

CS21 Decidability and Tractability

Lecture 10
January 28, 2009

January 28, 2009

CS21 Lecture 10

1

Outline

- nondeterministic TMs
- Church-Turing Thesis
- decidable, RE, co-RE languages

January 28, 2009

CS21 Lecture 10

2

Nondeterministic TMs

- A important variant: **nondeterministic TM**
- informally, several possible next configurations at each step
- formally, a NTM is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ where:
 - everything is the same as a TM except the transition function:

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

January 28, 2009

CS21 Lecture 10

3

NTM acceptance

- start configuration: $q_0 w$ (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, \dots, C_k
- C_1 is start configuration of M on input w
 - $C_i \Rightarrow C_{i+1}$ for $i = 1, 2, 3, \dots, k-1$
 - C_k is an accepting configuration

January 28, 2009

CS21 Lecture 10

4

Nondeterministic TMs

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

Proof:

- Idea: simulate NTM with a deterministic TM

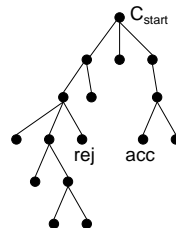
January 28, 2009

CS21 Lecture 10

5

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.

January 28, 2009

CS21 Lecture 10

6

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 any branch**: we will not halt...

January 28, 2009

CS21 Lecture 10

7

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 $\{1,2,\dots,b\}^*$)
- Initially, tape 1 has input, others blank
- **STEP 1**: copy tape 1 to tape 2

January 28, 2009

CS21 Lecture 10

8

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 $DONE = 1$

January 28, 2009

CS21 Lecture 10

9

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 $+$, $-$, $*$, $/$

January 28, 2009

CS21 Lecture 10

10

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

January 28, 2009

CS21 Lecture 10

11

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 $\langle X \rangle$

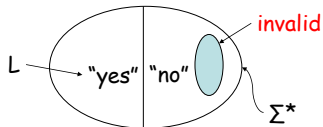
January 28, 2009

CS21 Lecture 10

12

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

January 28, 2009

CS21 Lecture 10

13

Universal TMs and encoding

- We can easily construct a **Universal TM** that recognizes the language:
 $A_{TM} = \{ \langle M, w \rangle : M \text{ is a TM and } M \text{ accepts } w \}$
 – 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

January 28, 2009

CS21 Lecture 10

14

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!

January 28, 2009

CS21 Lecture 10

15

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.

January 28, 2009

CS21 Lecture 10

16

Recursive Enumerability

- Why is “Turing-recognizable” called RE?
- Definition: a language $L \subset \Sigma^*$ is **recursively enumerable** if there exists a TM (an “enumerator”) that writes on its output tape
 $\#x_1\#x_2\#x_3\#\dots$
 and $L = \{x_1, x_2, x_3, \dots\}$.

- The output may be infinite

January 28, 2009

CS21 Lecture 10

17

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.

January 28, 2009

CS21 Lecture 10

18

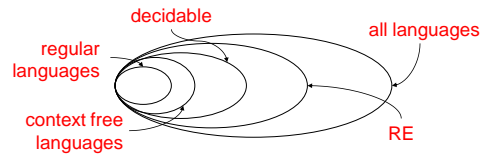
Recursive Enumerability

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

Proof:

- (\Rightarrow) Let M recognize language $L \subset \Sigma^*$.
- let s_1, s_2, s_3, \dots be enumeration of Σ^* in lexicographic order.
- for $i = 1, 2, 3, 4, \dots$
 - simulate M for i steps on $s_1, s_2, s_3, \dots, s_i$
- if any simulation accepts, print out that s_i

Undecidability



$\text{decidable} \subset \text{RE} \subset \text{all languages}$

our goal: prove these containments proper

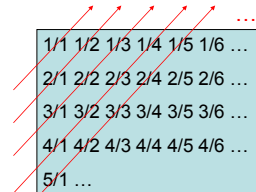
Countable and Uncountable Sets

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

$$f: \mathbf{N} \rightarrow S$$

Countable and Uncountable Sets

- Theorem: the positive rational numbers $\mathbf{Q} = \{m/n : m, n \in \mathbf{N}\}$ are countable.
- Proof:



