6.045: Automata, Computability, and Complexity
Or, Great Ideas in Theoretical Computer Science
Spring, 2010

Class 9

Nancy Lynch

Today

- Mapping reducibility and Rice's Theorem
- We've seen several undecidability proofs.
- Today we'll extract some of the key ideas of those proofs and present them as general, abstract definitions and theorems.
- Two main ideas:
  - A formal definition of reducibility from one language to another. Captures many of the reduction arguments we have seen.
  - Rice's Theorem, a general theorem about undecidability of properties of Turing machine behavior (or program behavior).

Today

- Mapping reducibility and Rice's Theorem
- Topics:
  - Computable functions.
  - Mapping reducibility, $\leq_m$
  - Applications of $\leq_m$ to show undecidability and non-recognizability of languages.
  - Rice's Theorem
  - Applications of Rice's Theorem
- Reading:
  - Sipser Section 5.3, Problems 5.28-5.30.

Computable Functions

Computable Functions

- These are needed to define mapping reducibility, $\leq_m$.
- Definition: A function f: $\Sigma_1^* \rightarrow \Sigma_2^*$ is computable if there is a Turing machine (or program) such that, for every w in Σ_1^*, M on input w halts with just f(w) on its tape.
- To be definite, use basic TM model, except replace q_{acc} and q_{rej} states with one q_{halt} state.

- So far in this course, we've focused on accept/reject decisions, which let TMs decide language membership.
- That's the same as computing functions from Σ^* to { accept, reject }.
- Now generalize to compute functions that produce strings.

Total vs. partial computability

- We require f to be total = defined for every string.
- Could also define partial computable (= partial recursive) functions, which are defined on some subset of Σ_1*.
- Then M should not halt if f(w) is undefined.

Computable functions

- Example 1: Computing prime numbers.
  - f: $\{ 0, 1 \}^* \rightarrow \{ 0, 1 \}^*$
  - On input w that is a binary representation of positive integer i, result is the standard binary representation of the i^{th} prime number.
  - On inputs representing 0, result is the empty string ε.
    - Probably don't care what the result is in this case, but totality requires that we define something.
  - For instance:
    - $f(\varepsilon) = f(0) = f(00) = \varepsilon$
    - f(1) = f(01) = f(001) = 10 (binary rep of 2, first prime)
    - f(10) = f(010) = 11 (3, second prime)
    - f(11) = 101 (5, third prime)
    - f(100) = 111 (7, fourth prime)
  - Computable, e.g., by sieve algorithm.

Computable functions

- Example 2: Reverse machine.
  - f: { 0, 1 }* $\rightarrow$ { 0, 1 }*
  - On input w = < M >, where M is a (basic) Turing machine, f(w) = < M′ >, where M′ is a Turing machine that accepts exactly the reverses of the words accepted by M.
  - $L(M') = \{ w^R \mid w \in L(M) \}$
  - On inputs w that don't represent TMs, f(w) = ε.
  - Computable:
    - M′ reverses its input and then simulates M.
    - Can compute description of M′ from description of M.

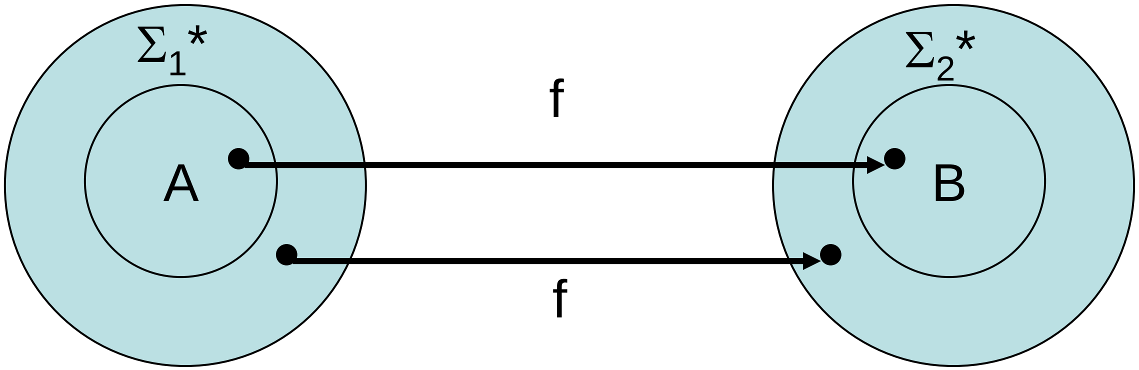
Computable functions

- Example 3: Transformations of DFAs, etc.
  - We studied several algorithmic transformations of DFAs and NFAs:
    - NFA $\rightarrow$ equivalent DFA
    - DFA for L $\rightarrow$ DFA for L^c
    - DFA for L $\rightarrow$ DFA for $\{ w^R \mid w \in L \}$
    - Etc.
  - All of these transformations can be formalized as computable functions (from machine representations to machine representations)

Mapping Reducibility

Mapping Reducibility

- Definition: Let $A \subseteq \Sigma_1^*$, $B \subseteq \Sigma_2^*$ be languages. Then A is mapping-reducible to B, $A \leq_m B$, provided that there is a computable function $f: \Sigma_1^* \rightarrow \Sigma_2^*$ such that, for every string w in Σ_1^*, $w \in A$ if and only if $f(w) \in B$.
- Two things to show for "if and only if":



- We've already seen many instance of $\leq_m$ in the reductions we've used to prove undecidability and non-recognizability, e.g.:

Mapping reducibility examples

- Example: $Acc_{TM} \leq_m Acc01_{TM}$

Accepts the string 01, possibly others

- $<M, w> \rightarrow <M'_{M,w}>$, by computable function f.
- $M'_{M,w}$ behaves as follows: If M accepts w then it accepts everything; otherwise it accepts nothing.
- This f demonstrates mapping reducibility because:
  - If $<M, w> \in Acc_{TM}$ then $<M'_{M,w}> \in Acc01_{TM}$.
  - If $<M, w> \notin Acc_{TM}$ then $<M'_{M,w}> \notin Acc01_{TM}$.
  - Thus, we have “if and only if”, as needed.
  - And f is computable.
- Technicality: Must also map inputs not of the form <M, w> somewhere.

