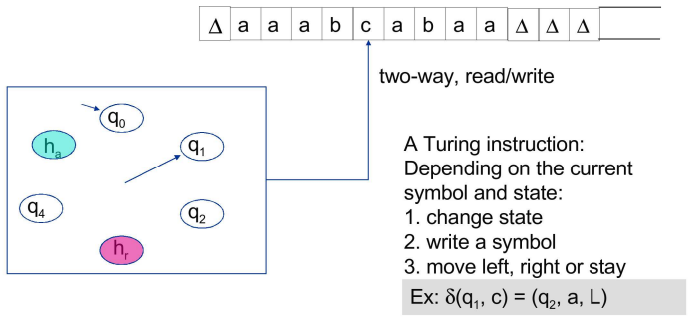
Turing machines – informally 1



Non-accepting runs of a Turing machine

- Four types of non-accepting runs from $(q_0, \underline{\Delta}x)$:
  - eventually enters h_r
  - crashing (1) - machine in a non-halt state, but no outgoing transition; convention: the machine then enters h_r
  - crashing (2) - head at the leftmost tape square, and step function tells machine to go left; convention: the machine then enters h_r
  - infinite loop - the machine never enters h_a or h_r

Program for today

- *Describe* fundamental computational classes
- *Describe* and *analyze* their properties
  - *supplement with intuition*
- *Prove* the existence of non-computable problems
  - diagonalization
- *Prove* concrete problems to be non-computable
  - reduction
  - *supplement formally*

Two new language classes

- **Definition 10.1**
  A language $L \subseteq \Sigma^*$ is said to be *recursively enumerable* (*semi-decidable*) iff it is accepted by some Turing machine
- **Lemma**
  A language $L \subseteq \Sigma^*$ is recursively enumerable iff it is accepted by a Turing machine, s.t.
  - if $x \in L$ then $(q_0, \underline{\Delta}x) \vdash^* (h_a, y\underline{a}z)$ for some $a \in (\Gamma \cup \{\Delta\})$ & $y, z \in (\Gamma \cup \{\Delta\})^*$
  - if $x \notin L$ then $(q_0, \underline{\Delta}x) \vdash^*$ loops

Two new language classes

- **Definition 10.1**
  A language $L \subseteq \Sigma^*$ is said to be *recursive* (*decidable*) iff there is a Turing machine computing its (total) characteristic function
  $\chi_L : \Sigma^* \rightarrow \{0, 1\}$ where $\chi_L(x) = 1$ if $x \in L$
  $\chi_L(x) = 0$ if $x \notin L$

- **Lemma**
  A language $L \subseteq \Sigma^*$ is recursive iff it is accepted by some *total* Turing machine, i.e.
  - if $x \in L$ then $(q_0, \underline{\Delta}x) \vdash^* (h_a, y\underline{a}z)$ for some $a \in (\Gamma \cup \{\Delta\})$ & $y, z \in (\Gamma \cup \{\Delta\})^*$
  - if $x \notin L$ then $(q_0, \underline{\Delta}x) \vdash^* (h_r, y\underline{a}z)$ for some $a \in (\Gamma \cup \{\Delta\})$ & $y, z \in (\Gamma \cup \{\Delta\})^*$

Some results

- **Theorem 10.1**
  Every recursive language is recursively enumerable

- **Theorem 10.3**
  The class of recursively enumerable languages is closed under union and intersection [Martin] - so is the class of recursive languages (Exercise!)

Some more results

- **Theorem 10.4**
  The class of recursive languages is closed under complement

- **Theorem 10.5**
  If L and its complement L' are both recursively enumerable, then L is recursive

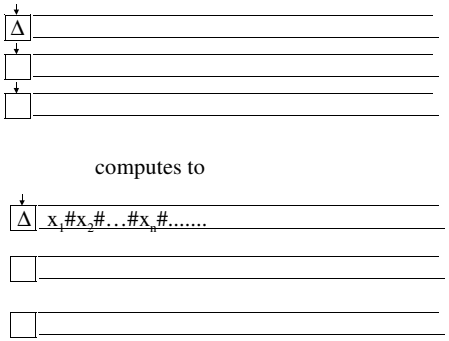
Enumerating a language

- **Definition 10.2**
  A language $L \subseteq \Sigma^*$ is said to be *enumerated* by a k-tape TM T iff
  - the tape head on the first tape never moves left, and no nonblank symbol printed is subsequently modified
  - the tape contents of tape 1 always has the form
    $x_1\#x_2\#\ldots\#x_n\#y$, where $x_1, x_2, \ldots, x_n \in L$
  - any $x \in L$ eventually appears on tape 1 between #'s

Enumerating a language

- **Theorem 10.6**
  A language $L \subseteq \Sigma^*$ is enumerated by a TM iff it is recursively enumerable

Enumerating a language



Enumerating a language

- **Theorem 10.6**
  A language $L \subseteq \Sigma^*$ is enumerated by a TM iff it is recursively enumerable
- **Definition** The *canonical ordering* of Σ^* is "first by length, secondly by lexicographic order" - example $\Sigma = \{a, b\}$:
  Δ, a, b, aa, ab, ba, bb, aaa,…, bbb, aaaa, aaab, ……
- **Theorem 10.6**
  A language $L \subseteq \Sigma^*$ is enumerated in canonical order by a TM iff it is recursive (Exercise!)

