Turing Machines

Narges Khakpour

Department of Computer Science,
Linnaeus University

Plan

Introduction

- Automaton: an abstract machine with
  finite instructions and finite memory
- Context-free languages
  need unbounded memory to be recognized,
  e.g. $\{0^n 1^n \mid n \in \mathbb{N}\}$ requires unbounded counting.
- Need for an abstract machine with
  finite instructions and infinite memory.

Turing Machine

- An **abstract** computational model to simulate real computers by Alan Turing in 1937
- Simulate solving problems by human:
  - Human has a language
  - Reads/writes symbols written on a sheet of paper
  - The human's state of mind changes based on what it sees and changes what s/he has written accordingly
- Automation: design a machine that follows instructions and do computations instead of human
- What does it mean for a task to be computable?
- A task is computable if
  it can be carried out by executing a sequence of commands in a machine

Turing Machine

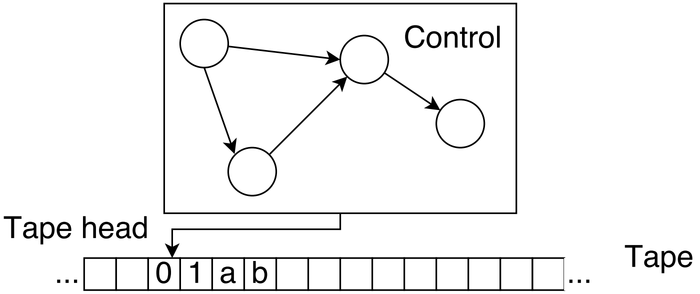
- An **abstract** computational model to simulate real computers by Alan Turing in 1937
- Simulate solving problems by human:
  - Human has a language
  - Reads/writes symbols written on a sheet of paper
  - The human's state of mind changes based on what it sees and changes what s/he has written accordingly
- Automation: design a machine that follows instructions and do computations instead of human
- What does it mean for a task to be computable?
- A task is computable if
  it can be carried out by executing a sequence of commands in a machine

Turing Machine

- To help show the limitations of what can be computed.
- Church-Turing Thesis: any computations that can be done in some way, can also be computed by a Turing machine.
  - For any problem L (given by a language) there exists an algorithm to solve that problem, if and only if there exists a Turing machine which terminates on every input.

Turing Machine

- A Turing machine is
  a finite automaton with
  an infinite tape as its memory
- A Turing machine consists of the following parts:
  - An infinite tape for input and blank spaces
  - A tape head that can read and write a single memory cell at a time.
  - A finite state control that issues commands



Turing Machine

At each step, the Turing machine

- reads the symbol from the tape cell under the tape head and writes a symbol to that cell,
- changes the state in the finite state control, and
- moves the tape head to the left or to the right

Turing Machine

- A Turing machine has two alphabets
  - Input alphabet Σ that is the alphabet of input written to the tape
  - Tape alphabet T that is all the symbols that can be written to the tape and $\Sigma \subset T$
- T contains at least the blank space ε and $\varepsilon \notin \Sigma$
- Input is written somewhere on the tape and is surrounded with an infinite number of blank cells
- The machine starts with the tape positioned at the start of input
- Once an accept/reject state is reached, the computation terminates.

Example 1

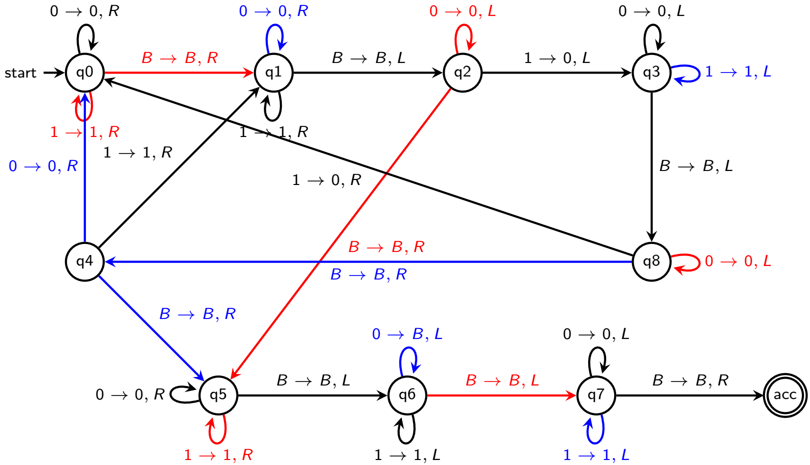
https://www.youtube.com/watch?v=E3keLeMwfHY&feature=youtu.be

