CS21 Decidability and Tractability

Lecture 11
January 30, 2023

1

Outline

- Turing Machines and variants
- Church-Turing Thesis
- decidable, RE, co-RE languages
- Recursive Enumerability
- a non-RE language

2

Nondeterministic TMs

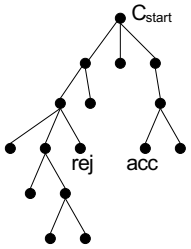
Theorem: every NTM has an equivalent (deterministic) TM.

Proof:
– Idea: simulate NTM with a deterministic TM

3

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.

4

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…

5

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

6

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

7

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 +, -, *, /

8

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

9

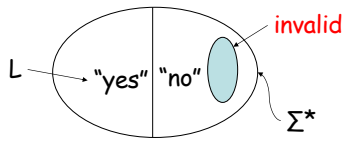
Universal TMs and encoding

- 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 ∈ $\{0,1\}^{n^2}$
- any reasonable encoding is OK
- emphasize "encoding of X" by writing <X>

10

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

11

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

12

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!

13

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.

14

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

15

Recursive Enumerability

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

Proof:

(⇐) 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.

16

Recursive Enumerability

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

Proof:

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

– let $s_1, s_2, s_3, \ldots$ be enumeration of Σ^* in lexicographic order.

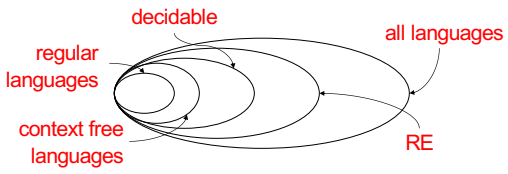
– for i = 1,2,3,4,…

- simulate M for i steps on $s_1, s_2, s_3, \ldots, s_i$

– if any simulation accepts, print out that s_j

17

Undecidability



decidable ⊆ RE ⊆ all languages

our goal: prove these containments proper

18

Countable and Uncountable Sets

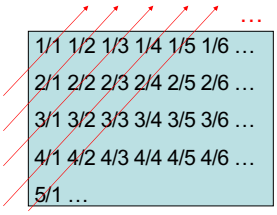
- the natural numbers **N** = {1,2,3,…} are countable
- Definition: a set S is countable if it is finite, or it is infinite and there is a bijection

  f: **N** → S

19

Countable and Uncountable Sets

- Theorem: the positive rational numbers Q = {m/n : m, n ∈ **N** } are countable.
- Proof:



20

Countable and Uncountable Sets

Theorem: the real numbers **R** are NOT countable (they are "uncountable").

- How do you prove such a statement?
  - assume countable (so there exists bijection f)
  - derive contradiction (some element not mapped to by f)
  - technique is called diagonalization (Cantor)

21

Countable and Uncountable Sets

- Proof:
  - suppose **R** is countable
  - list **R** according to the bijection f:

n	f(n)
1	3.14159…
2	5.55555…
3	0.12345…
4	0.50000…
…	

22

Countable and Uncountable Sets

- Proof:
  - suppose **R** is countable
  - list **R** according to the bijection f:

n	f(n)
1	3.14159…
2	5.55555…
3	0.12345…
4	0.50000…
…	

set x = $0.a_1a_2a_3a_4…$

where digit $a_i \neq i^{th}$ digit after decimal point of f(i) (not 0, 9)

e.g. x = 0.2312…

x cannot be in the list!

23

non-RE languages

Theorem: there exist languages that are not Recursively Enumerable.

Proof outline:

- the set of all TMs is countable
- the set of all languages is uncountable
- the function L:{TMs} →{languages} cannot be onto

24