CS411 Theory of Computation
Lecture 2

Sergey Kitaev

University of Strathclyde

20 September 2017

Turing Machines

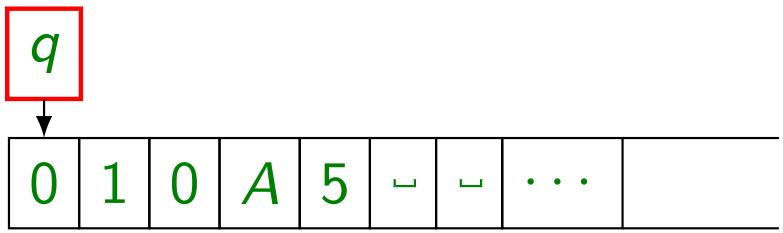
A Turing Machine (TM) is a model of computation that was proposed by Alan Turing in 1936.

It can do everything that a real (conventional) computer can do.

However, a Turing Machine cannot solve all problems, and these unsolvable problems are beyond the theoretical limits of computation.

How does it work ?

- The Turing Machine uses an infinite tape as its unlimited memory.
- It has a tape head that can read and write symbols and move around on the tape.
- Initially the tape contains only an input string and is otherwise blank (␣).



Turing Machines

If the machine needs to store information then it writes this information on the tape.

To read the information it has written, the machine can move its head back over it. The machine continues computing until it decides to produce an output. The outputs accept and reject are obtained by entering designated accepting and rejecting states.

If it doesn't enter an accepting or a rejecting state, then it will go on forever, never halting.

- A TM can both write on the tape and read from it.
- The read-write head can move both to the left and to the right.
- The tape is (right) infinite.
- The special states for rejecting and accepting take effect immediately.

The Turing Machine M_1

This first Turing machine is for testing membership of the language:

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

This is the language of all binary strings where the first part is the same as the second part, but is separated by the # symbol.

e.g. $010\#010 \in B$ but $010\#01 \notin B$.

Testing membership is trivial (for a human) for short sequences, but what about long sequences?

Strategy: Traverse the sequence in some way so that corresponding positions are compared and mark those which have corresponding values. (This can be done in a multitude of ways.)

The Turing Machine M_1

Let's try to describe this behaviour in more detail:

- The head will zig zag over the tape which contains the input string to be tested for membership in B.
- On each pass it will match one of the characters on each side of the $\#$ symbol.
- To keep track of which characters have been checked so far, M_1 will cross off each symbol as it is examined.
- If all symbols are crossed off then both sides match and M_1 goes into an accept state.
- If it discovers a mismatch then it enters a reject state.

The Turing Machine M_1

More succinctly:

- Zig-zag across tape to corresponding positions on either side of # symbol to check whether these positions contain same symbol.
  If they do not, or if no # found, then reject.
  Cross off symbols as they are checked to keep track of which symbols correspond.

- When all symbols to the left of # have been crossed off, check for any remaining symbols to the right of #.
  If any symbols remain, then reject, otherwise accept.

For example, consider the following inputs on the tape:

(Input 1) 0 1 1 0 0 # 0 1 1 0 0

(Input 2) 1 0 1 0 1 # 1 0 0 0

The Turing Machine M_1

- The previous description of the Turing Machine M_1 is only a sketch.
- We can describe TMs in complete detail by giving formal descriptions of the operations in terms of states and head location values.
- These formal descriptions specify each of the parts of the formal definition of the TM model (introduced next).
- In actuality we almost never give formal descriptions of TMs because they tend to be very big.

Turing Machine – Formal Definition

- At any given moment in time the Turing machine will be in one of several possible states. The set of possibly states is usually written as Q.
- The input to the machine is written at the beginning of the tape and its symbols come from an 'input' alphabet Σ that does not contain the blank symbol $\sqcup$
- The set of symbols that may appear on the infinite tape is Γ, and this must include all symbols from Σ along with the blank symbol $\sqcup$
- The heart of the definition of a TM is the transition function δ:
- The transition function tells us what to do next based on both the current system 'state' q and the current symbol a beneath the tape-head.
  If $\delta(q, a) = (r, b, L)$ then this means write the symbol b to replace a and change the system state to r, and move the head to the left (L).

Turing Machine – Formal Definition

Definition

A Turing Machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ where Q, Σ, Γ are all finite sets and

(1) Q is the set of states

(2) Σ is the input alphabet not containing the blank symbol $\sqcup$

(3) Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$

(4) $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function

(5) $q_0 \in Q$ is the start state

(6) $q_{\text{accept}} \in Q$ is the accept state

(7) $q_{\text{reject}} \in Q$ is the reject state where $q_{\text{reject}} \neq q_{\text{accept}}$.