http://aturingmachine.com/examplesSub.php

Example 1

- Subtraction of two numbers written on the tape and separated by a space.
- We show n with n number of 1's on the tape.
- Assume that left number is greater than or equal to the right number.

Example 1


start
q0
q1
q2
q3
q4
q5
q6
q7
q8
acc
0 → 0, R
0 → 0, R
0 → 0, L
0 → 0, L
B → B, R
B → B, L
1 → 0, L
1 → 1, L
1 → 1, R
1 → 1, R
1 → 1, R
0 → 0, R
1 → 0, R
B → B, L
B → B, R
B → B, R
0 → 0, L
B → B, R
0 → B, L
0 → 0, L
0 → 0, R
B → B, L
B → B, L
B → B, R
1 → 1, R
1 → 1, L
1 → 1, L

Plan

Turing Machine

Definition

A Turing machine is a tuple $\langle Q, \Sigma, T, \delta, q_0, q_{acc}, q_{rej} \rangle$ where:

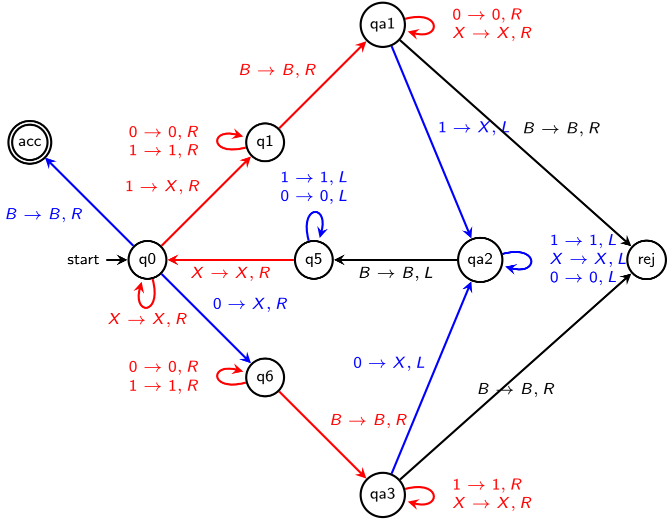
- Q is a finite set called the states;
- Σ is a finite set called the alphabet that does not contain the blank symbol B;
- T is a finite set called the tape alphabet, where $B \in T$ and $\Sigma \subset T$;
- $\delta : Q \times T \rightarrow Q \times T \times \{L, R\}$ is the partial transition function;
- $q_0 \in Q$ is the start state;
- $q_{acc} \in Q$ is the accept state, and
- $q_{rej} \in Q$ is the reject state, where $q_{acc} \neq q_{rej}$.

Turing Machine

A transition $\delta(q, s) = (q', s', D)$ means that if the machine is in the state q with the tape head pointing to a cell containing s, it first writes s' in that cell, moves the tape head to the direction D and evolves its state to the state q'.

Example 2

$\Sigma = \{1, 0\}$



Configurations

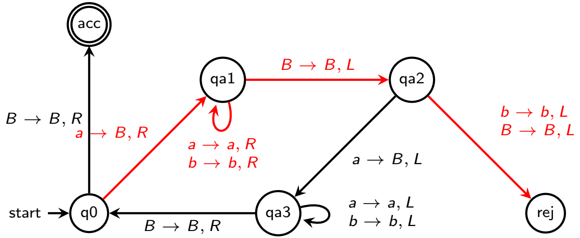
- A snapshot of the system during computation is called a configuration.
- A configuration includes
  - the current state
  - the tape contents
  - the current head location.

