Theory of Computation
The Church-Turing Thesis

Dimitris Diochnos
School of Computer Science
University of Oklahoma

Outline

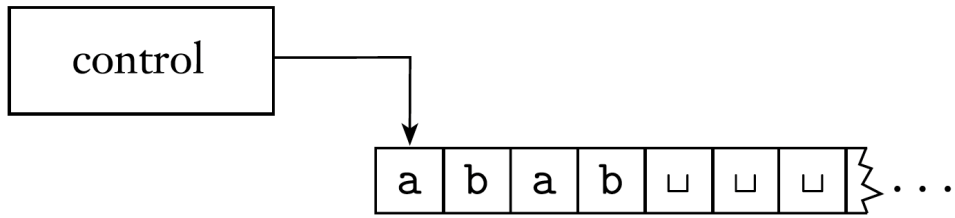
Introduction

- Some very easy tasks cannot be done by DFAs or PDAs.
- We need a different model for general purpose computers.

Table of Contents

Turing Machine

- Similar to a finite automaton, but it uses an unlimited and unrestricted tape as its memory.
- It has accept and reject states
- If it doesn't enter an accepting or a rejecting state, it will go on forever, never halting.



Differences with Finite Automata

Differences between finite automata and Turing machines:

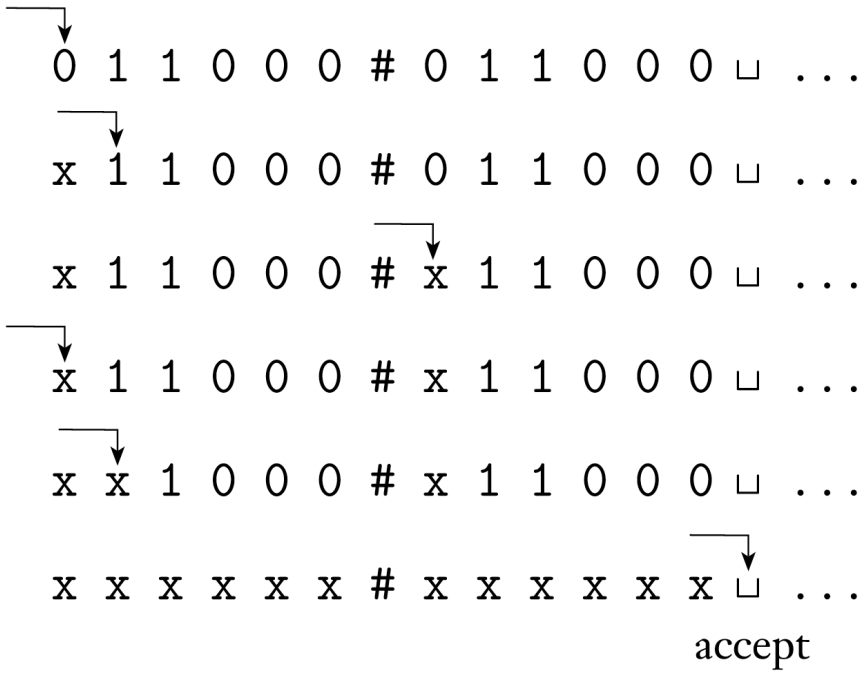
1. A Turing machine can also **write** on the tape (apart from reading).
2. The read/write head can move both to the left and to the right.
3. The tape is infinite.
4. The special states "accept" and "reject" take effect immediately.

Example

A TM for testing membership

Design a TM for testing membership in the language $B = \{w\#w | w \in \{0,1\}*\}$.

e.g.



Formal Definition of Turing Machine

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. $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function,
5. $q_0 \in Q$ is the start state,
6. $q_{accept} \in Q$ is the accept state, and
7. $q_{reject} \in Q$ is the reject state ($q_{reject} \neq q_{accept}$).