Mapping reducibility examples

- Example: $Acc_{TM} \leq_m (E_{TM})^c$

Nonemptiness, { M | M accepts some string}

- <M, w> $\rightarrow$ <$M'_{M,w}$>, by computable function f.
- Use same f as before: If M accepts w then $M'_{M,w}$ accepts everything; otherwise it accepts nothing.
- But now we must show something different:
  - If <M, w> $\in Acc_{TM}$ then <$M'_{M,w}$> $\in (E_{TM})^c$.
    - Accepts something, in fact, accepts everything.
  - If <M, w> $\notin Acc_{TM}$ then <$M'_{M,w}$> $\in E_{TM}$.
    - Accepts nothing.
  - f is computable.
- Note: We didn't show $Acc_{TM} \leq_m E_{TM}$.
  - Reversed the sense of the answer (took the complement).

Mapping reducibility examples

- Example: $Acc_{TM} \leq_m REG_{TM}$.

TMs accepting a regular language

- <M, w> $\rightarrow$ <$M'_{M,w}$>, by computable function f.
- We defined f so that: If M accepts w then $M'_{M,w}$ accepts everything; otherwise it accepts exactly the strings of the form 0^n1^n, $n \geq 0$.
- So <M, w> $\in Acc_{TM}$

  iff $M'_{M,w}$ accepts a regular language

  iff <$M'_{M,w}$> $\in REG_{TM}$.

Mapping reducibility examples

- Example: $Acc_{TM} \leq_m MPCP$.

Modified Post Correspondence Problem

- $<M, w> \rightarrow <T_{M,w}, t_{M,w}>$, by computable function f, where $<T_{M,w}, t_{M,w}>$ is an instance of MPCP (set of tiles + distinguished tile).
- We defined f so that $<M, w> \in Acc_{TM}$

  iff $T_{M,w}$ has a match starting with $t_{M,w}$

  iff $<T_{M,w}, t_{M,w}> \in MPCP$

- Example: $Acc_{TM} \leq_m PCP$.
- $<M, w> \rightarrow < T_{M,w}>$ where $<M, w> \in Acc_{TM}$ iff $T_{M,w}$ has a match iff $< T_{M,w}> \in PCP$.

Basic Theorems about $\leq_m$

- Theorem 1: If $A \leq_m B$ and B is Turing-decidable then A is Turing-decidable.
- Proof:
  - To decide if $w \in A$:
    - Compute f(w)
      - Can be done by a TM, since f is computable.
    - Decide whether $f(w) \in B$.
      - Can be done by a TM, since B is decidable.
    - Output the answer.
- Corollary 2: If $A \leq_m B$ and A is undecidable then B is undecidable.
- So undecidability of Acc_{TM} implies undecidability of E_{TM}, REG_{TM}, MPCP, etc.

Basic Theorems about $\leq_m$

- Theorem 3: If $A \leq_m B$ and B is Turing-recognizable then A is Turing-recognizable.
- Proof: On input w:
  - Compute f(w).
  - Run a TM that recognizes B on input f(w).
  - If this TM ever accepts, accept.
- Corollary 4: If $A \leq_m B$ and A is not Turing-recognizable then B is not Turing-recognizable.
- Theorem 5: $A \leq_m B$ if and only if $A^c \leq_m B^c$.
- Proof: Use same f.
- Theorem 6: If $A \leq_m B$ and $B \leq_m C$ then $A \leq_m C$.
- Proof: Compose the two functions.

Basic Theorems about $\leq_m$

- Theorem 6: If $A \leq_m B$ and $B \leq_m C$ then $A \leq_m C$.
- Example: PCP
  - Showed $Acc_{TM} \leq_m MPCP$.
  - Showed $MPCP \leq_m PCP$.
  - Conclude from Theorem 6 that $Acc_{TM} \leq_m PCP$.

