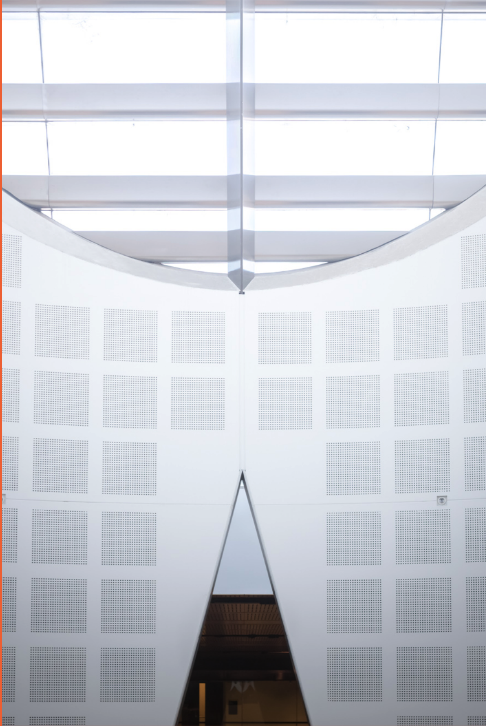
COMP2022|2922 Models of Computation

Turing Machines are Robust

Sasha Rubin

August 31, 2022

THE UNIVERSITY OF SYDNEY



Turing machines are fairly robust, i.e., variations and extensions of the basic model do not change the languages that can be recognised.[1]

- Let's call our TMs the basic TMs.
  - deterministic
  - single doubly-infinite tape
  - can move left, right, or stay
- Two machines are equivalent if they recognise the same language.

[1] Although "robustness" is not a formally defined concept, we will justify it with theorems about variations of TMs.

Must-move TMs

Basic TMs can move left, right or stay put in one step. Sipser's variation disallows 'stay' – let's call them must-move TMs.

Theorem

Every basic TM is equivalent to a must-move TM.

Proof idea of how to simulate our TMs by a must-move TM.

– Replace each S-transition with two transitions, an R-transition followed by an L-transition, *i.e.*, replace

```
q_i  a  b  S  q_j   by   q_i  a  b  R  q_i,j
                         q_i  *  *  L  q_j
```

Left-bounded TM

- The head starts on the left-most square of the tape with the head on the first letter of the input.
- Should the transition function suggest to move left when the head is already at the left-most position, the head stays put.

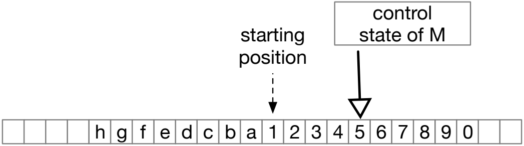
Theorem
Every basic TM is equivalent to a left-bounded TMs.[2]

Proof idea of how to simulate a doubly infinite TM by a left-bounded one.

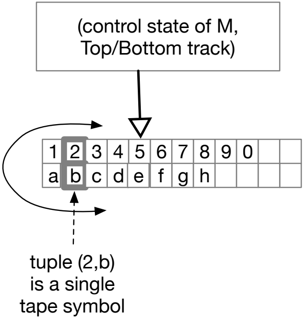
- Split the left-bounded tape into an upper and a lower half.
- The upper half represents the right half of the tape (from the initial head position) and the lower half represents the left half.
- An extra state component keeps track of the half in which the head currently is.

[2] The vice-versa is a tutorial question.

doubly
infinite tape



simulation using
left-bounded
infinite tape



Multitape TMs

- A multitape TM has multiple tapes, each with its own head for reading and writing.
- Initially the input appears on tape 1, and the others start out blank.
- The type of the transition function of a k-tape TM becomes:

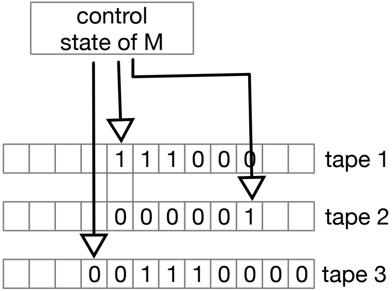
$$\delta : Q \times \Gamma^k \to \Gamma^k \times \{L, R, S\}^k \times Q$$