Countable sets

- **Definition 10.6**
  A set is said to be *countably infinite* iff there is a bijection from N to S.
  A set is said to be *countable* iff it is either finite or countably infinite
- **Lemma 10.2**
  Every subset of a countable set is countable (Exercise!)

Countable sets

- **Theorem 10.13**
  If for all natural numbers $i \geq 0$, S_i is a countable set,
  then $\cup_{i \geq 0} S_i$ is countable.

- Examples of countable sets
  - N - the natural numbers
  - N×N - pairs of natural numbers
  - Z - the integers
    Σ* - the set of all finite strings over a finite alphabet Σ
  - Any $L \subseteq \Sigma^*$

Uncountable sets

- **Theorem 10.14**
  The set of reals $[0, 1) = \{x \in R \mid 0 \leq x < 1\}$ is an uncountable set

- **Theorem 10.15**
  If S is a countably infinite set, then 2^S (the set of all subsets of S) is uncountable

Uncountable sets

- **Theorem 10.15 (cntd.)**
  For any nonempty alphabet Σ, the set of languages over Σ is uncountable

- **Theorem**
  For any nonempty alphabet Σ, the set of recursively enumerable languages over Σ is countable

- **Corollary 10.1**
  For any nonempty alphabet Σ, the set of languages over Σ which are NOT recursively enumerable is uncountable!

A specific language which is not recursively enumerable

- **Definition 11.1**
  $NSA = \{w \in \{0, 1\}^* \mid w = e(T) \text{ for some TM T, and } w \notin L(T)\}$
  $SA = \{w \in \{0, 1\}^* \mid w = e(T) \text{ for some TM T, and } w \in L(T)\}$

- **Theorem 11.1**
  NSA is NOT recursively enumerable

- **Theorem 11.2**
  SA IS recursively enumerable, but NOT recursive

Encoding of (T, x) in alphabet {0, 1}

- Assume states of T: $\{q_1, q_2, \ldots, q_m\}$
- Asume alphabet of T: $\{a_1, a_2, \ldots, a_n\}$

- $s(\Delta) = 0 \quad s(a_i) = 0^{i+1}$
- $s(h_a) = 0 \quad s(h_r) = 00 \quad s(q_i) = 0^{i+2}$
- $s(S) = 0 \quad s(L) = 00 \quad s(R) = 000$

Encoding of (T, x) in alphabet {0, 1}

- Each move m: $\delta(p, a) = (q, b, D)$ encoded by
  $e(m) = s(p)\ 1\ s(a)\ 1\ s(q)\ 1\ s(b)\ 1\ s(D)\ 1$
- Turing machine T with moves $\{m_1, m_2, .., m_k\}$encoded by
  $e(T) = 1\ s(q_0)\ 1\ e(m_1)\ 1\ e(m_2)\ 1 \ldots .\ e(m_k)\ 1$
- Input $x = a_1 a_2 \ldots a_l$ encoded by
  $e(x) = 1\ s(a_1)\ 1\ s(a_2)\ 1 \ldots s(a_l)\ 1$

A specific language which is not recursively enumerable

- **Definition 11.1**
  NSA = $\{w \in \{0, 1\}^* \mid w = e(T)$ for some TM T, and $w \notin L(T)\}$
  SA = $\{w \in \{0, 1\}^* \mid w = e(T)$ for some TM T, and $w \in L(T)\}$
- **Theorem 11.1**
  NSA is NOT recursively enumerable
- **Theorem 11.2**
  SA IS recursively enumerable, but NOT recursive
  Proof:
  NSA = SA' \ $\{w \in \{0, 1\}^* \mid w$ not on the form e(T) for some TM T$\}$

Decision Problems

- **Definition** Decision Problem (Martin)
  A *decision problem* P is a mapping from over a set of problem instances, I, to {yes, no}

  Definition Reasonable encoding
  e is a *reasonable encoding* of P iff
  - e is a one-to-one mapping from instances to strings over some alphabet Σ
  - an instance I can be decoded from e(I) (algorithmically!)
  - one can check (algorithmically!) whether a string $x \in \Sigma^*$ represents an instance

Decision Problems

- **Definition 11.2**
  If e is a reasonable encoding of a decision problem P over the alphabet Σ, then
  P is said to be *solvable* (or *decidable*) iff
  $Y(P) = \{e(I) \mid I \text{ is a yes-instance of } P\} \subseteq \Sigma^*$
  is recursive

Decision problems examples

- *Instances*: A text T
  Problem: is T a syntactically correct Java program? SOLVABLE!
- *Instances*: A graph G
- *Problem*: is G connected? SOLVABLE!
- *Instances*: A chess configuration C
- *Problem*: Is C winning for white? SOLVABLE!
- *Instances*: A syntactically correct Java program J and an input i
  Problem: Does J halt when run on input i?