More Applications of Mapping Reducibility

Applications of $\leq_m$

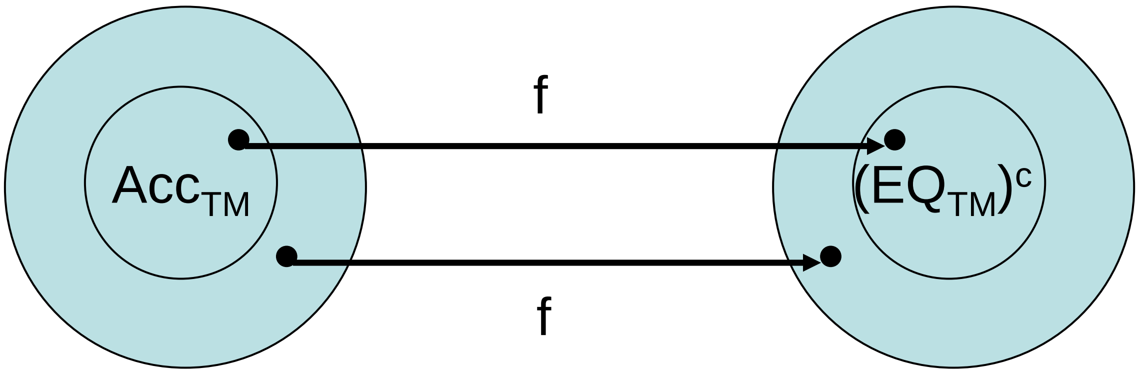
- We have already used $\leq_m$ to show undecidability; now use it to show non-Turing-recognizability.
- Example: $Acc01_{TM}$
  - We already know that $Acc01_{TM}$ is Turing-recognizable.
  - Now show that $(Acc01_{TM})^c$ is not Turing-recognizable.
  - We showed that $Acc_{TM} \leq_m Acc01_{TM}$.
  - So $(Acc_{TM})^c \leq_m (Acc01_{TM})^c$, by Theorem 5.
  - We also already know that $(Acc_{TM})^c$ is not Turing recognizable.
  - So $(Acc01_{TM})^c$ is not Turing-recognizable, by Corollary 4.

Applications of $\leq_m$

- Now an example of a language that is not Turing-recognizable and whose complement is also not Turing-recognizable.
- That is, it's neither Turing-recognizable nor co-Turing-recognizable.
- Example: $EQ_{TM} = \{ < M_1, M_2 > \mid M_1$ and M_2 are TMs and $L(M_1) = L(M_2) \}$
  - Important in practice, e.g.:
    - Compare two versions of the "same" program.
    - Compare the result of a compiler optimization to the original un-optimized compiler output.
- Theorem 7: EQ_{TM} is not Turing-recognizable.
- Theorem 8: $(EQ_{TM})^c$ is not Turing-recognizable.

Applications of $\leq_m$

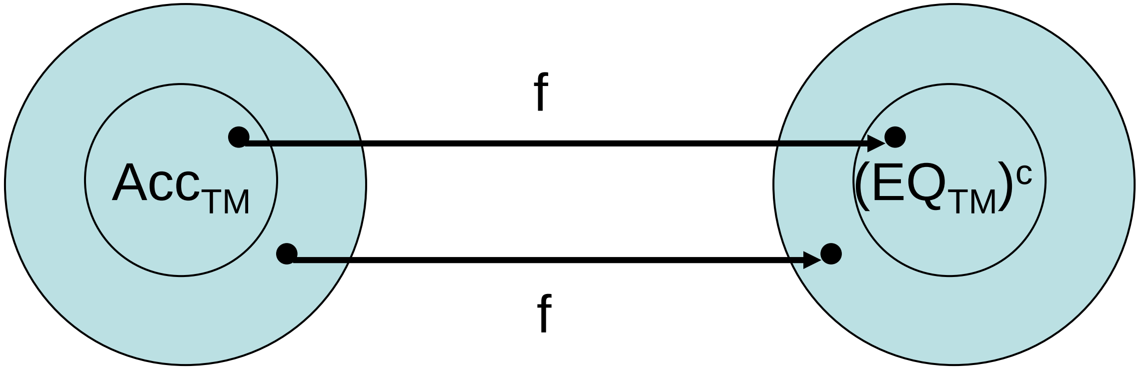
- $EQ_{TM} = \{ < M_1, M_2 > \mid L(M_1) = L(M_2) \}$
- Theorem 7: EQ_{TM} is not Turing-recognizable.
- Proof:
  - Show $(Acc_{TM})^c \leq_m EQ_{TM}$ and use Corollary 4.
    - Already showed $(Acc_{TM})^c$ is not Turing-recognizable.
  - Equivalently, show $Acc_{TM} \leq_m (EQ_{TM})^c$.
    - Equivalent by Theorem 5.
  - Need:



  - Accepting iff not equivalent.

EQ_{TM} is not Turing-recognizable.

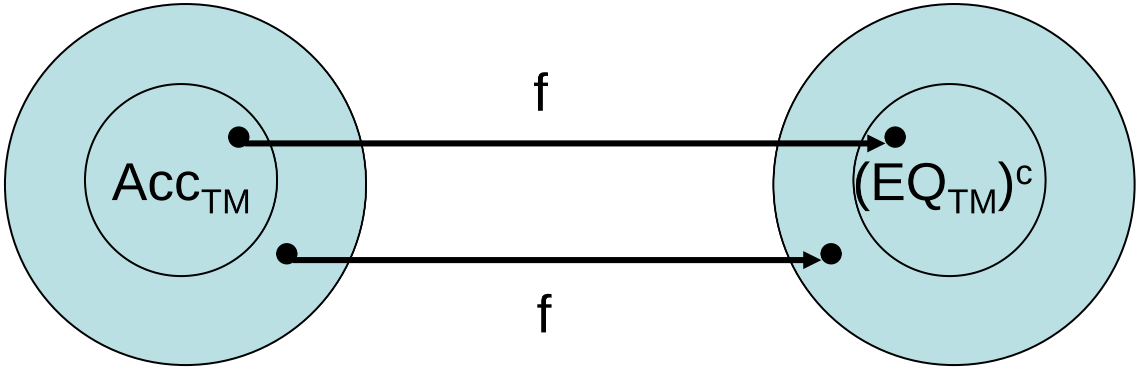
- $Acc_{TM} \leq_m (EQ_{TM})^c$:



- Define f(x) so that $x \in Acc_{TM}$ iff $f(x) \in (EQ_{TM})^c$.
- If x is not of the form <M, w> define f(x) = $<M_0, M_0>$, where M_0 is any particular TM.
- Then $x \notin Acc_{TM}$ and $f(x) \in EQ_{TM}$, which fits our requirements.
- So now assume that x = <M, w>.
- Then define f(x) = $<M_1, M_2>$, where:
  - M_1 always rejects, and
  - M_2 ignores its input, runs M on w, and accepts iff M accepts w.
- Claim: $x \in Acc_{TM}$ iff $f(x) \in (EQ_{TM})^c$.

