left, right, or don't move (as specified by $L, R, \ldots, L$)

Note. Finite number of k.

Multitape Turing Machine – Formal Definition

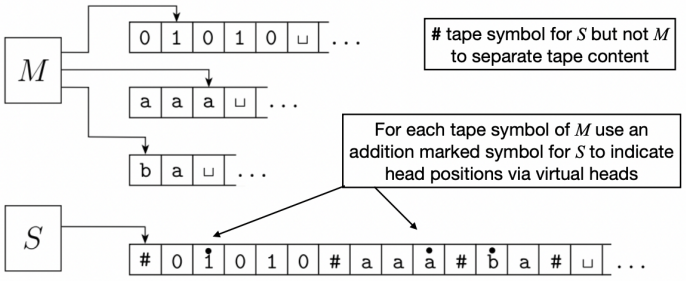
A **Multitape Turing Machine** with k tapes is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ where...

- Q, Σ, Γ are finite sets
  - Q set of states
  - Σ input alphabet not containing blank symbol $\sqcup$
  - Γ tape alphabet where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
- $\delta = Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$
- $q_0 \in Q$ start state
- $q_{accept} \in Q$ accept state
- $q_{reject} \in Q$ reject state with $q_{reject} \neq q_{accept}$

Theorem

For every Multitape Turing Machine there is an equivalent Single-Tape Turing Machine.

Proof. Let M be a Multitape Turing Machine with k tapes. We convert M into a Single-Tape Turing Machine S.



On input $w = w_1 w_2 \dots w_n$

- Prepare the tape of S to represent all k tapes of M: $\#\dot{w}_1 w_2 \dots w_n \# \dot{\sqcup} \# \dot{\sqcup} \# \dots \#$
- Simulating a move of M
  - Determine symbols under virtual heads via scan from left to right
  - Execute transition via second scan

More formally let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ be Multitape TM with $k4 tapes.

Convert M into Single-Tape Turing Machine $S = (Q, \Sigma, \Gamma', \delta', q_0, q_{accept}, q_{reject})$ with...

- $\Gamma' = \Gamma \cup \{\#\} \cup \{\dot{\#}\} \cup \{\dot{a} | a \in \Gamma, \dot{a} \notin \Gamma\}, \#, \dot{\#} \notin \Gamma$
- and transition function δ' for S simulates each move of M
  - for each step of M: S moves from virtual tape to virtual tape
    - **First Scan**. Search for all current head positions and determine content of current cells on all virtual tapes.
    - **Second Scan**. Update content of current cells and update head positions, according to transition that M is executing.

- If a virtual tape head moves right encountering $\#$, make room on virtual tape: all tape content starting at $\#$ is shifted by 1 cell to the right and adds a blank for the $\#$.

Corollary

A language L is Turing-Recognizable if and only some (Single-Tape) TM recognizes it if and only if some Multiple TM recognizes it.

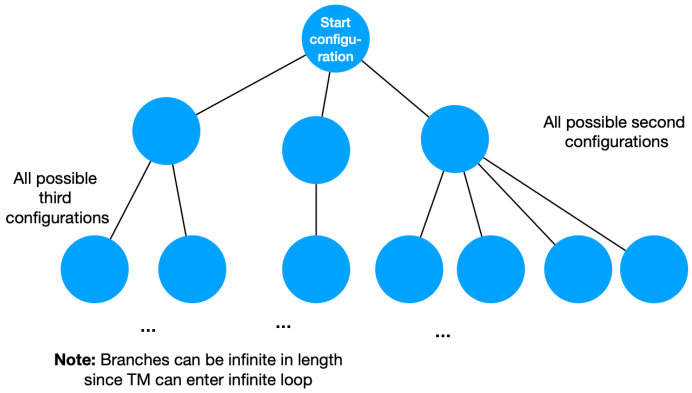
Nondeterministic TMS

All TMS so far have been deterministic TMs. Their computation is deterministic due to definition of transition function.

A **Nondeterministic TM** M is defined just like the (Single-Tape) TM, but its transition function is defined as follows...

- $\delta = Q \times \Gamma \to P(Q \times \Gamma \times \{L, R\})$

Computation of A Nondeterministic TM N



TM do not have to stop. There can be many reject paths, but only one correct path. We have to check all of the paths. We can't just stop at the first reject. (We can only do that for Deterministic TM's).

