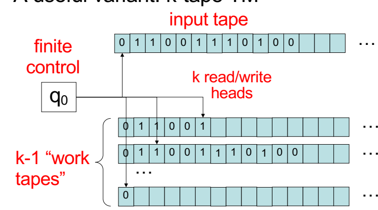
CS21
Decidability and Tractability

Lecture 10
January 29, 2025

1

Multitape TMs

- A useful variant: k-tape TM



2

Multitape TMs

Theorem: every k-tape TM has an equivalent single-tape TM.

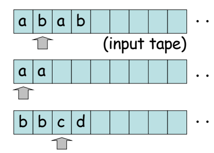
Proof:

– Idea: simulate k-tape TM on a 1-tape TM.

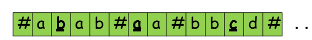
3

Multitape TMs

simulation of k-tape TM by single-tape TM:

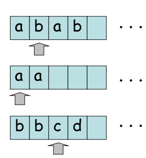


- add new symbol **x̲** for each old x
- marks location of "virtual heads"



4

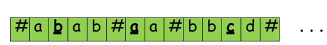
Multitape TMs



. . . Repeat:

- scan tape, remembering the symbols under each virtual head in the state (how many new states needed?)
- make changes to reflect 1 step of M
- if hit #, shift to right to make room

if M halts, erase all but 1st string



5

Nondeterministic TMs

- A important variant: nondeterministic TM
- informally, several possible next configurations at each step
- formally, a NTM is a 7-tuple

  $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ where:

  – everything is the same as a TM except the transition function:

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

6

NTM acceptance

- start configuration: q_0w (w is input)
- accepting config.: any config.with state q_{accept}
- rejecting config.: any config. with state q_{reject}

NTM M accepts input w if there exist configurations $C_1, C_2, \ldots, C_k$

- C_1 is start configuration of M on input w
- $C_i \Rightarrow C_{i+1}$ for i = 1, 2, 3, …, k-1
- C_k is an accepting configuration

Nondeterministic TMs

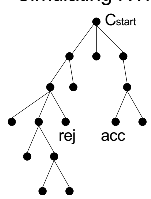
Theorem: every NTM has an equivalent (deterministic) TM.

Proof:

- Idea: simulate NTM with a deterministic TM

Nondeterministic TMs

Simulating NTM M with a deterministic TM:



- computations of M are a tree
- nodes are configs
- fanout is b = maximum number of choices in transition function
- leaves are accept/reject configs.

Nondeterministic TMs

Simulating NTM M with a deterministic TM:

- idea: breadth-first search of tree
- if M accepts: we will encounter accepting leaf and accept
- if M rejects: we will encounter all rejecting leaves, finish traversal of tree, and reject
- if M does not halt on some branch: we will not halt…

Nondeterministic TMs

Simulating NTM M with a deterministic TM:

- use a 3 tape TM:
  - tape 1: input tape (read-only)
  - tape 2: simulation tape (copy of M's tape at point corresponding to some node in the tree)
  - tape 3: which node of the tree we are exploring (string in {1,2,…b}*)
- Initially, tape 1 has input, others blank
- STEP 1: copy tape 1 to tape 2

Nondeterministic TMs

Simulating NTM M with a deterministic TM:

- STEP 2: simulate M using string on tape 3 to determine which choice to take at each step
  - if encounter blank, or a # larger than the number of choices available at this step, abort, go to STEP 3
  - if get to a rejecting configuration: DONE = 0, go to STEP 3
  - if get to an accepting configuration, ACCEPT
- STEP 3: replace tape 3 with lexicographically next string and go to STEP 2
  - if string lengthened and DONE = 1 REJECT; else DONE = 1

Examples of basic operations

- Convince yourself that the following types of operations are easy to implement as part of TM "program"

  (but perhaps tedious to write out…)

  – copying
  – moving
  – incrementing/decrementing
  – arithmetic operations +, -, *, /

13

Universal TMs and encoding

- the input to a TM is always a string in Σ^*
- often we want to interpret the input as representing another object
- examples:
  – tuple of strings (x, y, z)
  – 0/1 matrix
  – graph in adjacency-list format
  – Context-Free Grammar

14

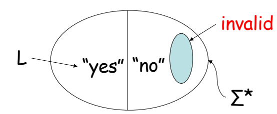
Universal TMs and encoding

- the input to a TM is always a string in Σ^*
- we must encode our input as such a string
- examples:
  – tuples separated by #: #x#y#z
  – 0/1 matrix given by: #n#x# where $x \in \{0,1\}^{n^2}$
- any reasonable encoding is OK
- emphasize "encoding of X" by writing <X>

15

Universal TMs and encoding

- some strings not valid encodings and these are not in the language



make sure TM can recognize invalid encodings and reject them

16

Universal TMs and encoding

- We can easily construct a Universal TM that recognizes the language:

  A_{TM} = {<M, w> : M is a TM and M accepts w}

  – how?
- this is a remarkable feature of TMs (not possessed by FA or NPDAs…)
- means there is a general purpose TM whose input can be a "program" to run

17

Church-Turing Thesis

- many other models of computation
  – we saw multitape TM, nondeterministic TM
  – others don't resemble TM at all
  – common features:
    - unrestricted access to unlimited memory
    - finite amount of work in a single step
- every single one can be simulated by TM
- many are equivalent to a TM
- problems that can be solved by computer does not depend on details of model!

18

Church-Turing Thesis

- the belief that TMs formalize our intuitive notion of an algorithm is:

  The Church-Turing Thesis

  everything we can compute on a physical computer

  can be computed on a Turing Machine
- Note: this is a belief, not a theorem.

19

Recursive Enumerability

- Why is "Turing-recognizable" called RE?
- Definition: a language $L \subseteq \Sigma^*$ is recursively enumerable if there is exists a TM (an "enumerator") that writes on its output tape

  $\#x_1\#x_2\#x_3\#\ldots$

  and $L = \{x_1, x_2, x_3, \ldots\}$.
- The output may be infinite

20

Recursive Enumerability

Theorem: A language is Turing-recognizable iff some enumerator enumerates it.

Proof:

($\Leftarrow$) Let E be the enumerator. On input w:

- Simulate E. Compare each string it outputs with w.
- If w matches a string output by E, accept.

21

Recursive Enumerability

Theorem: A language is Turing-recognizable iff some enumerator enumerates it.

Proof:

($\Rightarrow$) Let M recognize language $L \subseteq \Sigma^*$.

- let $s_1, s_2, s_3, \ldots$ be enumeration of Σ^* in lexicographic order.
- for i = 1,2,3,4,…
  - simulate M for i steps on $s_1, s_2, s_3, \ldots, s_i$
- if any simulation accepts, print out that s_j

22