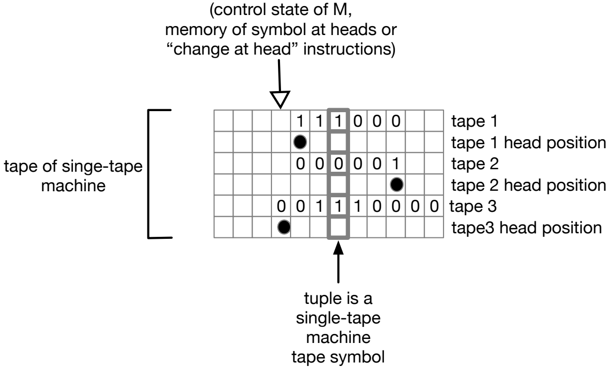
Theorem

Every multitape TM has an equivalent basic TM.

Proof Idea.

Split the tape into $2k$-many "tracks", with the $(2i-1)$th track corresponding to the ith tape and the $2i$th track storing the position of the ith head. So, the input alphabet contains symbols that correspond to a “slice” of all the tapes (including their heads).


control
state of M
1 1 1 0 0 0
tape 1
0 0 0 0 0 1
tape 2
0 0 1 1 1 0 0 0 0
tape 3

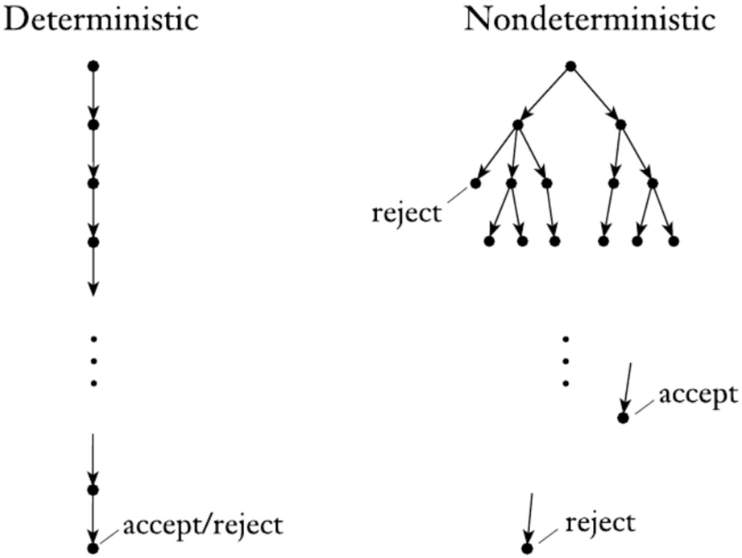

(control state of M,
memory of symbol at heads or
“change at head” instructions)
tape of singe-tape
machine
1 1 1 0 0 0
tape 1
tape 1 head position
0 0 0 0 0 1
tape 2
tape 2 head position
0 0 1 1 1 0 0 0 0
tape 3
tape3 head position
tuple is a
single-tape
machine
tape symbol

Non-Deterministic TMs

The type of the transition function of a non-deterministic TM N becomes:

$$\delta : Q \times \Gamma \rightarrow P(\Gamma \times \{L, R, S\} \times Q)$$

- A computation of N on an input u is a tree.
- If some branch of the tree has an accepting configuration then N accepts u.
- So, if no branch of the tree has an accepting configuration (because each rejects or diverges) then N does not accept u.


Deterministic
Nondeterministic
reject
accept
accept/reject
reject

Theorem

Every non-deterministic TM N has an equivalent deterministic TM D.

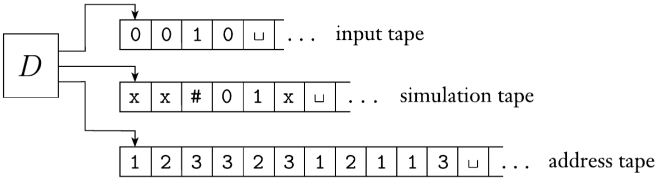
The idea is to that D will "search the computation tree of N".

High-level description of TM D:

- D does a breadth-first search of N's computation tree on given input. If find q_{accept} accept, otherwise diverge.

Implementation level description.

