The lecture introduced Turing Machines, a general model of computation that mimics how a person might compute with pen and paper. We think of a TM as a low-level programming language that has some finite internal memory and just a single data structure, a tape. We used the simulator http://morphett.info/turing/turing.html to visualise TM computations. The language of a TM is called "Turing-recognisable" or just "recognisable". A TM may, on a given input, run forever (something that DFAs do not do), and in this case it does not accept its input. If a TM halts on all inputs, it is called a "decider". A language is "Turing-decidable" or just "decidable" if it is recognised by a decider. Deciders naturally model the sorts of algorithms we typically write, i.e., ones that always terminate! But sometimes we write algorithms that do not halt on every input, and these algorithms are captured by TMs that need not terminate on each input.

After this tutorial you should be able to:

1. Design/program Turing Machines to do certain tasks
2. Show how to simulate TMs with certain properties by TMs with other properties (typically to show that the class of TMs is robust).

---

1 Turing Machines

Problem 1. Prove that every regular language is decidable.

Problem 2. Let $\Sigma = \{0, 1, \#\}$. Build a decider for the set of strings $u\#v$ for $u, v \in \{0,1\}^*$ such that $|u| < |v|$. For instance 111#0000 should be accepted and 101#011 should not.

Problem 3. Devise a TM that recognises the language $\{w\#w \mid w \in \{0,1\}^\star\}$. You might find it easier to write an implementation-level description first, and then write the low-level description (i.e., the full transition table).

Additional problems

Problem 4. Let $\Sigma = \{1, \#\}$. For each of the following languages, build a decider for it.

1. The set of strings of the form $u\#v$ for $u, v \in \{1\}^*$ where $|u|$ divides $|v|$.
   - For instance, 11#1111 should be accepted, but 11#111 should not.
2. The set of strings of the form $u\#v\#w$ for $u, v, w \in \{1\}^*$ such that $|u|$ multiplied by $|v|$ equals $|w|$.
   - For instance 11#11#1111 should be accepted, but 11#111#11 should not.

Problem 5. Let $\Sigma = \{0, 1, \#\}$. Build a decider for the set of strings $u\#v$ for $u, v \in \{0,1\}^*$ such that u is length-lexicographically smaller than v. This means

that either (i) $|u| < |v|$, or (ii) $|u| = |v|$, $u \neq v$, and if i is the left-most position where u differs from v then $u_i = 0$ and $v_i = 1$. For instance, 111#0000 and 0010#0101 should be accepted, but 100#1 and 010#001 should not.

Problem 6. In this problem you will see that TMs can also compute functions (and not just decide things).

Devise a TM M that, given an input string of the form 0^n10^m reaches an accepting configuration with 0^{n+m} written on the tape.

1. Give an implementation description of M.
2. Give the formal description of M, i.e., states, transitions, etc.
3. Give the computation, i.e., sequence of configuration, of M on input 00001000.