Computation

- Input $w = w_1 w_2 \cdots w_n$ written on the leftmost n squares of the tape (the rest of the tape is blank, i.e., filled with $\sqcup$'s).
- Head starts on the leftmost square of the tape (applying L on the head at the square leaves the head intact).
- Computation proceeds according to what δ dictates.
- Computation ends if either an accept or a reject state is reached.

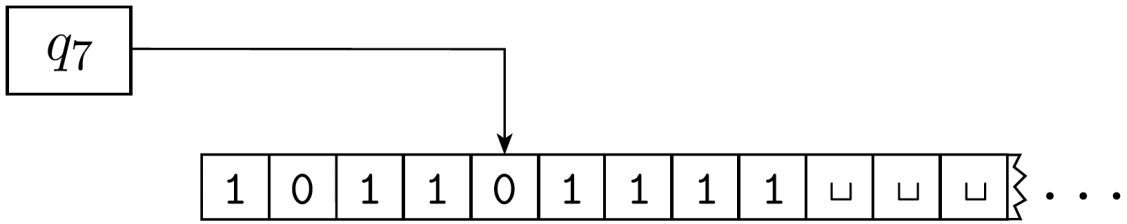
Configuration of a Turing Machine

Configuration of a Turing Machine:

- current state
- current tape contents
- current head location.

uqv: head is at leftmost symbol in v ($u, v \in \Gamma^*$).

E.g., $1011q_701111$ represents the configuration when the tape is 101101111, the current state is q_7, and the head is currently on the second 0.



Configuration of a Turing Machine

Say that a configuration C_1 **yields** configuration C_2 if the Turing machine M can legally go from C_1 to C_2 in a single step.

E.g., uaq_ibv yields uq_jacv if

$$\delta(q_i, b) = (q_j, c, L).$$

Note: Similarly rightward moves...

Configuration of a Turing Machine

- Start configuration of M on input w: $q_0 w$,
- Accepting and rejecting configurations are **halting configurations** and do not yield further configurations.
- A Turing machine M **accepts** input w if a sequence of configurations $C_1, C_2, \ldots, C_k$ exists, where
  - C_1 is the start configuration of M on input w,
  - each C_i yields C_{i+1}, and
  - C_k is an accepting configuration.
- The collection of strings that a TM M accepts is **the language of** M, or **the language recognized by** M, denoted $L(M)$.

Turing-recognizable and Turing-decidable languages

Definition 1

A **language** is called **Turing-recognizable** (or **recursively enumerable**) if some Turing machine recognizes it.

Definition 2

A **language** is called **Turing-decidable** (or **recursive**) or simply **decidable** if some Turing machine **decides it**.

Example

Example 3

Let $A = \{0^{2^n} | n \geq 0\}$, the language consisting of all strings of 0s whose length is a power of 2.
Give a Turing machine M_2 that **decides** this language.

Example (cont.)

Solution.

M2 = "On input string w:

