Nondeterministic TMs

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

Proof:
- Idea: simulate NTM with a deterministic TM

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…

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 $\{1,2,\ldots,b\}^*$)
- 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 $+, -, *, /$

Universal TMs and encoding

- the input to a TM is always a string in $\Sigma^*$
- 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

Universal TMs and encoding

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

Universal TMs and encoding

- some strings not valid encodings and these are not in the language
  - $L \rightarrow \text{"yes"} \quad \text{invalid}$
  - $\Sigma^*$
  - make sure TM can recognize invalid encodings and reject them

Universal TMs and encoding

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

Note: this is a belief, not a theorem.

The Church-Turing Thesis

- the belief that TMs formalize our intuitive notion of an algorithm is:
  - everything we can compute on a physical computer can be computed on a Turing Machine

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

  $\#x_1\#x_2\#x_3\#…$

  and $L = \{x_1, x_2, x_3, \ldots\}$.

- The output may be infinite

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

Proof:

$\Rightarrow$ 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.

Recursive Enumerability

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

Proof:

$\Leftarrow$ Let $M$ recognize language $L \subseteq \Sigma^*$.
  - let $s_1, s_2, s_3, \ldots$ be enumeration of $\Sigma^*$ 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 $s_j$

Undecidability

decidable $\subseteq$ RE $\subseteq$ all languages

our goal: prove these containments proper
Countable and Uncountable Sets

- the natural numbers $\mathbb{N} = \{1, 2, 3, \ldots\}$ are **countable**

- Definition: a set $S$ is **countable** if it is finite, or it is infinite and there is a bijection $f: \mathbb{N} \rightarrow S$

---

Countable and Uncountable Sets

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

**Proof:**

\[
\begin{array}{cccccccc}
1/1 & 1/2 & 1/3 & 1/4 & 1/5 & 1/6 & \ldots \\
2/1 & 2/2 & 2/3 & 2/4 & 2/5 & 2/6 & \ldots \\
3/1 & 3/2 & 3/3 & 3/4 & 3/5 & 3/6 & \ldots \\
4/1 & 4/2 & 4/3 & 4/4 & 4/5 & 4/6 & \ldots \\
5/1 & \ldots \\
\end{array}
\]

---

Countable and Uncountable Sets

**Theorem:** the real numbers $\mathbb{R}$ 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)

**Proof:**

- suppose $\mathbb{R}$ is countable
- list $\mathbb{R}$ according to the bijection $f$:

\[
\begin{array}{ccc}
1 & 2 & 3 \\
3.14159\ldots & 5.55555\ldots & 0.12345\ldots \\
3.14159\ldots & 5.55555\ldots & 0.12345\ldots \\
3.14159\ldots & 5.55555\ldots & 0.12345\ldots \\
\ldots & \ldots & \ldots \\
\end{array}
\]

- set $x = 0.a_1a_2a_3a_4\ldots$
  - where digit $a_i$ \(\neq i^{th}\) digit after decimal point of $f(i)$ (not 0, 9)
  - e.g. $x = 0.2312\ldots$

- $x$ cannot be in the list!

---

Countable and Uncountable Sets

**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

---

non-RE languages

**Theorem:** there exist languages that are not Recursively Enumerable.