Comp 330 - Theory of Computation

Lecture 18 - Fall 2023

Claude Crépeau, Cesare Spinoso-Di Piano

November 2nd 2023

McGill University - School of Computer Science

Plan for today

1. Turing machines

IE: Un freddo da cane
A cold dog
A freezing cold

A4 due Friday, A5 released tomorrow evening

Prateek: 6 - 7:30 PM
MC 204
→ Halting problem
→ Intro comp theory

Turing machines

Automata theory

Objective: To create a theoretical model of computation.

Automata theory

Objective: To create a theoretical model of computation.

But why?

The Entscheidungsproblem revisited

The Entscheidungsproblem revisited

Is there a general effectively computable procedure for determining whether a given statement is provable? (Hilbert & Ackermann, 1928)

The Entscheidungsproblem revisited

log

Is there a general effective computable procedure for determining whether a given statement is provable? (Hilbert & Ackermann, 1928)

Effectively computable

At the time, a procedure/method was understood to be effectively computable when (and only when):

Effectively computable

At the time, a procedure/method was understood to be effectively computable when (and only when):

1. It could be expressed as a finite sequence of exact instructions.

Effectively computable

At the time, a procedure/method was understood to be effectively computable when (and only when):

1. It could be expressed as a finite sequence of exact instructions.
2. It would produce the desired result, if carried out without error, in a finite number of steps.

Effectively computable

At the time, a procedure/method was understood to be effectively computable when (and only when):

1. It could be expressed as a finite sequence of exact instructions.
2. It would produce the desired result, if carried out without error, in a finite number of steps.
3. It could be carried out by a human with a pencil, an eraser and paper.

Effectively computable

↳ Intuitive idea of an algorithm

At the time, a procedure/method was understood to be effectively computable when (and only when):

1. It could be expressed as a finite sequence of exact instructions.
2. It would produce the desired result, if carried out without error, in a finite number of steps.
3. It could be carried out by a human with a pencil, an eraser and paper.
4. It would require no ingenuity from the human carrying out the operation.

Alan Turing

The Turing machine

- The Turing machine[1]: The formalism Turing used to make this intuitive notion of effective computability precise.

[1] a-machine or logical computing machines

The Turing machine

- The Turing machine[1]: The formalism Turing used to make this intuitive notion of effective computability precise.

  λ-calculus: $\lambda xy.\, x+y$

- Church-Turing thesis: Anything that can be effectively computed can be computed by a Turing machine.

[1] a-machine or logical computing machines

The Turing machine

- The Turing machine[1]: The formalism Turing used to make this intuitive notion of effective computability precise.
- Church-Turing thesis: Anything that can be effectively computed can be computed by a Turing machine.
- Turing showed that, under this definition, the answer to the *Entscheidungsproblem* was no!

[1] a-machine or logical computing machines

The Turing machine

- The Turing machine[1]: The formalism Turing used to make this intuitive notion of effective computability precise.
- Church-Turing thesis: Anything that can be effectively computed can be computed by a Turing machine.
- Turing showed that, under this definition, the answer to the *Entscheidungsproblem* was no!
- What about other formalisms? → Quantum computers.
  1985

Turing complete

[1] a-machine or logical computing machines

The inspiration for the Turing machine

In 1936, a computer was *a human computer*.

The inspiration for the Turing machine

In 1936, a computer was *a human computer*. To formalize effective computation Turing asked:

> *What are the most basic operations and resources required for a human computer to carry out numerical computation?*

Addition

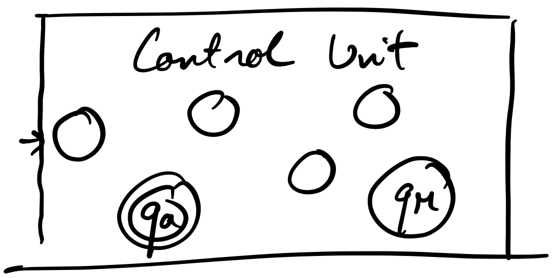
$$
\begin{array}{rccc}
 & 2 & 9 & 3 \\
+ & & 4 & 2 \\
\hline
 & 3 & 3 & 5
\end{array}
$$

Addition

$$\begin{array}{cccc} & 2 & 9 & 3 \\ + & & 4 & 2 \\ \hline & 3 & 3 & 5 \end{array}$$

Turing Machines

Def (Informal) A deterministic semi-infinite tape TM is a computational model with:

