Computational Complexity and Computability

Lecture 5 - NTMs & Enumerators

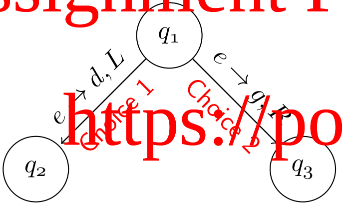
Koushik Pal

University of Toronto

January 25, 2021

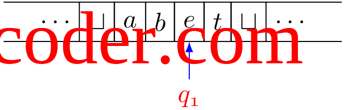
Nondeterministic TM

Transition function: $\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R\})$
($\mathcal{P}(S)$ denotes the power set of S)



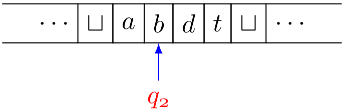
Nondeterministic Choice

Time 5

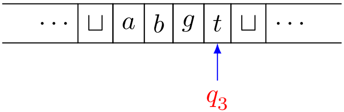


Time 6

Choice 1



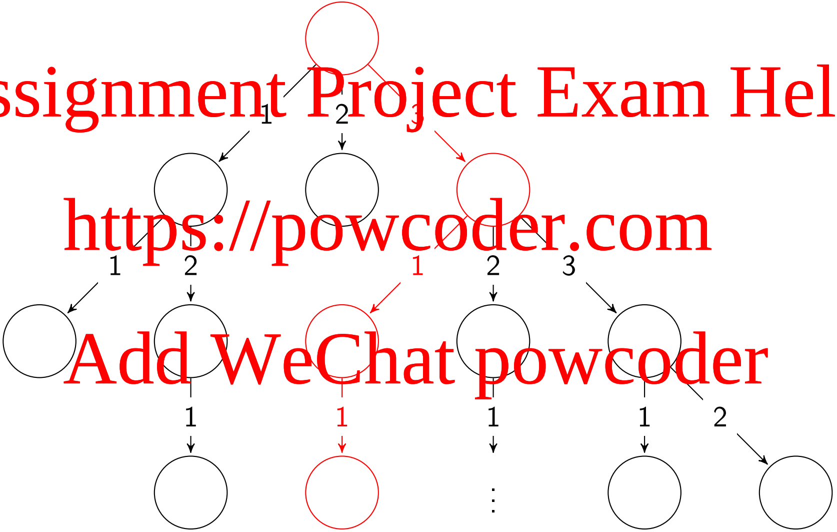
Choice 2



Nondeterministic TM

- Computation of a nondeterministic TM on an input can be viewed as a tree.
- Each node of the tree represents a configuration.
- The root node represents the initial configuration.
- Each branch corresponds to a different possibility for the machine.
- If some branch of the computation leads to the accept state, the machine accepts its input.

Nondeterministic TM



3^{11}

Nondeterministic TM

Theorem
The class of nondeterministic TMs has the same power as the class of standard TMs.

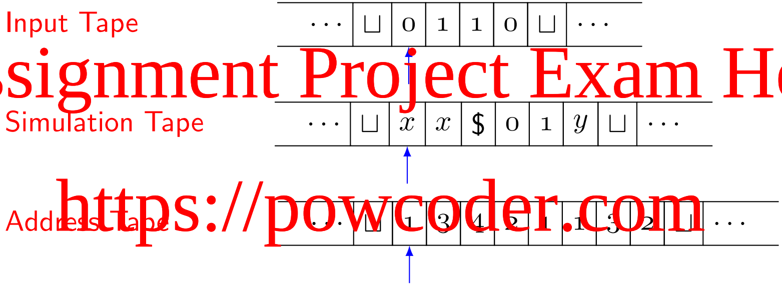
Proof.
A standard (deterministic) TM is trivially a nondeterministic TM that has exactly one branch at each node.

To simulate a nondeterministic TM N on a standard TM, we need to evaluate each branch of the computation tree and accept the input when one of the branches accepts the input.

But we need to be careful with the order in which we choose the branches to evaluate. DFS does not work! Use BFS!

Nondeterministic TM

Define a 3-tape deterministic TM D as follows:



- Input tape contains only the input and is never altered.
- Simulation tape maintains a copy of N's tape on some branch of its nondeterministic computation.
- Address tape keeps track of D's location in N's computation tree by generating strings in the lexicographic ordering from the language Σ^+, where $\Sigma = \{1, 2, \ldots, b\}$ with b being the maximum number of choices at any given node.

Nondeterministic TM

Exercise. Design a TM that generates strings in the lexicographic ordering from the language Σ^+, where $\Sigma = \{1, 2, \ldots, b\}$.

Example: Lexicographic ordering on $\Sigma = \{1, 2, 3\}$ is as follows:

$\{1, 2, 3, 11, 12, 13, 21, 22, 23, 31, 32, 33, 111, 112, 113, 121, 122, 123, \ldots\}$

Exercise. Convince yourself that lexicographically ordered strings from the language Σ^+ is equivalent to a BFS search on N's nondeterministic computation tree.

