EECS 2001N : Introduction to the Theory of Computation

Suprakash Datta
Office: LAS 3043

Course page: http://www.eecs.yorku.ca/course/2001N
Also on Moodle

Turing Machines - Decidability

- A language $L = L(M)$ is **decided** by the TM M if on *every* input w, the TM finishes in a halting configuration.
  That is: q_{accept} for $w \in L$ and q_{reject} for all $w \notin L$.
- A language L is Turing-decidable if and only if there is a TM M that decides L
- Also called: a *recursive* language

Turing Machines - Recognizability

- A language $L = L(M)$ is **recognized** by the TM M if on *every* input $w \in L$, the TM finishes in the halting configuration q_{accept}
- On an input $w \notin L$, the machine M can halt in the rejecting state q_{reject}, or it can 'loop' indefinitely
- A language L is Turing-recognizable if and only if there is a TM M such that $L = L(M)$
  Recall: The language that consists of all inputs that are **accepted** by a TM M is denoted by $L(M)$
- Also called: a *recursively enumerable* language

Turing Machines - Variants

- Multiple tapes
- 2-way infinite tapes
- Non-deterministic TM's

Multi-tape Turing Machines (Ch 4.3)

Theorem 4.3.1: Let $k \geq 1$ be an integer. Any k-tape Turing machine can be converted to an equivalent one-tape Turing machine.

- Proving and understanding these kinds of robustness results is essential for appreciating the power of the Turing Machine model

- From this theorem it follows that:
  A language L is TM-recognizable if and only if some multi-tape TM recognizes L.

Proof of Theorem 4.3.1

- Take a 2-tape TM M and construct an equivalent one-tape TM N
  "N can **simulate** M"

- Tape alphabet of N: $\Gamma \cup \{\dot{x} | x \in \Gamma\} \cup \{\#\}$

- Idea: the contents of the two tapes will be maintained on one tape separated by $\#$ and the dotted version of a character will be used to indicate the location of the head

Proof of Theorem 4.3.1 - contd.

N simulates the computation of M in each step

- At the start of the step, the tape head of N is on the leftmost symbol $\#$
- N "remembers" the state of M in its state
- In each step, N moves right until it has read both dotted symbols
- The second and then the first dotted symbol is changed as M would change them
- In either case above the contents of the tape may have to be shifted
- Finally, N remembers the new state of M and moves to the leftmost symbol $\#$

2-way Infinite Tape Turing Machines

- For every 2-way infinite tape TM M, there is a 2-tape TM M' such that $L(M) = L(M)$

- Suppose the cells are numbered 0,1,2,.... and -1,-2,....

- Idea: Store the contents of cell 0 and everything to its right on the first tape of M' and everything to the left of cell 0 on the second tape, and simulate the computation of M as usual

Non-deterministic Turing Machines

A Non-deterministic one-tape Turing Machine M is defined by a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$:

- finite set of states Q
- finite input alphabet Σ
- finite **tape** alphabet Γ
- start state $q_0 \in Q$
- accept state $q_{accept} \in Q$
- reject state $q_{reject} \in Q$
- transition function $\boxed{\delta : Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R, N\})}$

Non-deterministic Turing Machines - 2

- Just like multi-tape TM's, nondeterministic TM's are not more powerful than simple TMs
- Every nondeterministic TM has an equivalent 3-tape TM, which in turn has an equivalent 1-tape TM
- Hence: "A language L is recognizable if and only if some nondeterministic TM recognizes it."
- The Turing machine model is extremely robust!

Non-deterministic Turing Machines - 3

- A non-deterministic TM's computation may be thought of as a tree of configurations rather than a path
- If there is (at least) one accepting leaf in this tree, then the TM accepts
- We have to traverse this tree using a deterministic TM
- Bad idea: "depth first" exploration. The TM may explore never-halting paths
- Good idea: "breadth first" exploration. For time steps 1,2,..., we list all possible configurations of the non-deterministic TM. The simulating TM accepts when it reaches an accepting configuration

Non-deterministic Turing Machines - 4

- Let M be the non-deterministic TM on input w
- The simulating TM uses three tapes:
  T_1 contains the input w
  T_2 the tape content of M on w at a node
  T_3 describes a node in the tree of M on w
- Inititally, T_1 contains w, T_2 and T_3 are empty
- Simulate M on w via the deterministic path to the node of tape 3.
  If the node accepts, "accept"
- Increase the node value on T_3, go to previous step

The Church Turing Thesis

- The Church-Turing thesis marks the end of a long sequence of developments that concern the notions of "way-of-calculating", "procedure", "solving", "algorithm"

- Theorem 4.4.1 The following computation models are equivalent, i.e., any one of them can be converted to any other one:
  1. One-tape Turing machines
  2. k-tape Turing machines, for any $k \geq 1$
  3. Non-deterministic Turing machines
  4. Java programs
  5. C++ programs
  6. Python programs