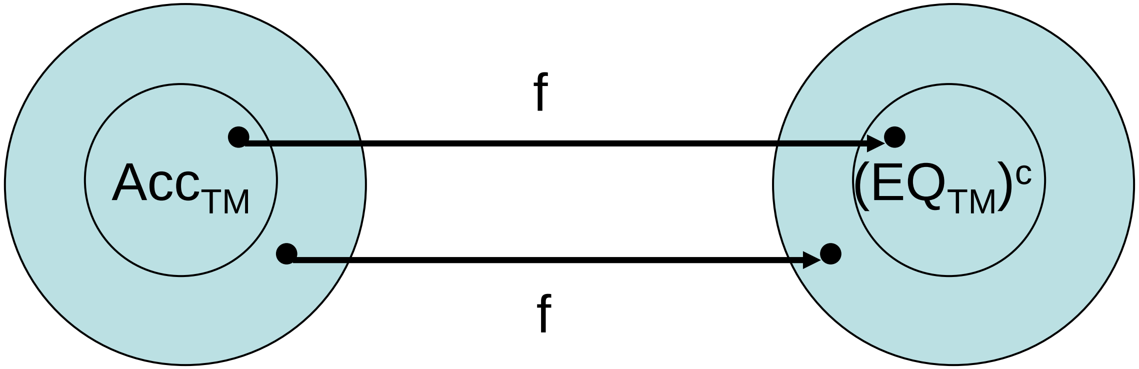
EQ_{TM} is not Turing-recognizable.

- $Acc_{TM} \leq_m (EQ_{TM})^c$:



- Assume $x = <M, w>$, define $f(x) = <M_1, M_2>$, where:
  - M_1 always rejects, and
  - M_2 ignores its input, runs M on w, and accepts iff M accepts w.
- Claim: $x \in Acc_{TM}$ iff $f(x) \in (EQ_{TM})^c$.
- Proof:
  - If $x \in Acc_{TM}$, then M accepts w, so M_2 accepts everything, so $<M_1, M_2> \notin EQ_{TM}$, so $<M_1, M_2> \in (EQ_{TM})^c$.
  - If $x \notin Acc_{TM}$, then M does not accept w, so M_2 accepts nothing, so $<M_1, M_2> \in EQ_{TM}$, so $<M_1, M_2> \notin (EQ_{TM})^c$.

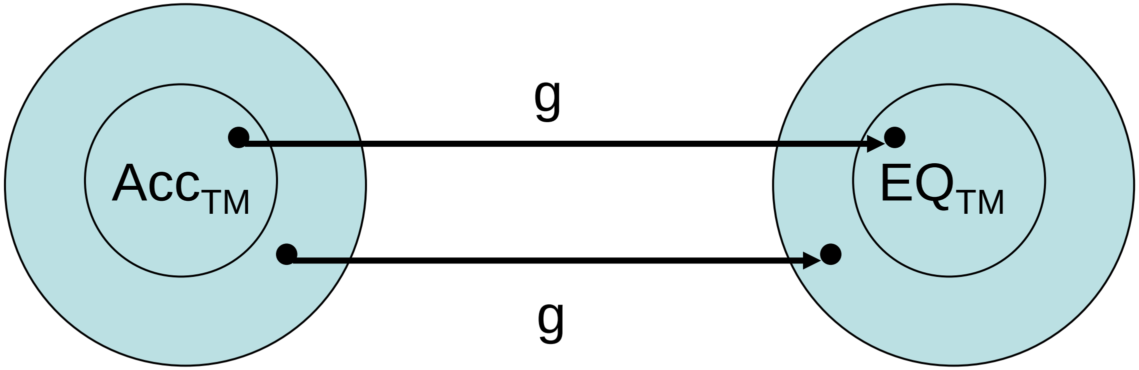
EQ_{TM} is not Turing-recognizable.



- Assume $x = \langle M, w\rangle$, define $f(x) = \langle M_1, M_2\rangle$, where:
  - M_1 always rejects, and
  - M_2 ignores its input, runs M on w, and accepts iff M accepts w.
- Claim: $x \in Acc_{TM}$ iff $f(x) \in (EQ_{TM})^c$.
- Therefore, $Acc_{TM} \leq_m (EQ_{TM})^c$ using f.
- So $(Acc_{TM})^c \leq_m EQ_{TM}$ by Theorem 5.
- So EQ_{TM} is not Turing-recognizable, by Corollary 4.