Nondeterministic TM

Actual description of D:

1. Initialize the input tape with the input w, the address tape with the empty string ε (aka, root configuration), and leave the simulation tape empty.
2. Copy the input w from the input tape to the simulation tape.
3. Simulate N with input w on one branch of its computation on the simulation tape. Before each step of N, consult the next symbol on the address tape to determine which choice to make among those allowed by N's transition function.
4. If no more symbols remain on the address tape or this choice is invalid or a rejecting configuration is encountered, abort this branch by going to stage 5. If an accepting configuration is encountered, accept the input and halt.
5. Replace the string on the address tape with the next string in the lexicographic ordering. Simulate the next branch of N's computation by erasing the simulation tape and going to stage 2. □

Turing-recognizable and Turing-decidable

Definition

A language is called Turing-recognizable / semi-decidable / recursively enumerable if some TM **recognizes** it.

Three outcomes are possible when a TM runs on an input:

1. Accept
2. Reject
3. Loop (aka, the machine doesn't halt).

Definition

A TM is called a decider if it halts on all inputs $x \in \Sigma^*$.

A language is called Turing-decidable / decidable / recursive if some TM **decides** it.

Examples

Example (Decidable languages)

1. $PAL = \{yy^{reverse} \mid y \in \{0,1\}^*\}$
2. $\mathcal{L} = \{a^n b^n c^n \mid n \in \mathbb{N}\}$
3. $\mathcal{L} = \{a^i b^j c^k \mid i + j = k\}$
4. Any finite language $\mathcal{L}$

Notation

- $D = \{A \subseteq \Sigma^* \mid A$ is decidable$\}$
- $SD = \{A \subseteq \Sigma^* \mid A$ is semi-decidable$\}$
- $P = \{A \subseteq \Sigma^* \mid A = \mathcal{L}(M)$ for some M which halts in polynomial time$\}$.

Theorem

$P \subsetneq D \subsetneq SD$.

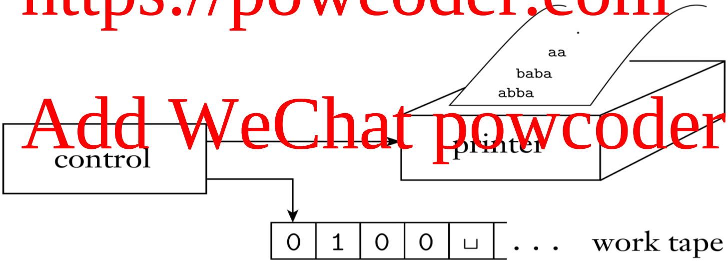
Enumerator

Definition

An enumerator E is a 2-tape TM with an input tape and an output tape and a transition function

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\} \times \{\text{print, don't print}\}.$$

E has an initial state, but no accept/reject states - it runs forever.



Note: The language $\mathcal{L}(E)$ enumerated by E is the set of strings it prints. E may print strings in any order and multiple times.

Semi-decidable iff enumerated by enumerator

Theorem

A language $A \subseteq \Sigma^*$ is semi-decidable iff some enumerator enumerates it.

Proof.

$(\longleftarrow)$

Suppose E enumerates A. We define a TM M as follows:

On input w:

- *Run E.*
- *Every time E computes a string, compare it with w; if they match, halt and accept.*

Clearly M recognizes A, and hence A is semi-decidable.

Semi-decidable iff enumerated by enumerator

($\longrightarrow$)

Conversely, suppose A is semi-decidable. Then $A = \mathcal{L}(M)$ for some TM M.

Let $s_1, s_2, \ldots \in \Sigma^*$ be all possible strings in lexicographic order.

Define E as follows:

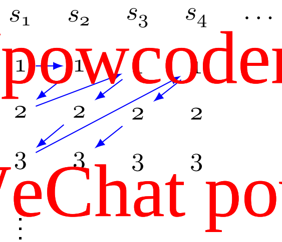
Repeat the following for $i = 1, 2, \ldots$:

- *Generate string s_i*
- *Append it to the existing strings with a delimiter*
- *Execute M's next move on the inputs $s_1, \ldots, s_i$*
- *If any of the computations result in an accept state, print out the corresponding s_j.*

Semi-decidable iff enumerated by enumerator

M executes

- first step on s_1; then
- first step on s_2, followed by second step on s_1; then
- first step on s_3, followed by second step on s_2, followed by third step on s_1; and so on . . .



Clearly, if M accepts s, then s eventually appears in $\mathcal{L}(E)$. □

Remark.

This technique is called DOVETAILING. It gives the effect of running M in parallel on all possible input strings.

Decidable iff enumerated by enumerator lexicographically

Theorem

A language $A \subseteq \Sigma^*$ is decidable iff some enumerator enumerates it in lexicographic order.

Proof.

Exercise. $\square$