Reductions

2/29/2015

Pete Manolios

Theory of Computation

A_{TM} is Undecidable

- A_{TM} = {<M,w>: M is a TM that accepts w}
- Theorem: A_{TM} is Undecidable
- Proof: Suppose there exists a TM H that decides A_{TM}. Then, for any input <M,w>, H accepts if M accepts w and rejects otherwise.
- Derive contradiction using diagonalization

Diagonalization!

		<M1>	<M2>	<M3>	<M4>	<M5>	…
	M1	acc	rej	acc	acc	rej	
	M2	acc	acc	acc	rej	rej	
H	**M3**	rej	acc	rej	rej	rej	
	M4	rej	rej	acc	rej	rej	
	M5	acc	rej	rej	acc	acc	
	…						

H accepts {<M, <M>> : M accepts <M>}

Diagonalization!

H

	<M1>	<M2>	<M3>	<M4>	<M5>	…	<D>
M1	**acc**	rej	acc	acc	rej		
M2	acc	**acc**	acc	rej	rej		
M3	rej	acc	**rej**	rej	rej		
M4	rej	rej	acc	**rej**	rej		
M5	acc	rej	rej	acc	**acc**		
…							
D	**rej**	**rej**	**acc**	**acc**	**rej**		**???**

H accepts {<M, <M>> : M accepts <M>}

Diagonalization: Let D be at TM that negates diagonal

D is a TM: Call H on <M, <M>> and negate, so on list

But D is different, by construction, from all Mi. ⇲

A_{TM} is Undecidable

- Theorem: A_{TM} is Undecidable. (A_{TM} = {<M,w>: M is a TM that accepts w})
- Proof: Suppose there exists a TM H that decides A_{TM}. Then, for any input <M,w>, H accepts if M accepts w and rejects otherwise.
- Consider a TM D that takes an input <M>, the description of M, and takes the following steps.
  - Run H on <M,<M>>
  - If H accepts, reject
  - If H rejects, accept
- Since H is a decider, D is also a decider.
- D on <D> = accept

  iff {def. D} H <D, <D>> = reject

  iff {def. H} D on <D> = reject (Go both directions!) ⇲

Reducibility

- We showed the undecidability of $HALT_{TM}$ by reducing A_{TM} to $HALT_{TM}$
- We write $A_{TM} \leq_M HALT_{TM}$
- This is read as "A_{TM} is mapping reducible to $HALT_{TM}$"
- If $A \leq_M B$ that means there is a *computable* function $f: \Sigma^* \rightarrow \Sigma^*$ s.t. for all w
  - $w \in A$ iff $f(w) \in B$
  - f is a *reduction* from A to B
- A function is computable if some TM, on every input w halts with f(w) on tape

Reducibility

- Theorem: If A $\leq_M$ B and B is decidable, then A is decidable
- Proof: Let M be a decider for B and f the reduction from A to B. Here is a decider, N, for A
  - Given w, compute f(w)
  - Run M on f(w), returning same output
- Why doesn’t the other direction work?
- Corollary: If A $\leq_M$ B and A is undecidable, then B is undecidable. Proof?
- Our proof of undecidability of $HALT_{TM}$ was essentially based on this corollary.
- Mapping reducibility version: f is defined by TM F: On input <M,w>
  - Construct M’: Given x: Run M on x. If M accepts, accept else loop
  - Output <M’,w>
- Note: <M, w> $\in A_{TM}$ iff f(<M,w>)(=<M’,w>) $\in HALT_{TM}$
- Theorem: If A $\leq_M$ B and B is R.E., then A is R.E. (Same proof as above)
- Corollary: If A $\leq_M$ B and A is not R.E., then B is not R.E.

Rice's Theorem

- P is undecidable if it is a language consisting of TM descriptions s.t.
  - P is nontrivial: $P \neq \emptyset$ & P does not include all TM descriptions
  - If $L(M_1) = L(M_2)$ then $\langle M_1 \rangle \in P$ iff $\langle M_2 \rangle \in P$
- Proof: By a reduction from A_{TM}, i.e., we show $A_{TM} \leq_M P$
- Let E be a TM s.t. $L(E) = \emptyset$. Assume $\langle E \rangle \notin P$ ($A_{TM} \leq_M \neg P$ works also)
- Note: there exists TM T s.t. $\langle T \rangle \in P$
- $f(\langle M,w \rangle)$ = TM M_w: On input x, simulate M on w. If M accepts, simulate T on x.
- f is a mapping reduction
  - $\langle M, w \rangle \in A_{TM} \Rightarrow L(\langle M_w \rangle)=L(T) \Rightarrow \langle M_w \rangle \in P$
  - $\langle M, w \rangle \notin A_{TM} \Rightarrow L(\langle M_w \rangle)=L(E) \Rightarrow \langle M_w \rangle \notin P$
- $\{\langle M \rangle$: M always halts$\}$, $\{\langle M \rangle : L(M)= \Sigma^*\}$, … all undecidable by Rice's Theorem