Applications of $\leq_m$

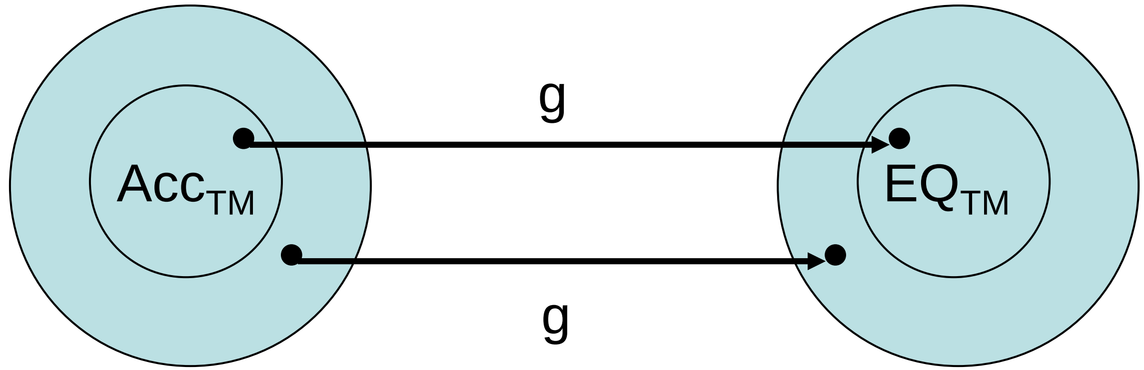
- We have proved:
- Theorem 7: EQ_{TM} is not Turing-recognizable.
- It turns out that the complement isn't T-recognizable either!
- Theorem 8: $(EQ_{TM})^c$ is not Turing-recognizable.
- Proof: Show $(Acc_{TM})^c \leq_m (EQ_{TM})^c$ and use Corollary 4.
    - We know $(Acc_{TM})^c$ is not Turing-recognizable.
  - Equivalently, show $Acc_{TM} \leq_m EQ_{TM}$.
  - Need:



  - Accepting iff equivalent.

$(EQ_{TM})^c$ is not Turing-recognizable.

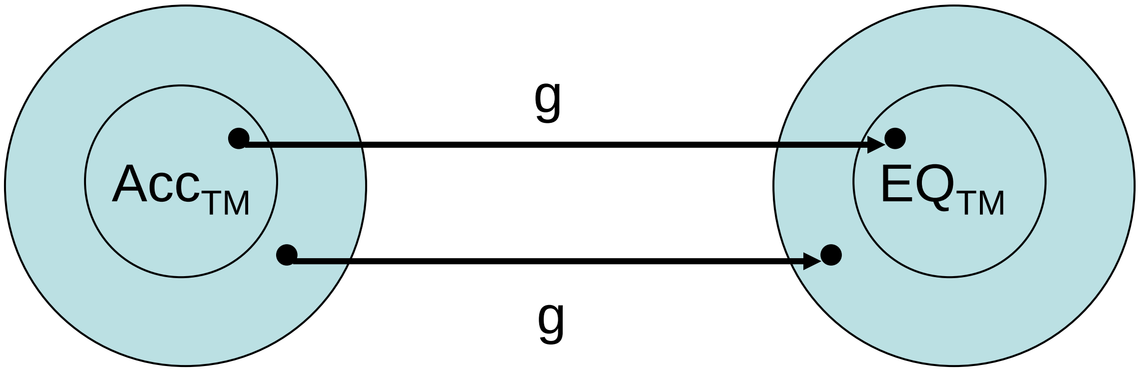
- $Acc_{TM} \leq_m EQ_{TM}$:



- Define g(x) so that $x \in Acc_{TM}$ iff $f(x) \in EQ_{TM}$.
- If x is not of the form <M, w> define $f(x) = <M_0, M_0'>$, where $L(M_0) \neq L(M_0')$.
- Then $x \notin Acc_{TM}$ and $g(x) \notin EQ_{TM}$, as required.
- So now assume x = <M, w>.
- Define $g(x) = <M_1, M_2>$, where:
  - M_1 accepts everything, and
  - M_2 ignores its input, runs M on w, accepts iff M does (as before).
- Claim: $x \in Acc_{TM}$ iff $g(x) \in EQ_{TM}$.

$(EQ_{TM})^c$ is not Turing-recognizable.

- $Acc_{TM} \leq_m EQ_{TM}$:



- Assume $x = \langle M, w \rangle$, define $g(x) = \langle M_1, M_2 \rangle$, where:
  - M_1 accepts everything, and
  - M_2 ignores its input, runs M on w, and accepts iff M does.
- Claim: $x \in Acc_{TM}$ iff $g(x) \in EQ_{TM}$.
- Proof:
  - If $x \in Acc_{TM}$, then M_1 and M_2 both accept everything, so $\langle M_1, M_2 \rangle \in EQ_{TM}$.
  - If $x \notin Acc_{TM}$, then M_1 accepts everything and M_2 accepts nothing, so $\langle M_1, M_2 \rangle \notin EQ_{TM}$.

$(EQ_{TM})^c$ is not Turing-recognizable.

- Assume $x = \langle M, w\rangle$, define $g(x) = \langle M_1, M_2\rangle$, where:
  - M_1 accepts everything, and
  - M_2 ignores its input, runs M on w, and accepts iff M does.
- Claim: $x \in Acc_{TM}$ iff $g(x) \in EQ_{TM}$.
- Therefore, $Acc_{TM} \leq_m EQ_{TM}$ using g.
- So $(Acc_{TM})^c \leq_m (EQ_{TM})^c$ by Theorem 5.
- So $(EQ_{TM})^c$ is not Turing-recognizable, by Corollary 4.