Turing Machine – Dynamics

Supposing we have a Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, what exactly happens?

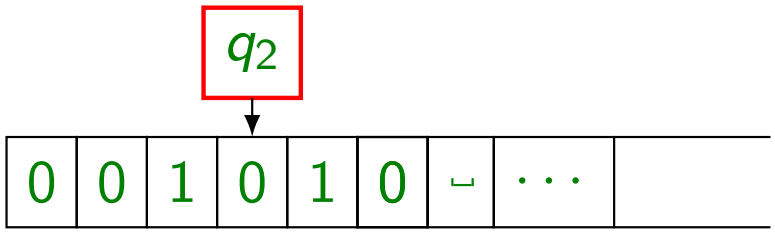
- Initially M receives as input $w = w_1 w_2 \dots w_n \in \Sigma^*$ on the leftmost n squares of the tape, and the rest of the tape is filled with blank symbols $\sqcup$
- The head starts on the leftmost square and the computation proceeds according to the transition function δ applied to the initial state and current positions, $\delta(q_0, w_1)$.
- If M ever tries to move its head left off the left-hand end of the tape, then the head stays in the same place for that move. (This overrides the instruction from the transition function to go left.)
- The computation continues until it enters the accept or reject states, at which point it halts. (If neither occurs then M goes on forever.)

Turing Machine configurations

As the Turing machine runs, it 'computes' and changes occur in its state, tape content, and current head location. A setting of these 3 items is called a configuration.

Configurations are often represented in the following way:

- For a state q and two strings u and v over the tape alphabet Γ we write $u\ q\ v$ for the configuration where the current state is q, the current tape contents is uv and the current location of the head is over the first (i.e. leftmost) symbol of v.
- For example $001q_20100$ represents the configuration where the tape contents is 0010100, the current state is q_2, and the head is located above the third zero of the tape contents:



Turing Machine configurations

We will say that a configuration C_1 yields a configuration C_2 if the Turing machine can legally go from C_1 to C_2 in a single step.

This notion of configuration provides another platform on which to see that behaviour of the system in terms of moving between configurations.

Suppose that $a, b, c \in \Gamma$ and $u, v \in \Gamma^*$.Let $q_i, q_j \in Q$. Let $C_1 = ua\ q_i\ bv$ and $C_2 = u\ q_j\ acv$ be two configurations. Then

$$ua\ q_i\ bv \text{ yields } u\ q_j\ acv \iff \delta(q_i, b) = (q_j, c, L)$$

$$ua\ q_i\ bv \text{ yields } uac\ q_j\ v \iff \delta(q_i, b) = (q_j, c, R).$$

Special cases occur when the head is at one of the ends of a configuration.

Exercise: Describe what happens in these cases.

Turing Machine configurations

The start configuration of M on input w is the configuration $C_1 = q_0 w$.

In an accepting configuration the state of the configuration is q_{accept}.

In a rejecting configuration the state of the configuration is q_{reject}.

Accepting and rejecting configurations are halting configurations and do not yield further configurations.

A Turing Machine accepts input w if a sequence of configurations $C_1, C_2, \ldots, C_k$ exists where

(1) C_1 is the start configuration of M on input w

(2) each C_i yields C_{i+1}

(3) C_k is an accepting configuration.

Turing-recognizable and Turing-decidable

The collection of strings that M accepts is the language recognized by M, and is denoted $L(M)$.

Definition

Call a language Turing-recognizable if some Turing machine recognizes it.

When a Turing machine is started on an input there are 3 possible outcomes: accept, reject, or loop. Looping means the machine will not halt.

A Turing machine can fail to accept an input by entering the q_{reject} state, or by looping.

Sometimes distinguishing a machine that is looping from one that is merely taking a long time is difficult. For this reason we prefer TMs that halt on all inputs, and therefore never loop.

Turing-recognizable and Turing-decidable

Machines which halt on all inputs are called deciders because they always make a decision to accept or reject.

A decider that recognizes some language is also said to decide that language.

Definition

Call a language Turing-decidable or simply decidable if some Turing machine decides it.

Some examples of decidable languages to follow in the next lecture.

Every decidable language is Turing-recognizable.

We present examples of languages that are Turing-recognizable but not decidable later in the course.