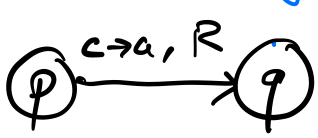


Semi-infinite tape — unbounded

⊢ a b c □ □ □

left endmarker

Pointer
- Read & write
- Move right & left



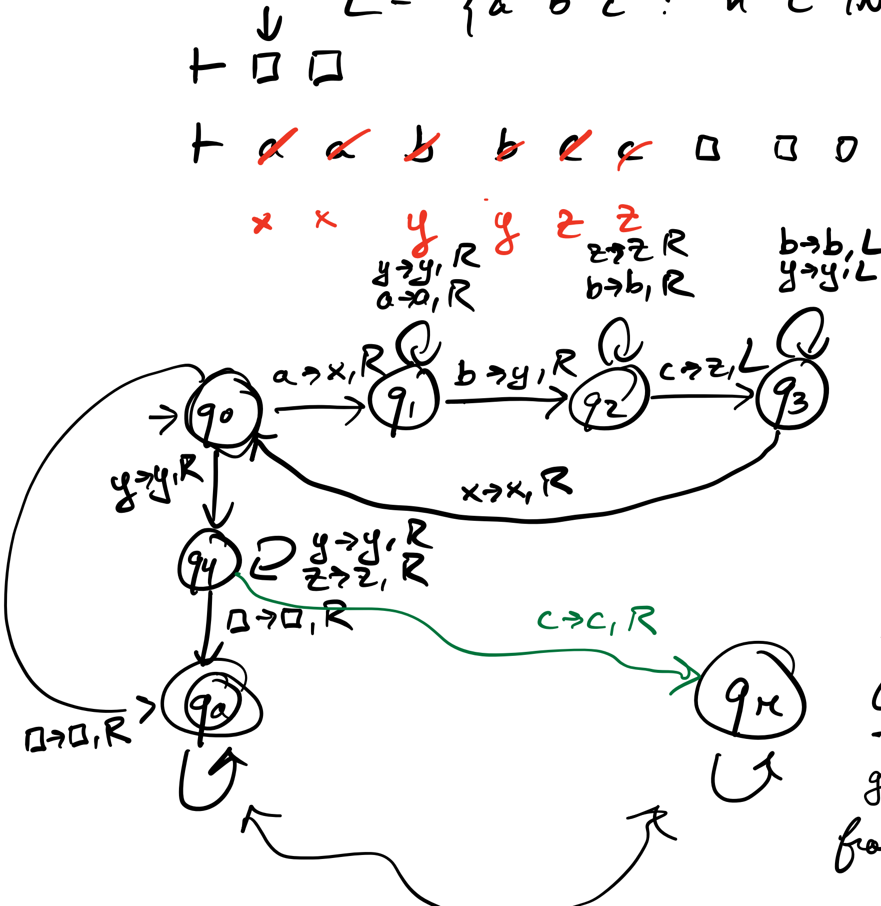
2-D Tape

0 ↔ 1 . . . 9 ↔ 100
0 9

⊢ + # 42 # . . . # □ □ □

qa → as the return state

Ex 1 Design a TM which recognizes
$L = \{a^n b^n c^n : n \in \mathbb{N}\}$

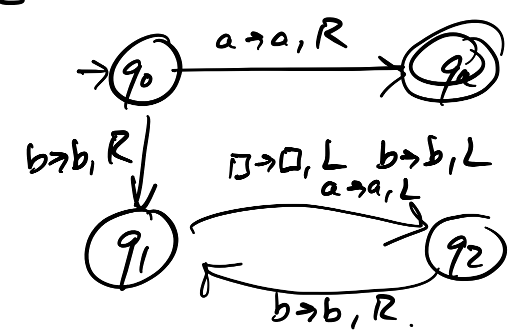


using different symbols is required for determinism.

For convenience: Omit any transition going to q_r from q_0 to q_4

Once you reach q_a or q_r, TM halts.

Ex 2 What strings are accepted by the following TM?



$\Sigma = \{a, b\}$

- $\vdash a$ $\qquad$ $L(M) = L(a(a+b)^*)$

  $\hookrightarrow$ language accepted / recognized

- $\vdash b$ $\quad$ a $\quad$ M loops on $L(b(a+b)^*)$.

  b

  $\square$

Any Turing complete formalism will have ∞-loops.

- ε is the only rejected string defined over Σ

Ex 3 $\quad$ $\Sigma = \{a\}$. Design a TM which computes the function copy(w).