Theorem

For every nondeterministic TM there exists an equivalent deterministic TM.

That Is. For each deterministic TM there exists a non-deterministic one that recognizes the same language, and vice versa (for each nondeterministic TM there exists a deterministic one that recognizes the same language).

Note. Since we only have 1 correct path for a deterministic TM, it can be viewed as a special case of a non-deterministic TM.

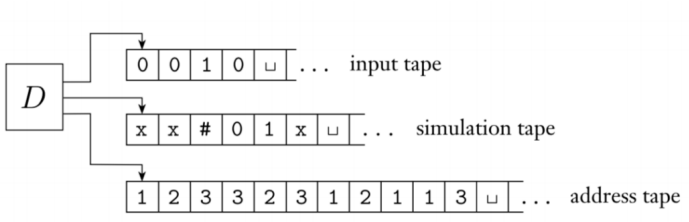
Every Nondeterministic TM Has Equivalent (Deterministic) TM

Proof. Simulate a nondeterministic TM N with (deterministic) Multitape TM D.

Idea. D executes computation of every branch in computation tree of N in BFS-Manner, unless it enters its accept state.

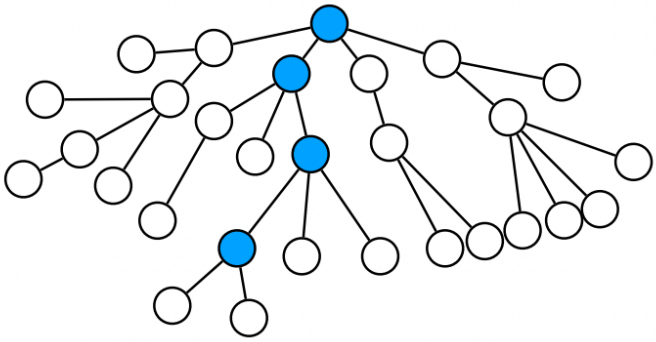
D: Multitape TM with 3 tapes

- **Tape 1**. Contains input string; never changes
- **Tape 2**. Maintains copy of N's tape on current simulated branch of its (nondeterministic) computation.
- **Tape 3**. Keeps track of D's location in N's computation tree.



Tape 3. Encoding location in computation tree via encoding choices. Every node in N's computation tree has at most b children; b: size of largest set of possible choices given by N's transition function.

- To every node in the tree assign address (string over $\Gamma_b = \{1, 2, \ldots, b\}$):
  - **Example**. Address 231 corresponds to node in computation tree when...
    - starting at root r
    - going to its 2^{nd} child c_2
    - going to c_2's 3^{rd} child g_3
    - going to g_3's 1^{st} child
  - Address of root: ε



1. **Initially**. Tape 1 contains input w; tapes 2 and 3 are empty
2. Copy tape 1 to tape 2 and set string on tape 3 to be ε (i.e., location of computation tree simulation is root)
3. On tape 2 D simulates N for w on a branch of its (nondeterministic) computation:
    - Before each step of N, look up the next symbol on tape 3 to determine N's next choice according to possibilities given by N's transition function
    - If no more symbols remain on tape 3, or if this nondeterministic choice is invalid, abort this branch and go to step 4
    - If N enters a reject state go to step 4
    - If N enters an accepting configuration accept w
4. Replace string on tape 3 with the next string in BFS-Ordering and go to step 2

BFS ordering of addresses avoids being trapped in infinite branch

$\epsilon, 1, 2, 3, 4, 11, 12, 13, 14, 21, 22, 23, \ldots, 231, \ldots$

Note. BFS ordering addresses avoids being trapped in infinite branch (i.e., DFA can result in getting trapped in an infinite branch).

Corollary

A language L is Turing-Recognizable if and only some (Single-Tape) TM recognizes it if and only if some Multiple TM recognizes it if and only if some Nondeterministic TM recognizes it.

Definition

A nondeterministic TM is called a (nondeterministic) **decider** if all branches halt, on all inputs.

Theorem

A language is decidable if and only if some nondeterministic TM decides it.

Proof Idea. Simulate nondeterministic decider N with deterministic TM D. Since N always halts on all branches of its computation, also D will always halt.

Previous Lecture

Lecture11

Next Lecture

Lecture13