Configurations

- Formally, a configuration is a triple $\langle q, u, v \rangle \in Q \times T^* \times T^*$ where
  - q is the current state,
  - u is the tape content on the left of the tape head
  - v is the string on the right of the head including the cell under the tape head, i.e. the first symbol of v is the content of the cell under the tape head
- The start configuration is of the form $\langle q_0, \varepsilon, w \rangle$

Configurations

- A transition leads in changing the machine configuration
- A computation/execution is described by a maximal sequence of configurations $C_0 C_1 \ldots C_n$ where
  - C_0 is the initial configuration
  - the configuration C_{i-1} yields C_i by performing a transition, $1 < i$.



$\langle q0, \varepsilon, aaa\rangle$ $\langle qa1, \varepsilon, aa\rangle$ $\langle qa1, a, a\rangle$ $\langle qa1, aa, \varepsilon\rangle$ $\langle qa2, a, a\rangle$ $\langle qa3, \varepsilon, a\rangle$ $\langle qa3, a, \varepsilon\rangle$ $\langle qa3, \varepsilon, \varepsilon a\rangle$ $\langle q0, \varepsilon, a\rangle$ $\langle qa1, \varepsilon, \varepsilon\rangle$ $\langle qa2, \varepsilon, \varepsilon\rangle$ $\langle rej, \varepsilon, \varepsilon\rangle$

Plan

Turing Language

- An execution/computation is either
  - infinite, or
  - ends in a configuration in which the state is accepting, or
  - ends in a configuration from which no further configuration can be derived, or
  - in a reject state.
- An accepting configuration is of the form $\langle q_{acc}, u, w \rangle$
- A configuration of the form $\langle q_{rej}, u, w \rangle$ is rejecting.

Turing Language

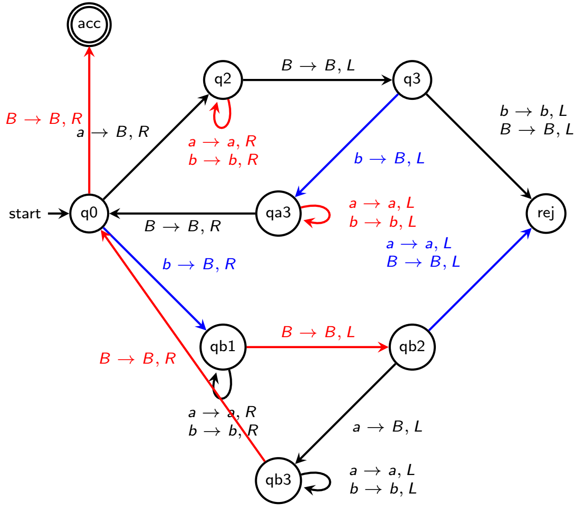
- A Turing machine may accept, reject or loop-forever on an input.
- A string $w \in \Sigma$ is accepted by the machine, if there is a configuration sequence $C_0 C_1 \ldots C_{acc}$ where $C_0 = \langle q_0, \varepsilon, w \rangle$ and $C_{acc} = \langle q_{acc}, u, v \rangle$ for some u and v.
- Language of a Turing machine M, denoted $L(M)$ is the set of strings accepted by M.

Example 3

- The language of Example 2 is

$$L(TM) = \{s \mid \begin{array}{l} s \in \{a, b\}^* \wedge \\ \text{number of 0's (1's) in the two strings are the same} \end{array} \}$$

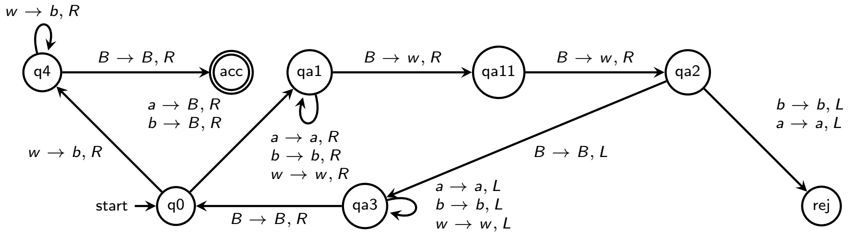
Example 4



BP What is the language accepted by this Turing machine?