$w \in \Sigma^+$ $\quad$ $\text{copy}(w) = w \cdot w$.

$$\vdash \text{input} \rightarrow \boxed{M} \rightarrow \vdash f(\text{input})$$

$$\vdash aa \rightarrow \boxed{M} \quad \vdash aa\ aa\ \square$$

$\vdash \cancel{a}\cancel{a}\ \cancel{\square}\ \cancel{\square}$

$\times \times\ \alpha\ \alpha$

$aa\ \ a\ a\ \square$

General procedure

M : On input $x \in \Sigma^+$

1. Mark a under the pointer with x
2. Move pointer right until it hits the $\square$
3. Write α
4. Move pointer left until it hits the first x
5. Move pointer to the right by 1 step
6. If it reads an a, repeat from Step 1. Otherwise, move pointer right until the first blank and then move left converting α to a & x to a.
7. Once pointer reaches $\vdash$, it moves right once & returns ($\rightarrow q_a$)

Leave state transition diag as exercise.
Will post later.

Def (TM) A deterministic semi-inf tape TM is a 9-tuple

$$(Q, \Sigma, \Gamma, \vdash, \square, \delta, s, q_a, q_r)$$

Q is the finite set of states
Σ is the input alphabet $\quad \Sigma \subseteq \Gamma$
Γ is the tape alphabet

$\vdash$ is the left endmarker $\quad \vdash \in \Gamma - \Sigma$

$\square$ is the blank symbol $\quad \square \in \Gamma - \Sigma$

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

$s \in Q$ is the start state

$q_a \in Q$ is the accept state

$q_r \in Q$ is the reject state.

$$\delta(\underset{}{q}, \underset{\text{read}}{a}) = (p, \underset{\text{write}}{b}, \underset{\text{direction}}{d}) \qquad q \xrightarrow{a \to b,\, d} p$$

Additional restrictions on δ:

$\forall q \in Q, \ \exists p \in Q$

$\quad \delta(q, \vdash) = (p, \vdash, R)$

$\forall \gamma \in \Gamma, \ \exists c, c' \in \Gamma, \ d, d' \in \{L, R\}$

$\quad \delta(q_a, \gamma) = (q_a, c, d)$

$\quad \delta(q_r, \gamma) = (q_r, c', d')$

Instantaneous config (IC) of TMs

$\Downarrow$

$\vdash u q a v \square \rightsquigarrow \square^\omega \ (\square\square\square\cdots)$

Def An IC of a TM M is a string from $\{\vdash\} \cdot \Gamma^* \cdot \underline{Q} \cdot \Gamma^* \cdot \{\square\}$

$\Downarrow$

"q_0" $\to$ interpreted as a letter

Def (Next-1 config relation $\xrightarrow[M]{1}$) M TM, $q, p \in Q$, $u, v \in \Gamma^*$, $a, b, c \in \Gamma$

$$\vdash u a q b v \square \xrightarrow[M]{1} \vdash u p a c v \square \qquad \delta(q,b) = (p,c,L)$$

$$\vdash u a q b v \square \xrightarrow[M]{1} \vdash u a c p v \square \qquad \delta(q,b) = (p,c,R)$$

<u>Edge cases</u> $\quad \ldots q \square \xrightarrow[M]{1} b p \square$

$$\vdash q w \xrightarrow[M]{1} p \vdash w$$

Write down the edge cases as an exercise.

Def ($\xrightarrow[M]{n}$) M TM, $n \in \mathbb{N}$, C, D IC

$$C \xrightarrow[M]{0} C \quad \forall C$$

$$C \xrightarrow[M]{n+1} D \qquad \exists \text{ IC } E \quad C \xrightarrow[M]{n} E \quad E \xrightarrow[M]{1} D$$

Def ($\xrightarrow[M]{*}$) M TM, $n \in \mathbb{N}$, C, D IC

$C \xrightarrow[M]{*} D$ if $\exists\, n \in \mathbb{N}$ s.t.
$C \xrightarrow[M]{n} D$

Given an input string $w \in \Sigma^*$, a TM M can either:

1) Accept w if $\exists\, x, y \in \Gamma^*$
$\vdash s w \square \xrightarrow[M]{*} \vdash x q_a y \square$

2) Reject w if $\exists\, x, y \in \Gamma^*$
$\vdash s w \square \xrightarrow[M]{*} \vdash x q_r y \square$

3) Loop (forever) on w if
$\nexists\, x, y \in \Gamma^*$ s.t. 1 or 2 holds

If M accepts or rejects on w, we say it halts on w. Otherwise it loops.

Def $\Sigma \neq \emptyset$, M TM, then
$L(M) = \{ w \in \Sigma^* : M \text{ accepts } w \}$