Rice’s Theorem

Rice's Theorem

- We've seen many undecidability results for properties of TMs, e.g., for:
  - $Acc01_{TM} = \{ < M > \mid 01 \in L(M) \}$
  - $E_{TM} = \{ < M > \mid L(M) = \varnothing \}$
  - $REG_{TM} = \{ < M > \mid L(M) \text{ is a regular language } \}$
- These are all properties of the language recognized by the machine.
- Contrast with:
  - $\{ < M > \mid M$ never tries to move left off the left end of the tape $\}$
  - $\{ < M > \mid M$ has more than 20 states $\}$
- Rice's Theorem says (essentially) that any property of the language recognized by a TM is undecidable.
- Very powerful theorem.
- Covers many problems besides the ones above, e.g.:
  - $\{ < M > \mid L(M)$ is a finite set $\}$
  - $\{ < M > \mid L(M)$ contains some palindrome $\}$
  - …

Rice's Theorem

- Rice's Theorem says (essentially) that any property of the language recognized by a TM is undecidable.
- Technicality: Restrict to nontrivial properties.
- Define a set P of languages, to be a nontrivial property of Turing-recognizable languages provided that
  – There is some TM M_1 such that $L(M_1) \in P$, and
  – There is some TM M_2 such that $L(M_2) \notin P$.
- Equivalently:
  – There is some Turing-recognizable language L_1 in P, and
  – There is some Turing recognizable language L_2 not in P.

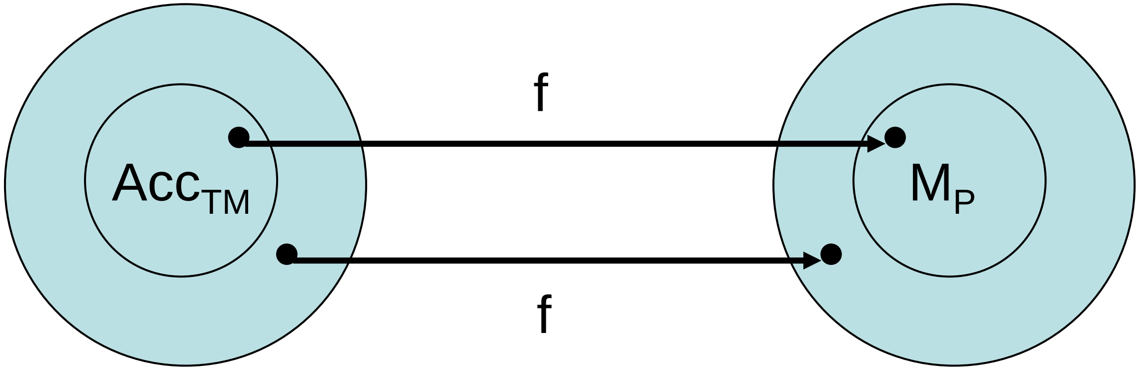
- Rice's Theorem: Let P be a nontrivial property of Turing-recognizable languages. Let $M_P = \{ < M > \mid L(M) \in P \}$. Then M_P is undecidable.
- !

Rice's Theorem

- P is a nontrivial property of T-recog. languages if:
  - There is some TM M_1 such that $L(M_1) \in P$, and
  - There is some TM M_2 such that $L(M_2) \notin P$.
- Rice's Theorem: Let P be a nontrivial property of Turing-recognizable languages. Let $M_P = \{ < M > \mid L(M) \in P \}$. Then M_P is undecidable.
- Proof:
  - Show $Acc_{TM} \leq_m M_P$.
  - Suppose WLOG that the empty language does not satisfy P, that is, $\emptyset \notin P$.
  - Why is this WLOG?
    - Otherwise, work with P^c instead of P.
    - Then $\emptyset \notin P^c$, continue the proof using P^c.
    - Conclude that M_{P^c} is undecidable.
    - Implies that M_P is undecidable.

Rice's Theorem

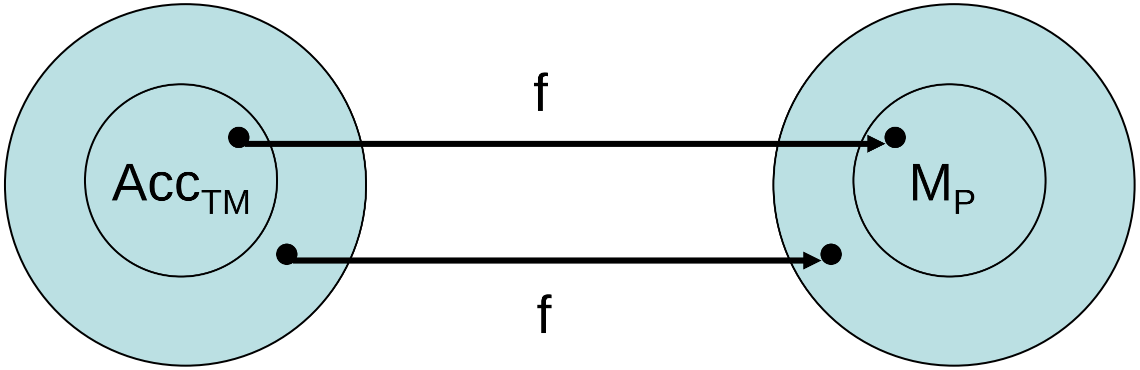
- Rice's Theorem: Let P be a nontrivial property of Turing-recognizable languages. Let $M_P = \{ < M > \mid L(M) \in P \}$. Then M_P is undecidable.
- Proof:
  - Show $Acc_{TM} \leq_m M_P$.
  - Suppose $\emptyset \notin P$.
  - Need:



  - Let M_1 be any TM such that $L(M_1) \in P$, so $< M_1 > \in M_P$.
    - How do we know such M_1 exists?
    - Because P is nontrivial.