Countable and Uncountable Sets

Theorem: the real numbers \mathbf{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)

Countable and Uncountable Sets

- Proof:
 - suppose \mathbf{R} is countable
 - list \mathbf{R} according to the bijection f :

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

Countable and Uncountable Sets

- Proof:

- suppose \mathbf{R} is countable

- list \mathbf{R} according to the bijection f :

n	$f(n)$	
1	3.14159...	set $x = 0.a_1a_2a_3a_4\dots$
2	5.55555...	where digit $a_i \neq i^{\text{th}}$ digit after decimal point of $f(i)$ (not 0, 9)
3	0.12345...	e.g. $x = 0.2312\dots$
4	0.50000...	x cannot be in the list!
...		

January 28, 2009

CS21 Lecture 10

25

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 $L:\{\text{TMs}\} \rightarrow \{\text{languages}\}$ cannot be onto

January 28, 2009

CS21 Lecture 10

26

non-RE languages

- Lemma: the set of all TMs is **countable**.

- Proof:

- each TM M can be described by a finite-length string $\langle M \rangle$

- can enumerate these strings, and give the natural bijection with \mathbf{N}

January 28, 2009

CS21 Lecture 10

27

non-RE languages

- Lemma: the set of all languages is **uncountable**

- Proof:

- fix an enumeration of all strings s_1, s_2, s_3, \dots (for example, lexicographic order)

- a language L is described by its **characteristic vector** χ_L whose i^{th} element is 0 if s_i is not in L and 1 if s_i is in L

January 28, 2009

CS21 Lecture 10

28

non-RE languages

- suppose the set of all languages is countable

- list characteristic vectors of all languages according to the bijection f :

n	$f(n)$
1	0101010...
2	1010011...
3	1110001...
4	0100011...
...	

January 28, 2009

CS21 Lecture 10

29

non-RE languages

- suppose the set of all languages is countable

- list characteristic vectors of all languages according to the bijection f :

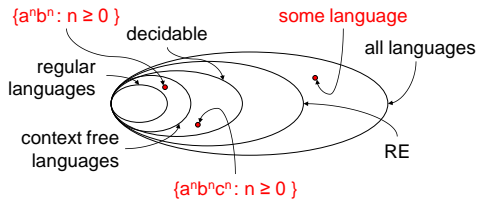
n	$f(n)$	
1	0101010...	set $x = 1101\dots$
2	1010011...	where i^{th} digit $\neq i^{\text{th}}$ digit of $f(i)$
3	1110001...	x cannot be in the list!
4	0100011...	therefore, the language with characteristic vector x is not in the list
...		

January 28, 2009

CS21 Lecture 10

30

So far...



- This language might be an esoteric, artificially constructed one. Do we care?
- We will show a natural undecidable L next.

January 28, 2009

CS21 Lecture 10

31

The Halting Problem

- Definition of the “Halting Problem”:
 $HALT = \{ \langle M, x \rangle : \text{TM } M \text{ halts on input } x \}$
- HALT is recursively enumerable.
 - proof?
- Is HALT decidable?

January 28, 2009

CS21 Lecture 10

32

The Halting Problem

Theorem: HALT is not decidable (undecidable).

Proof:

- Suppose TM H decides HALT
- Define new TM H': on input $\langle M \rangle$
 - if H accepts $\langle M, M \rangle$ then loop
 - if H rejects $\langle M, M \rangle$ then halt

January 28, 2009

CS21 Lecture 10

33

The Halting Problem

Proof:

- define new TM H': on input $\langle M \rangle$
 - if H accepts $\langle M, M \rangle$ then loop
 - if H rejects $\langle M, M \rangle$ then halt
- consider H' on input $\langle H' \rangle$:
 - if it halts, then H rejects $\langle H', H' \rangle$, which implies it cannot halt
 - if it loops, then H accepts $\langle H', H' \rangle$ which implies it must halt
- contradiction.

January 28, 2009

CS21 Lecture 10

34