Halting Problem

- $HALT_{TM}$ = {<M, w>: M halts on w}
- Theorem: $HALT_{TM}$ is undecidable.
- Proof: We show that if $HALT_{TM}$ is decidable, then so is A_{TM}.
- Preview of reduction: We reduce from A_{TM} to $HALT_{TM}$ ($A_{TM} \leq_M HALT_{TM}$).
- Suppose H is the decider for $HALT_{TM}$. Then define a decider A for A_{TM} as follows. On input <M, w>, A calls H on input <M, w>. If H accepts, then A runs M on w and accepts if M accepts w, rejecting otherwise. If H rejects, then A rejects.
- Consider <M,w> in A_{TM}. Since M accepts w, M halts on w. So H accepts <M, w>. A calls H, which accepts, and then runs M on w, which accepts, so A accepts.
- Consider <M,w> not in A_{TM}. If M does not halt on w, H rejects <M, w>, and so does A. Otherwise, M halts on w and rejects w. So A calls H, which accepts <M, w>. A then calls M on w, which terminates in a reject state, so A rejects.

E_{TM} is undecidable

- E_{TM} = { <M> | L(M) = ∅ } is undecidable
- Proof: Suppose it is decidable. Let R be a TM deciding it.
- Define S, a decider for A_{TM}: On input <M,w>
    - Construct Machine M_1: if input ≠ w, reject else run M on w
        - Note: language of M_1 is either ∅ or {w}
    - Runs R on <M_1>
    - If R accepts, reject; if R rejects, accept
- S is a decider for A_{TM}
- Note: S has to construct M_1: add extra states to check input=w
- Reduction: f takes <M,w> and produces <M_1>. M accepts w iff L(M1) **≠** ∅, so we showed
    - $A_{TM} \leq_M \neg E_{TM}$
    - which implies E_{TM} is not decidable (decidability is *not* affected by complementation)

EQ_{TM} is undecidable

- EQ_{TM} = { <M, N> | L(M) = L(N) } is undecidable }
- Proof: E_{TM} is just a special case where L(N) = ∅. So, show $E_{TM} \leq_M EQ_{TM}$. Let R be a TM deciding EQ_{TM}.
- Define S, a decider for E_{TM}: On input <M>
  - Runs R on <M, N> where N is a TM that rejects all inputs
  - If R accepts, accept; if R rejects, reject
- S is a decider for A_{TM}
- Reduction: f takes <M> and produces <M, N> where N is a TM that always rejects. L(M)=∅ iff L(M)=L(N) (where L(N) = ∅)

EQ_{TM} is not R.E.

- EQ_{TM} = { <M, N> | L(M) = L(N) } is not R.E.
- Recall the corollary: If A $\leq_M$ B and A is not R.E., then B is not R.E.
- But A $\leq_M$ B iff ¬A $\leq_M$ ¬B so to show B is not R.E. we can instead show $A_{TM} \leq_M \neg B$
- Plan: Show $A_{TM} \leq_M \neg EQ_{TM}$
- Proof: F = Given <M, w> (1) construct M_1: always reject and M_2: Run M on w (2) Output <M_1, M_2>
  - If M accepts w, M_2 accepts everything, so M_1, M_2 are not equivalent
  - If M doesn't accept w, M_2 accepts nothing, so M_1, M_2 are equivalent

$\neg EQ_{TM}$ is not R.E.

- $\neg EQ_{TM}$ = { <M, N> | L(M) ≠ L(N) } is not R.E.
- Plan: Show $A_{TM} \leq_M EQ_{TM}$
- Proof: G = Given <M, w> (1) construct M_1: always accept and M_2: Run M on w (2) Output <M_1, M_2>
  - If M accepts w, M_2 accepts everything, so M_1, M_2 are equivalent
  - If M doesn't accept w, M_2 accepts nothing, so M_1, M_2 are not equivalent
- We showed that neither of EQ_{TM}, $\neg EQ_{TM}$ are R.E. so EQ_{TM} is neither R.E. nor co-R.E.!

$REGULAR_{TM}$ is undecidable

- $REQULAR_{TM}$ = { <M> | L(M) is a regular language }
- Plan: $A_{TM} \leq_M EQ_{TM}$
- Proof: Let R be a TM that decides $REQULAR_{TM}$ and construct S, which decides A_{TM} as follows
- S: Given <M,w>
  - (1) Construct N: On input x: If $x \in 0^n1^n$, accept, otherwise run M on w
  - (2) Run R on <N>
  - (3) If R accepts, accept, else reject.
- If M accepts w, N accepts everything, so N is regular
- If M doesn't accept w, N accepts $\{x \in 0^n1^n\}$ so N is not regular