Rice's Theorem

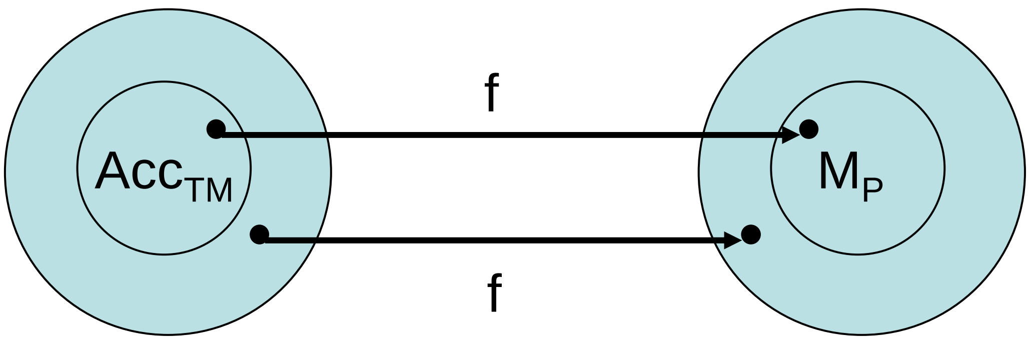
- Rice's Theorem: Let P be a nontrivial property of Turing-recognizable languages. Let $M_P = \{ < M > | L(M) \in P \}$. Then M_P is undecidable.
- Proof:
  - Show $Acc_{TM} \leq_m M_P$.
  - Suppose $\emptyset \notin P$.
  - Need:



  - Let M_1 be any TM such that $L(M_1) \in P$, so $< M_1 > \in M_P$.
  - Let M_2 be any TM such that $L(M_2) = \emptyset$, so $< M_2 > \notin M_P$.

Rice's Theorem

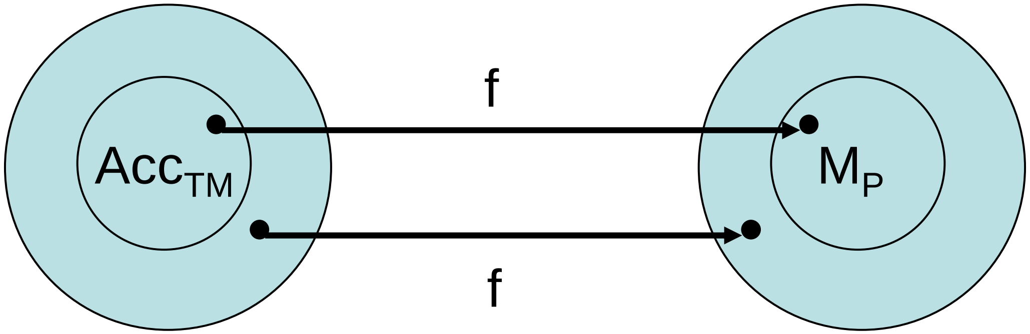
- Rice's Theorem: Let P be a nontrivial property. Then $M_P = \{ < M > \mid L(M) \in P \}$ is undecidable.
- Proof:
  - Need:



  - Let M_1 be any TM such that $L(M_1) \in P$, so $< M_1 > \in M_P$.
  - Let M_2 be any TM such that $L(M_2) = \emptyset$, so $< M_2 > \notin M_P$.
  - Define f(x):
    - If x isn't of the form <M, w>, return something $\notin M_P$, like $< M_2 >$.
    - If x = <M, w>, then f(x) = $< M'_{M,w} >$, where:
      - $M'_{M,w}$: On input y:
        - …

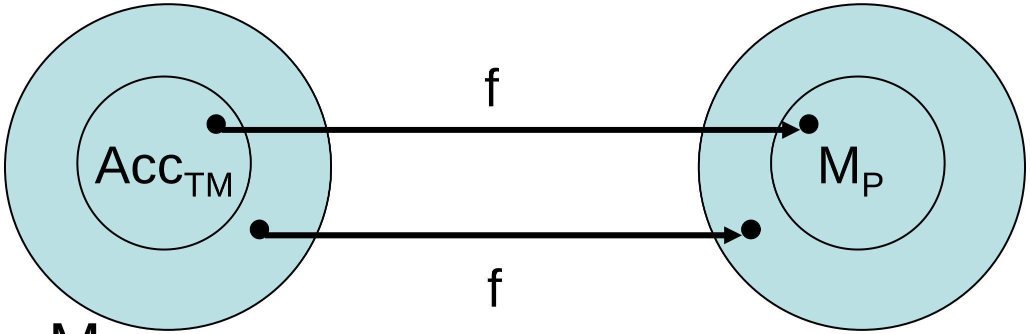
Rice's Theorem

- Proof:
  - Show $Acc_{TM} \leq_m M_P$.



