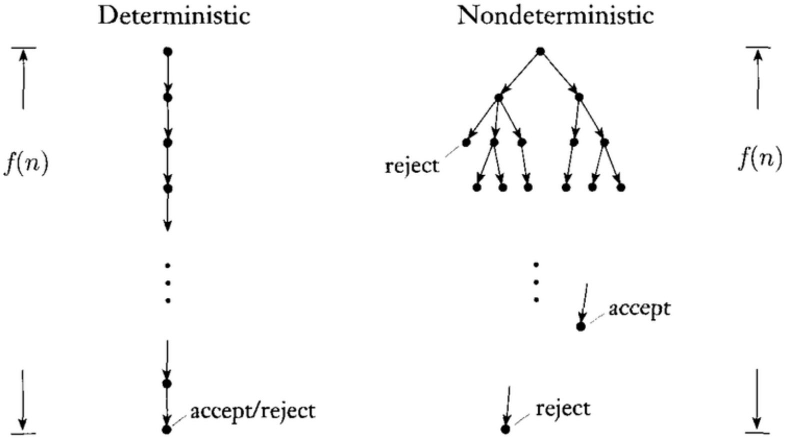
Nondeterministic Turing machines

Textbook: Chapter 3.2



Idea

- We turned DFA into NFA by modifying the form of δ
- **What if we do this to a TM?**

Recall: A TM's transition function is of the form:

$$\delta : (Q \times \Gamma) \rightarrow (Q \times \Gamma \times \{L, R\})$$

For state set Q, tape alphabet Γ. A nondeterministic version would look like:

$$\delta : (Q \times \Gamma) \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\})$$

Definition of NTMs

Def: Nondeterministic Turing machines (NTMs). An NTM is a 6-tuple

$$(Q, \Sigma, \Gamma, \delta, q_0, q_{\mathtt{accept}})$$

where Q, Σ, Γ are all finite sets and:

1. Q is the state set
2. Σ is the input alphabet, $\sqcup \notin \Sigma$
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
4. δ is the nondeterministic transition function

$$\delta : (Q \times \Gamma) \to \mathcal{P}(Q \times \Gamma \times \{L, R\})$$

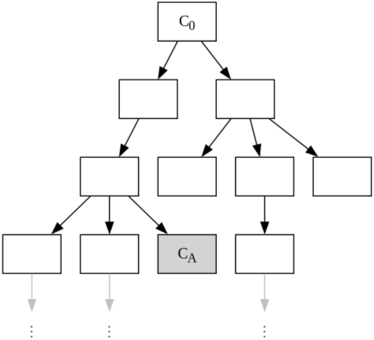
5. $q_0 \in Q$ is the start state
6. $q_{\mathtt{accept}} \in Q$ is the accept state

An NTM accepts iff any branch of its nondeterminism does

Equivalence with (D)TM

Thm: TM and NTM are computationally equivalent.

Pf: By construction. If we let an NTM configuration be a node in a tree from which zero or more may follow:



We want to find any C_A in the tree following C_0. How?
Breadth-First search!

Notes

- Note: We cannot use depth-first search, since any given branch may never terminate (DFS doesn't work on infinite trees)
- Note: This algorithm will always find the shortest accepting computation history if one exists

Convince yourself that queues are legal data structures to use within a TM (recall TMs and algorithms are equivalent)

Breadth-first search on NTM configuration trees

Let TM R take input $\langle M \rangle, w$, where w is the input to NTM M:

1. Create an empty queue of configurations
2. Push the entrance configuration $q_0 w$ to the queue
3. For as long as the queue is nonempty:
   3.1 Pop a configuration c from the queue
   3.2 If c contains a halting state, halt
   3.3 Push each configuration that follows from c to the queue

Note that R accepts iff M does on some branch, R rejects iff M does on some branch, and R loops iff M does on all branches. R is a legal TM emulating an arbitrary NTM, so TM and NTM are equivalent. End of proof.

Complexity

Thm: A TM can simulate i steps of an NTM in $O(2^i)$ time and space.

- If we let b be the max number of configurations output by δ, then there are at most b^i nodes after i steps
- Therefore, the TM will have to simulate a total of up to $\sum_{j=0}^{i} b^j = b^{i+1} - 1$ nodes! **Very bad!**

Computational equivalence does not imply polynomiality!

Corollaries

Corollary: A system is Turing-recognizable iff some NTM recognizes it, Turing-decidable iff some NTM decides it, Turing-complete iff it simulates all NTM, and Turing-equivalent iff it is equivalent to an NTM.

- A language is decided by an NTM iff it halts on all branches of its determinism

Next up: Advanced computability theory