Decision Problems

- **Observation**
  The encoding e of Turing machines over {0, 1} from the construction of the Universal Turing machine is reasonable
- **Definition** Self-accepting problem
  Instances: Turing Machines T (with encoding e)
  Problem: Does T accept e(T)?
- **Theorem 11.3**
  Self-accepting is unsolvable

An interesting unsolvable problem

- **Definition** Accepts
  Instances: A Turing Machine T and a string w
  Problem: is $w \in L(T)$?
- **Theorem 11.4**
  Accepts is unsolvable!

Reduction

- **Definition 11.3**
  Given two decision problems P_1 and P_2, P_1 is said to be *reducible* to P_2 ($P_1 \leq P_2$)
  iff
  there is an *algorithm* F, translating instances of P_1 to instances of P_2, such that for every instance I_1 of P_1, the answers to I_1 and $F(I_1)$ are the same

Reduction

- **Definition 11.3a**
  Given two languages $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$, L_1 is said to be *reducible* to L_2 ($L_1 \leq L_2$) iff

  there is a *Turing computable function* f: Σ_1^* -> Σ_2^*, such that for every instance $x \in \Sigma_1^*$, $x \in L_1$ iff $f(x) \in L_2$

Reduction

- **Theorem 11.4**
  Given two languages $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$, such that $L_1 \leq L_2$

- if L_2 is recursive, then so is L_1
- if L_2 is recursively enumerable, then so is L_1 *(Exercise!)*

- if L_1 is *not* recursive, then neither is L_2
- if L_1 is *not* recursively enumerable, then neither is L_2

Reduction - applications

- **Theorem 11.5**
  Acc = {e(T)e(w) | w ∈ L(T) } is not recursive
  Proof: Show SA ≤ Acc

- **Theorem 11.6**
  Halts = {e(T)e(w) | T halts on input w} is not recursive
  Proof: Show Acc ≤ Halts

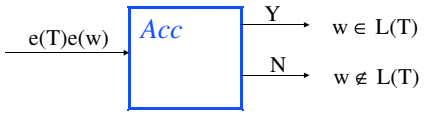
Reduction - applications

- **Theorem 11.5**
  Acc = {e(T)e(w) | w ∈ L(T) } is not recursive
  Proof: Show SA ≤ Acc

- **Theorem 11.6**
  Halts = {e(T)e(w) | T halts on input w} is not recursive
  Proof: Show Acc ≤ Halts

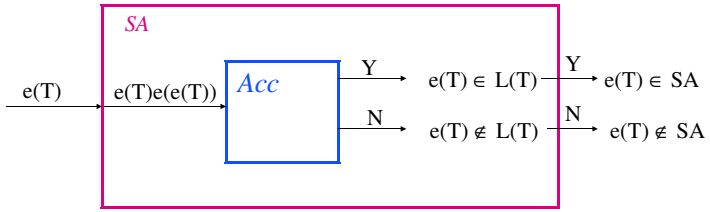
Reduction: *SA* ≤ *Acc*

- Assume you had TM accepting *Acc*



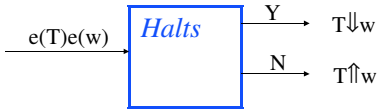
Reduction: *SA* ≤ *Acc*

- Construct TM accepting *SA*



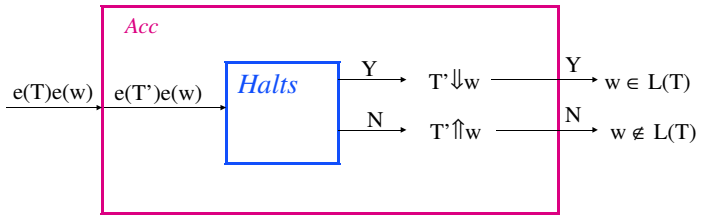
Reduction: *Acc* ≤ *Halts*

- Assume you had TM accepting *Halts*



Reduction: *Acc* ≤ *Halts*

- Construct TM accepting *Acc*



Where T' is T with all crashes and h_r replaced with divergence

Program for today

- *Describe* fundamental computational classes
- *Describe* and *analyze* their properties
- *Prove* the existence of non-computable problems
  - diagonalization
- *Prove* concrete problems to be non-computable
  - reduction

Exercises in [Martin]

- *Describe* properties of computability classes
  - 10.1: show closure properties of R
  - 10.3: RE closed under infinite union?
  - 10.5: show that R is the same as "TM *canonical* enumeration"
  - 10.35: RE closed under Kleene *
- *Describe* countability properties
  - 10.24: show that countability is closed under subsets
  - 10.27: L uncountable, M countable -> L\M uncountable
  - 10.29: alternative proof of countability of RE
  - 10.31 (b,d,e,h): (un)countable sets?
- *Describe* properties of non-computable problems
  - 10.33: uncountably many "difficult" languages!
- *Analyze* properties of non-computability
  - 10.39: Non-existence of virus-tester!!

Competition

- Construct a (deterministic) Turing Machine with
  - three states
  - a tape which is infinite "both ways"
  - tape-alphabet {1}

  such that when started on a blank tape, the Turing Machine halts and prints as many 1's as possible
- Do the same for four states
- Test your solution on the TM simulator from last week