Lecture 33: Reductions and Undecidability

CSCI 81
Spring, 2015

Kim Bruce

Undecidable Problems

The Problem View	The Language View
Does TM M halt on w?	$H = \{<M, w> :$ M halts on $w\}$
Does TM M not halt on w?	$\neg H = \{<M, w> :$ M does not halt on $w\}$
Does TM M halt on the empty tape?	$H_\varepsilon = \{<M> : M$ halts on $\varepsilon\}$
Is there any string on which TM M halts?	$H_{ANY} = \{<M>$: there exists at least one string on which TM M halts $\}$
Does TM M accept all strings?	$A_{ALL} = \{<M> : L(M) = \Sigma^*\}$
Do TMs M_a and M_b accept the same languages?	$EqTMs = \{<M_a, M_b> : L(M_a) = L(M_b)\}$
Is the language that TM M accepts regular?	$TM_{reg} = \{<M>:L(M)$ is regular$\}$

Text's Reduction Example

- Theorem: There exists no general procedure to solve the following problem:
  - Given an angle A, divide A into sixths using only a straightedge and a compass.
- Proof:
  - Suppose that there were such a procedure, call it sixth. Then we could trisect an arbitrary angle using that procedure as follows ...
  - ... but know can't trisect an angle. Therefore algorithm for sixth cannot exist .

Reductions

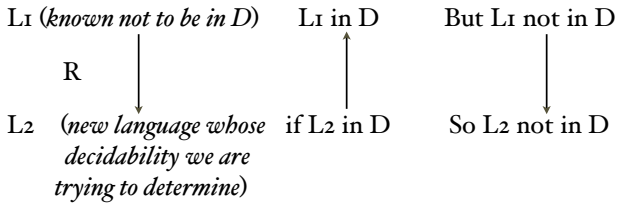
- A reduction R from L_1 to L_2 is one or more Turing machines such that:
  - If there exists a Turing machine Oracle that decides (or semi-decides) L_2, then the Turing machines in R can be composed with Oracle to build a deciding (or a semi-deciding) Turing machine for L_1.
  - I.e., a solution to L_2 gives a solution to L_1
  - Write $L_1 \leq L_2$. (L_1 is easier than L_2)
- Last time showed reduction from H_{TM} to $L_{\varepsilon O}$

Using Reductions

- If R is a reduction from L_1 to L_2 and L_2 is in D, then L_1 is in D
- If R is a reduction from L_1 to L_2 and L_1 is not in D, then L_2 is not in D
- Because H_{TM} not decidable, neither is $L_{\varepsilon o}$

How to use reduction

- Showing that L2 is not in D:



To show L_2 undecidable

- Choose a language L_1:
  - that is already known not to be in D, and
  - that can be reduced to L_2.
- Suppose some *Oracle* solves L_2
- Use *Oracle* as subroutine in TM M to solve L_1.
  - Ensure that if $x \in L_1$, then M(x) accepts, and
  - If $x \notin L_1$, then M(x) rejects.

Mapping Reductions

- L_1 is *mapping reducible* to L2 ($L_1 \leq_M L_2$) iff there exists some computable function f such that:
  - $\forall x \in \Sigma^* (x \in L_1 \leftrightarrow f(x) \in L_2)$.
- To decide whether x is in L_1, we transform it, using f, into a new object and ask whether that object is in L_2.

Mapping Reductions

- L_1 is *mapping reducible* to L_2 ($L_1 \leq_M L_2$) iff there exists some computable function f such that:
  - $\forall x \in \Sigma^*$ ($x \in L_1 \leftrightarrow f(x) \in L_2$).
- Example:
  - $H_{TM} \leq_M H_\varepsilon$ where H_ε = {<M>| M halts on empty tape}
    - Take <M,w>, build new machine M'_w that writes w on tape and then simulates M. Then
    - $\langle M,w\rangle \in H_{TM}$ iff $\langle M'_w\rangle \in H_\varepsilon$

Equality of TM's

- Showed a few lectures back that E_{TM} ={<M> | M is a TM and L(M)=∅} is undecidable.
- Use this to show
  Eq_{TM} = {$\langle M_1,M_2\rangle$ | $L(M_1) = L(M_2)$ }
  is undecidable.
- Show $E_{TM} \leq_M Eq_{TM}$

