CS411 Theory of Computation
Lecture 4

Sergey Kitaev

University of Strathclyde

27 September 2017

Formal description of the Turing Machine M_1

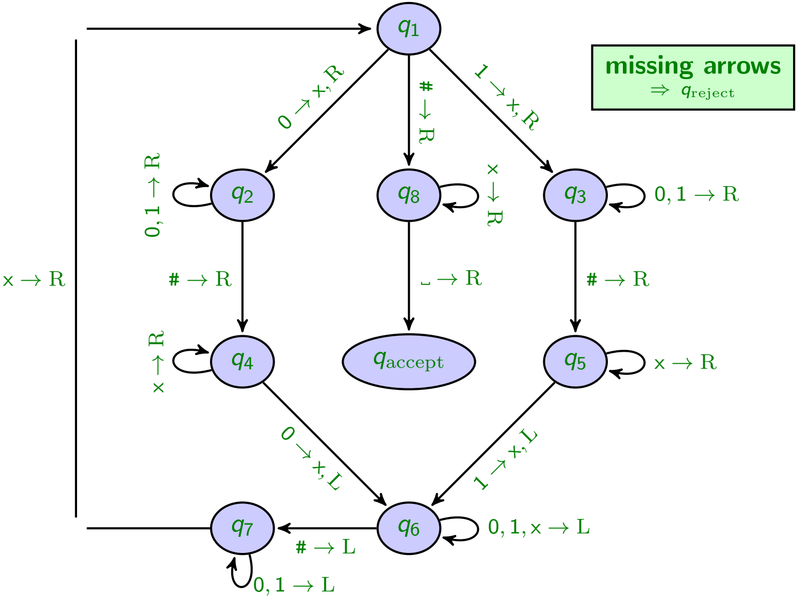
Recall that M_1 is the Turing machine for deciding the language

$$B = \{w\#w \mid w \in \{0,1\}^*\}.$$

Here $M_1 = (Q, \Sigma, \Gamma, \delta, q_1, q_{\mathrm{accept}}, q_{\mathrm{reject}})$ where

- $Q = \{q_1, \ldots, q_8, q_{\mathrm{accept}}, q_{\mathrm{reject}}\}$.
- $\Sigma = \{0, 1, \#\}$ and $\Gamma = \{0, 1, \#, \mathrm{x}, \sqcup\}$.
- δ is given as a state diagram on the next slide.
  Note that missing arrows imply going to q_{reject}.
- The start, accept and reject states are q_1, q_{accept} and q_{reject}, resp.

Transition function δ for M_1



Some comments concerning M_1:

- In the state diagram of M_1, the label on the transition from q_3 to itself is $0, 1 \to \mathrm{R}$. This means that if the head is in state q_3 and currently reading 0 or 1, then it leaves that value unchanged, moves to the right, and remains in state q_3.
- Stage 1 is implemented by states q_1 through q_7, and Stage 2 by the remaining states.
- The reject states aren't shown in the state diagram **but** a lack of an edge/arrow implies transition to the reject state. For example, since there is no outgoing arrow from state q_5 with a #, then we understand $\delta(q_5, \#) = (q_{\mathrm{reject}}, \#, R)$.

Running M_1 on different inputs

1. 0#0
2. 01#0
3. 1#10
4. 00#00

The Turing Machine M_3

Let M_3 be the Turing Machine that decides the language

$$C = \{\mathrm{a}^i \mathrm{b}^j \mathrm{c}^k \mid i \times j = k \text{ and } i, j, k \geq 1\}.$$

This is set of all strings of a's followed by b's which are followed by c's such that the number of a's times the number of b's equals the number of c's.

For example, aaabbcccccc $\in C$, but abbc $\notin C$.

If we have a string of a's followed by a string of b's, which is in turn followed by a string of c's, and there is at least one of each symbol, then we say that the string is a member of $\mathrm{a}^+\mathrm{b}^+\mathrm{c}^+$.

Equivalently, this set can be written $\mathrm{aa}^*\mathrm{bb}^*\mathrm{cc}^*$.

The Turing Machine M_3

$$C = \{\mathrm{a}^i\mathrm{b}^j\mathrm{c}^k \mid i \times j = k \text{ and } i, j, k \geq 1\}.$$

$M_3 =$ " On input string w:

1. Scan the input from left to right to determine whether it is a member of $\mathrm{a}^+\mathrm{b}^+\mathrm{c}^+$ and reject if not.
2. Return the head to the left-hand end of the tape.
3. Cross off an a and scan to the right until a b occurs. Shuttle between the b's and c's, crossing off one of each until all b's are gone. If all c's have been crossed off and some b's remain, reject.
4. Restore the crossed off b's and repeat stage 3 if there is another a to cross off. If all a's have been crossed off, determine whether all the c's have been crossed off. If Yes, accept. Otherwise, reject."

The Turing Machine M_4

This Turing Machine will solve the element distinctness problem.

It is given a list of strings over $\{0,1\}$ separated by #'s and its job is to accept if all the strings are different.

The language is

$$E = \{\#x_1\#x_2\#\cdots\#x_\ell \mid \text{each } x_i \in \{0,1\}^* \text{ and } x_i \neq x_j \text{ for each } i \neq j\}.$$

The machine works by comparing x_1 with x_2 through x_ℓ, then by comparing x_2 with x_3 through x_ℓ, and so on.

Informal description of M_4

M_4 = "On input w:

1. Place a mark on top of the leftmost tape symbol. If that symbol was ␣ then accept. If it was # then continue to next stage. Otherwise reject.
2. Scan right to the next # and place a second mark on top of it. If no # is encountered before ␣ only x_1 was present so accept.
3. By zig-zagging, compare the two strings to the right of the marked #'s. If they are equal reject.
4. Move the rightmost of the two marks to the next # symbol to the right. If no # symbol is encountered before a ␣ then move the leftmost mark to the next # to its right and the rightmost mark to the # after that. This time, if no # is available for the rightmost mark, all the strings have been compared, so accept.
5. Go to stage 3.

Some notes on M_4

- M_4 illustrates the technique of marking tape symbols, i.e. $\#$ becomes $\mathring{\#}$.
- In actuality, marking $\#$ means adding $\mathring{\#}$ to its tape alphabet, and then marking $\#$ corresponds to writing $\mathring{\#}$.
  Unmarking $\mathring{\#}$ corresponds to writing $\#$.
- We may want to mark all symbols on a tape, and in this case we simply include marked versions of all symbols in the tape alphabet.
- The four examples considered show that languages A, B, C and E are decidable.
- All decidable languages are Turing recognizable, so these languages are Turing-recognizable.

Variants of TMs and Robustness

- There are many alternative definitions of Turing Machines. These variants include versions with multiple tapes and 1-head, or those with non-determinism.
- The original model and its (reasonable) variants all have the same power in that they recognize the same class of languages.
- We now describe some of these variants and the proofs that they are equivalent.
- This invariance to certain changes in the definition is called robustness.

The ordinary Turing Machine that can Stay put

Example (Staying-put)

Consider a Turing Machine that is equipped with the ability for the head to stay put (S). The transition function of such a variant will have the form

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}.$$

Question: Will this new feature allow the machine to recognize additional languages?

Answer: No! We could model this with a Turing machine that moves to the left and then back to the right in two steps.

Important point: This example contains the prototype of proofs of equivalence of Turing machine variants, i.e. that we can simulate one by another.