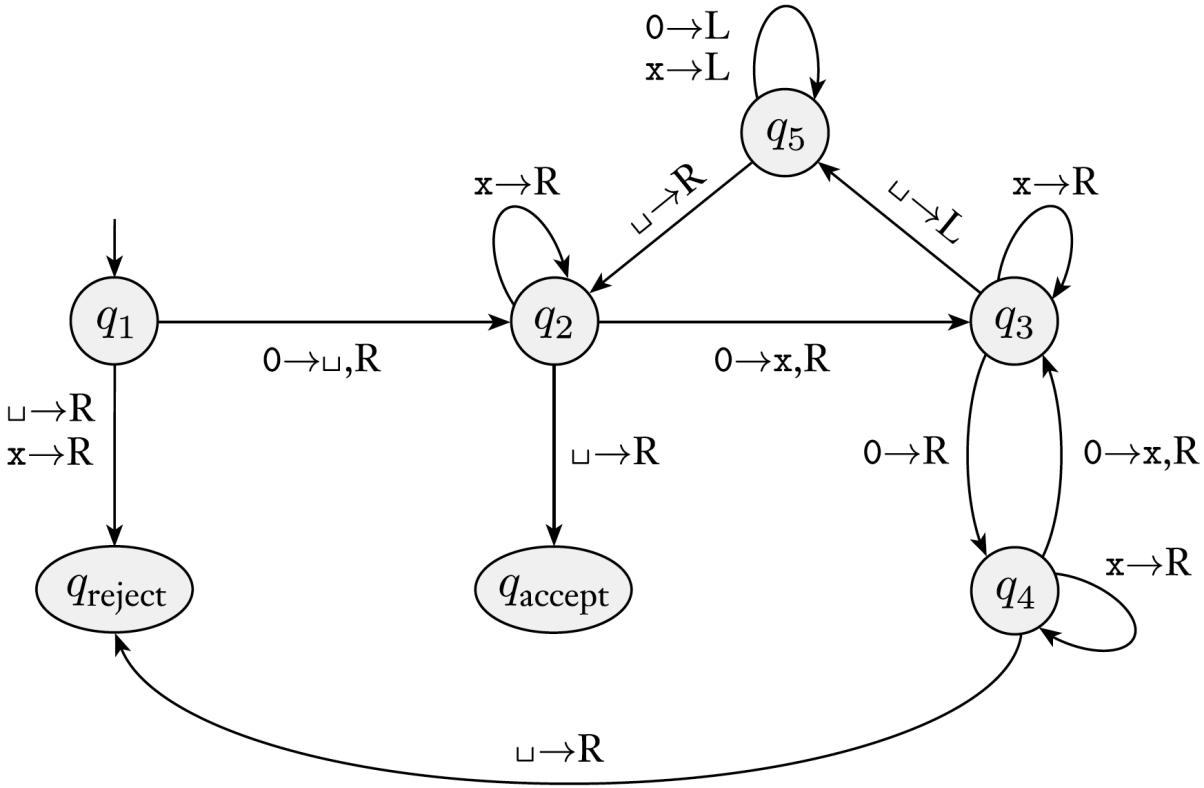
1. Sweep left to right across the tape, crossing off every other 0.
2. If in stage 1 the tape contained a single 0, accept.
3. If in stage 1 the tape contained more than a single 0 and the number of 0s was odd, reject.
4. Return the head to the left-hand end of the tape.
5. Go to stage 1."

Example (cont.)

Now we give the formal description of $M_2 = (Q, \Sigma, \Gamma, \delta, q_1, q_{\text{accept}}, q_{\text{reject}})$:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{\text{accept}}, q_{\text{reject}}\}$,
- $\Sigma = \{0\}$, and
- $\Gamma = \{0, x, \sqcup\}$.
- δ is given with a state diagram (see next slide).
- The start, accept, and reject states are q_1, q_{accept}, and q_{reject}, respectively.

Example (cont.)



- $a \to b, R$: means that we read 'a', overwrite it with 'b' and move (e.g., $0 \to \_, R$)
- $a \to L$: means that we read 'a', overwrite it with 'a' and move (i.e., the symbol did not change, so omit it; e.g., $0 \to R$)

Example

Example 4

Give a TM M_3 that **decides** the language
$C = a^i b^j c^k | i \times j = k$ and $i, j, k \geq 1$.

Example (Solution)

Solution.

$M_3 =$ "On input string w:

1. Scan the input from left to right to determine whether it is of the form $a^+b^+c^+$ and **reject** if it is not.
2. Return the head to the left-hand end of the tape. (Can be tricky! Can you do it? How?)
3. Cross off an a and scan to the right until a b occurs.
   Shuttle between the b's and the 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 c's also have been crossed off.
   - If yes, **accept**;
   - otherwise, **reject**."

Table of Contents

Introduction

- Multiple tapes, or with nondeterminism, etc.
- **Robustness**: invariance to the languages recognized by such modifications
- e.g., how to handle Turing machines that allow the head to stay put? (simulate...)

Multitape Turing machines

- Input on tape 1
- Other tapes initially blank
- $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$, where k is the number of tapes.