Reduction

- To find if $\langle M\rangle \in E_{TM}$:
  - Let $M_\emptyset$ be a TM that never accepts anything: $L(M_\emptyset) = \emptyset$.
  - Suppose Oracle solves EQ_{TM}. I.e., Oracle(<M,M'>) always halts and accepts iff L(M) = L(M').
  - Then $\langle M\rangle \in E_{TM}$ iff $\langle M, M_\emptyset\rangle$ accepted by Oracle.
  - Thus EQ_{TM} is undecidable.
- Showed $E_{TM} \leq_M Eq_{TM}$ via $f(M) = \langle M,M_\emptyset\rangle$

Consequences

- If f and g are computable functions, then cannot decide if f = g.
  - Full employment for graders!!
- If V is program for virus, then cannot tell if suspicious program V' is equivalent.

Is everything undecidable?

- L = {<M> | M's program has less than 20 transitions }
- L = {<M,w> | M halts on w in exactly 47 steps }
- Difference between questions about the TM
  - structural properties
- and questions about the languages accepted
  - functional properties

Rice's Theorem

- No non-trivial property of the SD languages is decidable
  or equivalently
- Any language that can be described as
  {<M> | P(L(M)) = true}
  for any non-trivial property P, is not in D
- A property is *trivial* if it is either true for all languages or false for all languages.

Applying Rice

- Must specify property P
- Show domain of P is SD languages
- Show P is non-trivial
  - *true of at least one , false of at least one.*

Applications:

1. {<M> : L(M) contains only even length strings}.
2. {<M> : L(M) contains an odd number of strings}.
3. {<M> : L(M) contains all strings that start with a}.
4. {<M> : L(M) is infinite}.
5. {<M> : L(M) is regular}.
6. {<M>: M accepts ε}.
7. {<M_a, M_b> : L(M_a) = L(M_b)}.

Non-Applicable

1. {<M> : M contains an even number of states}.
2. {<M> : M has an odd number of symbols in its tape alphabet}.
3. {<M> : M accepts ε within 100 steps}.

More Undecidable

- Is L(M) regular?
- Is L(M) context-free?
- Can we automatically check if your program is correct (e.g., matches the solution)?
- Does M ever halt with an error?

Non-SD Languages

- Intuition: Usually involves infinite search or knowing TM will loop.
- Examples:
  - not-H_{TM}
  - {<M> | L(M) = Σ*}
  - {<M> | M halts for no values }

How to prove not SD?

- Show contradiction
- Show L is complement of SD/D language
- Reduction from known non-SD language

L is complement of SD/D language

- Let $H_{\neg ANY}$ = {<M> | M does not accept any string}
- Complement: {<M> | M accepts some string} is in SD/D
- Therefore $H_{\neg ANY}$ not in SD.

Alternate Proof using Reduction

- Show not-$H_{TM} \leq_M H_{\neg ANY}$
- Given <M,w>, build M' s.t. M' erases tape and then runs M on w.
  - Then <M,w> $\in$ not-H_{TM} iff M' $\in H_{\neg ANY}$
- Therefore $H_{\neg ANY}$ not in SD.

Decision Problems for Regular Languages

- For FSMs/Regular languages, most things decidable:
  - A_{FSM} = {⟨M,w⟩ | M is an FSM and w $\in$ L(M)}
  - E_{FSM} ={M | M is a FSM and L(M)=∅}
  - $TOTAL_{FSM}$ ={M | M is a FSM and L(M)=Σ*}
  - $EQUAL_{FSM}$ ={<M,N> | M,N are FSMs and L(M)=L(N)}

Decision Problems for CFGs

- Not for PDAs/CFGs:
  - A_{CFG} = {⟨G,w⟩ | G is a CFG and w $\in$ L(G)}
  - E_{CFG} ={G | G is a CFG and L(G)=∅}
  - $Finite_{CFG}$ = {G | G is a CFG and L(G) is finite}
  - $TOTAL_{CFG}$ ={G | G is a CFG and L(G)=Σ*}
  - $EQUAL_{CFG}$ ={< G, G' > | G, G' are CFGs and L(G)=L(G')}
  - ....

Not Decidable!