Bonus Point

BP What is the language of the following TM? $\Sigma = \{a, b\}$



Turing-Recognizability

- A language L is called Turing-Recognizable, if there is a Turing machine that accepts it.
- If a machine M recognizes a language L, then:
  - The strings of L lead to an accepting configuration
  - The strings that are not in L either lead to a rejecting configuration or never terminates.
- To show that a language L is Turing-recognizable, we should build a Turing machine M and prove that M recognizes L

Turing-Recognizability and Turing-Decidability

- If a Turing machine never loop-forever on any input (always terminates), we call it a decider.
- A languages is called Turing-Decidable, if there is a decider Turing machine that accepts it.
- A problem is undecidable if no program can solve it.
- In our context, decision problem is deciding if a string is in a language or not.

BP Is the problem of checking if a number is even or not decidable?

Other Types of TM

- Multi-tape Turing machine, several tape heads work in parallel
- Multi-tape is as powerful as single-tape, i.e. any computation done by a multi-tape machine can also be performed by a single-tape machine
- Non-deterministic Turing machine: its control is non-deterministic, i.e. more than one transition might become enabled in a configuration
- Non-deterministic and deterministic Turing machines are equivalent

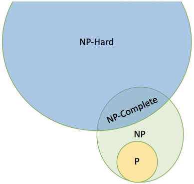
Plan

Running time of a TM

- The running time of a TM:
  the total number of steps (moving the tape head) before halting.
- if it doesn't stop, the running time is infinite
- The time complexity is a function $T(n)$
  returns the *maximum* number of taken steps of *all* possible input strings with the length n.
- We are interested in problems that its TM halts on every inputs, i.e. this can be done by a computer as well.
- **BP** Why polynomial time and not exponential or other high-growing time?

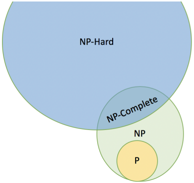
Inherent Intractability

- Decision problem is a problem whose answer is either yes or no
- P problems: set of (decision) problems solvable by a (deterministic) Turing machine in polynomial Time
  - i.e. the answer of yes or no can be decided in polynomial time
- NP (Non-deterministic Polynomial) problems : set of (decision) problems verifiable by a Turing machine in polynomial Time
  - given a guess at a solution for some instance of size n, we can check if the guess is correct in $O(n^k)$ time where k is a constant

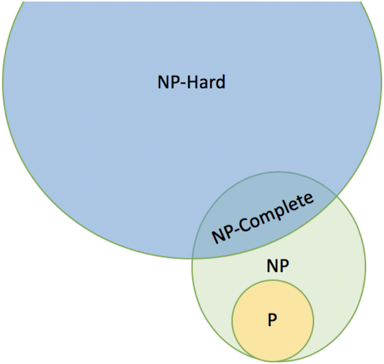


Inherent Intractability

- NP-complete:
  - problems in NP that can be proven to be as hard as any other problem in NP
  - the set that we can reduce any other NP problem to this set in polynomial time
  - *Satisfiability* is there a truth assignment that makes a logical expression e true?
- NP-hard:
  - problem not known to be in NP but as hard or harder than any problem in NP, i.e. at least as hard as any NP-problem
  - Tautology problem in propositional logic (Is E a tautology?)



Inherent Intractability



Assumption: $P \neq NP$ (An unsolved problem in computer science)
The P versus NP problem: to determine whether every language accepted by some nondeterministic algorithm in polynomial time is also accepted by some (deterministic) algorithm in polynomial time.

Plan

Conclusions

- An introduction to Turing machine
- Turing languages, decidablity, recognizability
- Time complexity
- Next lecture on propositional logic