- Three tapes.
- Tape 1 holds the input, Tape 2 simulates N, Tape 3 keeps track of D's location in N's computation tree.
  - if N has at most b choices (from every state/symbol), then every node in N's computation tree is addressed by a string over alphabet $\{1, 2, 3, \cdots, b\}$.
  - e.g., string 231 is an address: take the second child of the root, the third child of that, the first child of that.
- D repeatedly replaces the address on tape 3 by the successive address (in the BFS order), and then simulates N on tape 2.
- If D ever finds a q_{accept} then it accepts. Otherwise it runs forever.



Suppose D were to use a **depth-first search (DFS)** instead. Would this work?

Vote now! (on mentimeter)

1. Yes, because we know that BFS and DFS both end up traversing the whole tree.
2. No: D might reach an accepting configuration even if N can't.
3. No: D might reach a rejecting configuration even if N can't.
4. No: D might not reach an accepting configuration even if N can.
5. No: D might not reach a rejecting configuration even if N can.

Takeaway

- Turing machines are a robust model of computation.
- The set of recognisable languages doesn't change if one makes certain variations (e.g., more tapes, allowing nondeterminism).

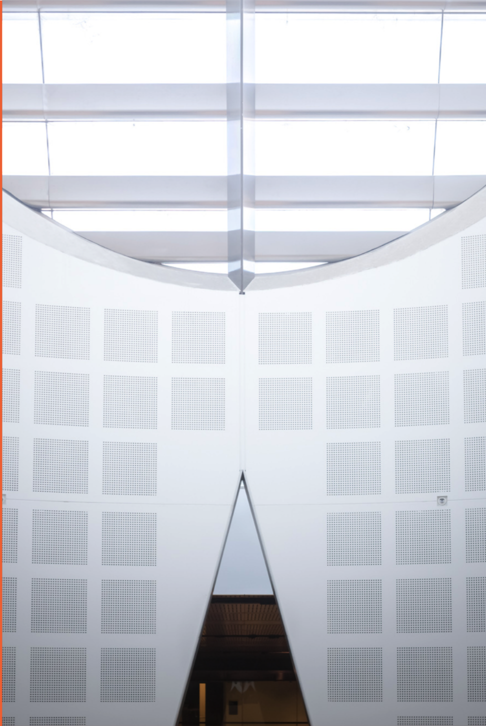
COMP2022|2922
Models of Computation

Decidability
Sipser Chapter 4

Sasha Rubin

August 31, 2022

THE UNIVERSITY OF SYDNEY



From week 1

We have been studying computational problems of the following form, for languages L:

Is the input string in L?

E.g., $L = L(R)$ for an RE R, or $L = L(M)$ for an automaton M.

What about other objects?

- How do we encode integers as strings?
- How do we encode graphs as strings?
- How do we encode TMs as strings?
- etc.

Encodings

- The input to a Turing Machine is always a string, but it can be useful to work with objects other than strings over Σ.
- We can encode almost any object as a string (integers, sets of integers, graphs, programs, Turing machines!)
- If O is an object, an encoding of O over Σ, written $\langle O \rangle$ is a representation of O as a string.
- There are many ways to encode objects ...

Encoding numbers as strings

- The number $n \in \mathbb{N}$ can be encoded as a string $\langle n \rangle$ in some fixed base.
- E.g., the number 3 is encoded by the string
  - "111" if we are using a unary encoding
  - "11" in binary
  - "10" in ternary
  - "3" in decimal
  - etc
  - In binary, we would write $\langle 3 \rangle = 11$.
- Integers can be encoded as a string by adding a symbol for the sign.
  - E.g., $\langle -3 \rangle = -11$ if we were using binary.

Encoding sequences of strings

- List the strings in order, separated by a new alphabet symbol.
- So the sequence $00, 1, 000, 10$ of binary strings is encoded by the string

  $$00\#1\#000\#10\#$$

  We would write $\langle 00, 1, 000, 10 \rangle = 00\#1\#000\#10\#$.
  Here I decided to use $\#$ as a separator.
- We can encode sets of strings in a similar way.