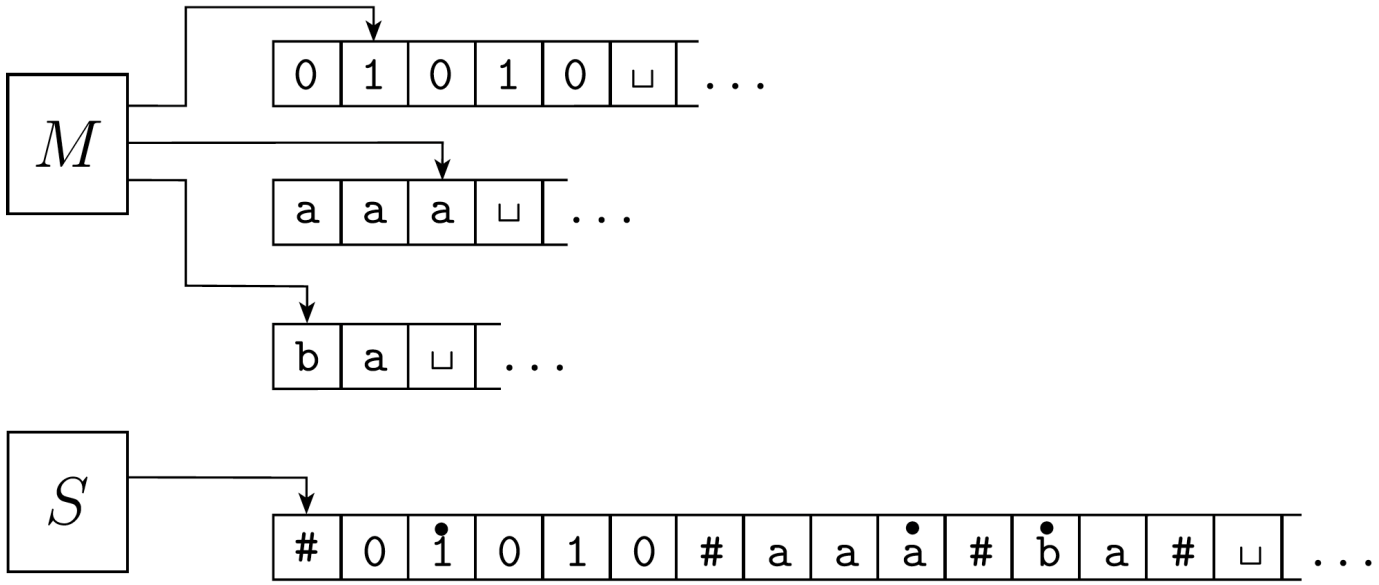
Note: S corresponds to 'stay'.

Multitape Turing machines

Theorem 5

Every multitape Turing machine has an equivalent single-tape Turing machine.

Proof.



The dots above symbols indicate where the heads are during the simulation.
[Initialize: $\#\dot{w}_1 w_2 \ldots w_n \# \dot{\sqcup} \# \dot{\sqcup} \# \ldots \#$] □

Multitape Turing machines

Corollary 6

A language is Turing-recognizable if and only if some multitape Turing machine recognizes it.

Nondeterministic Turing Machines

- $\delta : Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{L, R\})$.
- The TM accepts its input, if some branch of the computation leads to an accept state.