  - $L(M_1) \in P$, so $< M_1 > \in M_P$.
  - $L(M_2) = \emptyset$, so $< M_2 > \notin M_P$.
  - Define f(x):
    - If x = <M, w>, then f(x) = < $M'_{M,w}$ >, where:
      - $M'_{M,w}$: On input y:
        - Run M on w.
        - If M accepts w then run M_1 on y, accept if M_1 accepts y.
        - (If M doesn't accept w or M_1 doesn't accept y, loop forever.)
    - Tricky…

Rice's Theorem



- Proof:
  - Show $Acc_{TM} \leq_m M_P$.
  - $L(M_1) \in P$, so $< M_1 > \in M_P$.
  - $L(M_2) = \emptyset$, so $< M_2 > \notin M_P$.
  - If $x = <M, w>$, then $f(x) = < M'_{M,w} >$, where:
    - $M'_{M,w}$: On input y:
      - Run M on w.
      - If M accepts w then run M_1 on y and accept if M_1 accepts y.
  - Claim $x \in Acc_{TM}$ if and only if $f(x) \in M_P$.
    - If $x = <M, w> \in Acc_{TM}$ then $L(M'_{M,w}) = L(M_1) \in P$, so $f(x) \in M_P$.
    - If $x = <M, w> \notin Acc_{TM}$ then $L(M'_{M,w}) = \emptyset \notin P$, so $f(x) \notin M_P$.
  - Therefore, $Acc_{TM} \leq_m M_P$ using f.
  - So M_P is undecidable, by Corollary 2.

Rice's Theorem

- We have proved:
- Rice's Theorem: Let P be a nontrivial property of Turing-recognizable languages. Let $M_P = \{ < M > \mid L(M) \in P \}$. Then M_P is undecidable.
- Note:
  - Rice proves undecidability, doesn't prove non-Turing-recognizability.
  - The sets M_P may be Turing-recognizable.
- Example: P = languages that contain 01
  - Then $M_P = \{ < M > \mid 01 \in L(M) \} = Acc01_{TM}$.
  - Rice implies that M_P is undecidable.
  - But we already know that $M_P = Acc01_{TM}$ is Turing-recognizable.
    - For a given input < M >, a TM/program can simulate M on 01 and accept iff this simulation accepts.

More Applications of Rice's Theorem

Applications of Rice's Theorem

- Example 1: Using Rice
  - { < M > | M is a TM that accepts at least 37 different strings }
  - Rice implies that this is undecidable.
  - This set = M_P, where P = "the language contains at least 37 different strings"
  - P is a language property.
  - Nontrivial, since some TM-recognizable languages satisfy it and some don't.

Applications of Rice’s Theorem

- Example 2: Property that isn’t a language property and is decidable
  – { < M > | M is a TM that has at least 37 states }
  – Not a language property, but a property of a machine’s structure.
  – So Rice doesn’t apply.
  – Obviously decidable, since we can determine the number of states given the TM description.

Applications of Rice’s Theorem

- Example 3: Another property that isn’t a language property and is decidable
  – { < M > | M is a TM that runs for at most 37 steps on input 01 }
  – Not a language property, not a property of a machine’s structure.
  – Rice doesn’t apply.
  – Obviously decidable, since, given the TM description, we can just simulate it for 37 steps.

Applications of Rice's Theorem

- Example 4: Undecidable property for which Rice's Theorem doesn't work to prove undecidability
  - Acc01SQ = { < M > | M is a TM that accepts the string 01 in exactly a perfect square number of steps }
  - Not a language property, Rice doesn't apply.
  - Can prove undecidable by showing $Acc01_{TM} \leq_m$ Acc01SQ.
    - $Acc01_{TM}$ is the set of TMs that accept 01 in any number of steps.
    - $Acc01SQ_{TM}$ is the set of TMs that accept 01 in a perfect square number of steps.
  - Design mapping f so that M accepts 01 iff f(M) = < M′ > where M′ accepts 01 in a perfect square number of steps.
  - f(<M>) = < M′ > where…

Applications of Rice's Theorem

- Example 4: Undecidable property for which Rice doesn't work to prove undecidability
  - Acc01SQ = { < M > | M is a TM that accepts the string 01 in exactly a perfect square number of steps }
  - Show $Acc01_{TM} \leq_m Acc01SQ$.
  - Design f so M accepts 01 iff f(M) = < M′ > where M′ accepts 01 in a perfect square number of steps.
  - f(<M>) = < M′ > where:
    - M′: On input x:
      - If $x \neq 01$, then reject.
      - If x = 01, then simulate M on 01. If M accepts 01, then accept, but just after doing enough extra steps to ensure that the total number of steps is a perfect square.
  - $\langle M \rangle \in Acc01_{TM}$ iff M′ accepts 01 in a perfect square number of steps, iff $f(\langle M \rangle) \in Acc01SQ$.
  - So $Acc01_{TM} \leq_m Acc01SQ$, so Acc01SQ is undecidable.

Applications of Rice's Theorem

- Example 5: Trivial language property
  - { < M > | M is a TM and L(M) is recognized by some TM having an even number of states }
  - This is a language property.
  - So it might seem that Rice should apply…
  - But, it's a trivial language property: Every Turing-recognizable language is recognized by some TM having an even number of states.
    - Could always add an extra, unreachable state.
  - Decidable or undecidable?
  - Decidable (of course), since it's the set of all TMs.

Applications of Rice’s Theorem

- Example 6:
  – { < M > | M is a TM and L(M) is recognized by some TM having at most 37 states and at most 37 tape symbols }
  – A language property.
  – Is it nontrivial?
  – Yes, some languages satisfy it and some don’t.
  – So Rice applies, showing that it’s undecidable.
  – Note: This isn’t { < M > | M is a TM that has at most 37 states and at most 37 tape symbols }
    - That’s decidable.
  – What about { < M > | M is a TM and L(M) is recognized by some TM having at least 37 states and at least 37 tape symbols }?
    - Trivial---all Turing-recognizable languages are recognized by some such machine.

Next time…

- The Recursion Theorem
- Reading:
  - Sipser Section 6.1

MIT OpenCourseWare
http://ocw.mit.edu

6.045J / 18.400J Automata, Computability, and Complexity
Spring 2011

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.