Encoding TMs

We already saw a specific encoding of a TM as a sequence of strings. E.g.,

q0	_	_	S	halt-accept
q0	a	a	R	q1
q0	b	b	R	q2
q1	a	a	R	q1
q1	b	b	R	q2
q1	_	_	R	halt-accept
q2	b	b	R	q2
q2	_	_	S	halt-accept
q1	a	a	S	halt-reject

- We can encode this as a single string by concatenating the lines, using B for Blank.
- So $\langle M \rangle =$ q0#B#B#S#halt-accept#q0#a#a#R#q1# ...#q1#a#a#S#halt-reject#

Encoding graphs

- A directed graph $G = (V, E)$ consists of a set V of vertices and a set $E \subseteq V \times V$ of edges.
- How can we encode it as a string $\langle G \rangle$?
- Say $V = \{1, 2, 3\}$ and $E = \{(1, 2), (1, 3), (3, 1)\}$.
  1. List the vertices, then list the edges (similar to Sipser)

     $\langle G \rangle = 1\#2\#3||1\#2||1\#3||3\#1$

     Here I decided to use the symbol $||$ as another separator.
  2. Adjacency matrix

     $\langle G \rangle = 011\#000\#100\#$
- In a similar we can encode automata.

  Why? since automata are just labeled graphs.
- So, e.g., $\langle D, w \rangle$ is an encoding of a DFA D and string w.

Decidability

Recall:

Definition

- If you run a basic TM on an input one of three things can occur:
  1. the TM eventually halts in an accept state;
  2. the TM evenutally halts in a reject state;
  3. the TM doesn't enter a halting state (aka it diverges, i.e., 'runs forever').
- A basic TM is a decider if it halts on every input.
- A language L is called Turing-decidable[3] if $L = L(M)$ for a decider M.

[3] aka decidable, computable, recursive

Decidable problems about automata

The acceptance problem for DFAs[4] is the language

$$L_{\textsf{DFA-acceptance}} = \{\langle D, w\rangle \mid D \text{ is a DFA that accepts } w\}$$

This problem is decidable.

Here is a high-level description of a decider for this language:

1. Simulate D on word w.
2. If D ends in an accepting state then accept, else reject.

[4] aka membership problem for DFAs

Decidable problems about RE and Automata

The following languages are decidable (tutorial):

$$
\begin{aligned}
L_{\text{NFA-acceptance}} &= \{\langle N, w\rangle \mid N \text{ is an NFA that accepts } w\} \\
L_{\text{RE-acceptance}} &= \{\langle R, w\rangle \mid R \text{ is a RE and } w \in L(R)\} \\
L_{\text{DFA-emptiness}} &= \{\langle D\rangle \mid D \text{ is a DFA and } L(D) = \emptyset\} \\
L_{\text{DFA-equivalence}} &= \{\langle A, B\rangle \mid A, B \text{ are DFAs and } L(A) = L(B)\}
\end{aligned}
$$

Decidable problems about TMs?

The acceptance problem for TMs is the language

$$L_{\textsf{TM-acceptance}} = \{\langle M, w\rangle \mid M \text{ is a TM that accepts } w\}$$

- This language is recognisable.
- To show this we will build a TM U that recognises it.

High level description of U:

- on input $\langle M, w\rangle$:
- simulate M on w and accepts if M enters q_{accept} and rejects if M enters q_{reject} (and diverge otherwise).

The TM U is called a universal TM since it can do what any other TM can do.

Decidable problems about TMs?

Implementation level description of U:

- Tape 1 holds the transition function δ_M of the input TM M.
- Tape 2 holds the simulated contents of M's tape.
- Tape 3 holds the current state of M, and the current position of M's tape head.
- U simulates M on input x one step at a time:

  In each step, it updates M's state and simulated tape contents and head position as dictated by δ_M. If ever M halts and accepts or halts and rejects, then U does the same.

Self-test

Is the TM U a decider?

Vote now! (on mentimeter)

1. Yes.
2. No.
3. It depends.

Decidable problems about TMs?

The acceptance problem for TMs is the language

$$L_{\textsf{TM-acceptance}} = \{\langle M, w\rangle \mid M \text{ is a TM that accepts } w\}$$

- U is a blueprint for a general purpose computer – it takes TMs (programs!) as input and executes them.
- This is similar to an interpreter of `C` written in `C`.
- U recognises the language, but it doesn't decide it. Why?
- Ok, but is the acceptance problem for TMs decidable?

Theorem (Turing)

$\{\langle M, w\rangle \mid M$ *is a TM that accepts* $w\}$ *is not decidable.*