Nondeterministic Turing Machines

Theorem 7

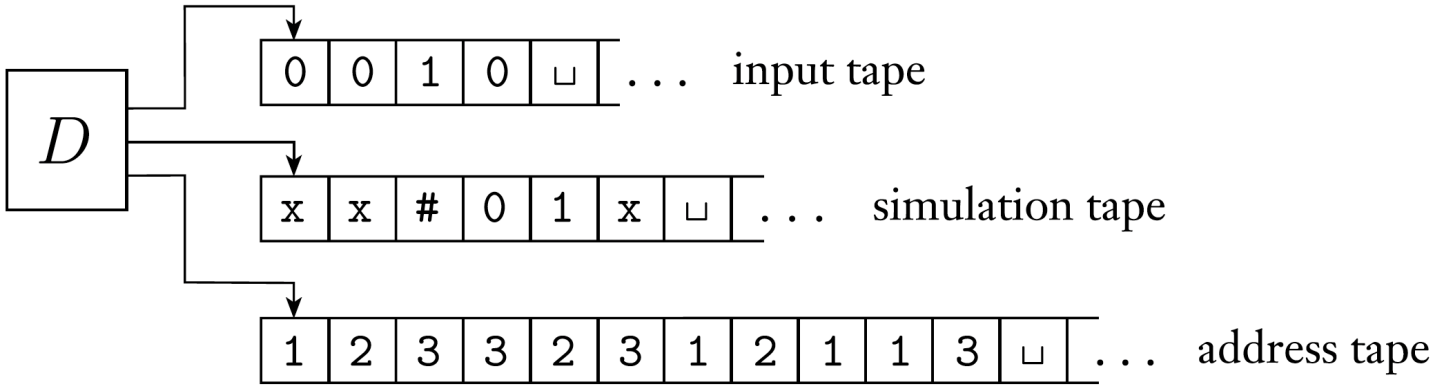
Every nondeterministic Turing machine has an equivalent deterministic Turing machine.

Proof idea.

Perform a breadth-first search (BFS) on the configuration tree that is produced when looking at all possible branches of computation.
(DFS is bad...) □

Nondeterministic Turing Machines

Proof.



Deterministic TM D simulating nondeterministic TM N

The sequence of numbers in the last tape indicates which choices we make in every branch, so that we reach a particular configuration in the computation.
The address may be invalid if, say, in some branches we have fewer than b (=3 in the example) options.

Corollary 8

A language is Turing-recognizable if and only if some nondeterministic Turing machine recognizes it.

Corollary 9

A language is decidable if and only if some nondeterministic Turing machine decides it.

Enumerators

- Enumerator = TM + Printer
- The language enumerated by an enumerator E is the collection (set) of all the strings that it eventually prints out.

Theorem 10

A language is Turing-recognizable if and only if some enumerator enumerates it.

Proof.

($\Rightarrow$)
Let $s_1, s_2, s_3, \ldots$ be a list of all strings in Σ^*.
Then construct the enumerator as follows:

1. Ignore the input
2. For $i = 1, 2, 3, \ldots$ do:
   Run M for i steps on each input $s_1, s_2, \ldots, s_i$.
   If any computations accept, print out the corresponding s_i's.

($\Leftarrow$)
"On input w for a TM M, do:

1. Run E. Compare each output string of E with w.
2. If w ever appears in the output of E, accept."

□

Equivalence of Turing Machines with other methods

- Unlimited memory (contrasting finite automata and PDAs)
- Limited computation in a single step
- Similar to the case of programming languages; the same algorithm can be written in **all the** languages
  $\Rightarrow$ **exactly** the same class of algorithms can be implemented by the various programming languages.

Table of Contents

The Definition of Algorithm

Algorithms are like recipes, as we know from other classes.

Hilbert's Problems

23 mathematical problems that appeared to be important for the previous century.
(Announced in 1900 at the International Congress of Mathematicians in Paris)

Tenth Problem

"Is there a process (algorithm) according to which it can be determined whether a polynomial has an integral root, by a finite number of operations?"

The Definition of Algorithm

We need a definition for the notion of an *algorithm*.

- 1936 papers: Alonzo Church (λ-calculus) and Alan Turing (Turing machines)
- 1970: Yuri Matijasevic showed that no algorithm exists (building on the work of Martin Davis, Hilary Putnam and Julia Robinson)

The Church–Turing thesis

Intuitive notion of algorithms equals Turing machine algorithms

The Definition of Algorithm

Let $D = \{p | p \text{ is a polynomial with an integral root}\}$.

Hilbert asks whether D is decidable.

The answer is NO.

BUT it is Turing-recognizable.

Note: p here is multi-variate. For uni-variate p the answer is "yes".