Harvard CS 121 and CSCI E-207

Lecture 17: Reductions

Harry Lewis

November 2, 2010

- Reading: Sipser §4.2, §5.1.

Last time we proved these problems undecidable:

- Given a TM M and an input w, does M accept w? (or: does M halt on input w?)
- For a particular fixed machine M_0: given w, does M_0 accept w?

These are proxies for two variants of:

- Will this program terminate given this input?

What if we vary the TM but keep the input fixed?

I.e. what about:

- Given M, does M halt on the empty string?

Given M, does M halt on the empty string? $= \mathrm{HALT}^{\varepsilon}_{\mathrm{TM}}$

Undecidable, Proof by contradiction:

Suppose H were a TM that decided $\{ \langle M \rangle \colon M$ halts on $\varepsilon \}$.
Then H could be used to decide $\mathrm{HALT}_{\mathrm{TM}}$:

Given $\langle M, w \rangle$,

Construct $\langle M_w \rangle$, where M_w is a TM that writes

w on the empty tape and then runs M.

Then run H on input $\langle M_w \rangle$

H halts on $\langle M_w \rangle \Leftrightarrow M_w$ halts on $\varepsilon \Leftrightarrow M$ halts on w

But $\mathrm{HALT}_{\mathrm{TM}}$ is undecidable. $\Rightarrow\Leftarrow$

- We have *reduced* $\mathrm{HALT}_{\mathrm{TM}}$ to the problem $\mathrm{HALT}^{\varepsilon}_{\mathrm{TM}}$

What about

- For a fixed M_0 *and* a fixed w_0, does M_0 halt on input w_0?

"Co-X"

- For any property X that a set might have, a set S is **co-X** iff $\overline{S}$ has property X.
- For example, a co-finite set of natural numbers is a set that is missing only a finite number of elements.
- A co-regular language is . . . ?
- A co-recursive language is . . . ?
- What about a co-CF language?
- Proved last time:
  - A language is recursive if and only if it is both r.e. and co-r.e.

Non-r.e. Languages

Theorem: The following co-r.e. languages are not r.e.:

- $\overline{\mathrm{A}_{\mathrm{TM}}} = \{\langle M, w\rangle : M$ does not accept $w\}$
- $\overline{\mathrm{HALT}_{\mathrm{TM}}} = \{\langle M, w\rangle : M$ does not halt on $w\}$
- $\overline{\mathrm{HALT}^{\varepsilon}_{\mathrm{TM}}} = \{\langle M\rangle : M$ does not halt on $\varepsilon\}$

Proof: If these languages were r.e., then A_{TM}, $\mathrm{HALT}_{\mathrm{TM}}$, and $\mathrm{HALT}^{\varepsilon}_{\mathrm{TM}}$ would be both r.e. and co-r.e. and hence recursive.

Is it possible to determine, given a TM M, whether M accepts a finite or infinite set?

- Let $A_{\text{finite}} = \{\langle M \rangle : L(M)$ is finite$\}$. Is A_{finite} recursive?

Is it possible to determine, given a TM M, whether M accepts a finite or infinite set?

- Let $A_{\text{finite}} = \{\langle M\rangle : L(M) \text{ is finite}\}$. Is A_{finite} recursive?
- Suppose F decides A_{finite}. To decide A_{TM}, given $\langle M\rangle$ and $\langle w\rangle$, construct $\langle M_w^*\rangle$ so that
  - If M accepts w then M_w^* accepts its input, regardless of what it is, and
  - If M does not accept w then M_w^* runs forever.
- Then run F on input $\langle M_w^*\rangle$.
- $L(M_w^*)$ is either Σ^* (and therefore infinite) or $\emptyset$ (and therefore finite) depending on whether or not M accepts w.

Reduce A_{TM} to A_{finite}; since A_{TM} is undecidable, so is A_{finite}

Formalizing the Notion of Reduction

- L_1 "reduces" to L_2 if we can use a "black box" for L_2 to build an algorithm for L_1.

- A function $f : \Sigma_1^* \to \Sigma_2^*$ is <u>computable</u> if there is a Turing machine that for every input $w \in \Sigma_1^*$, M halts with just $f(w)$ on its tape.

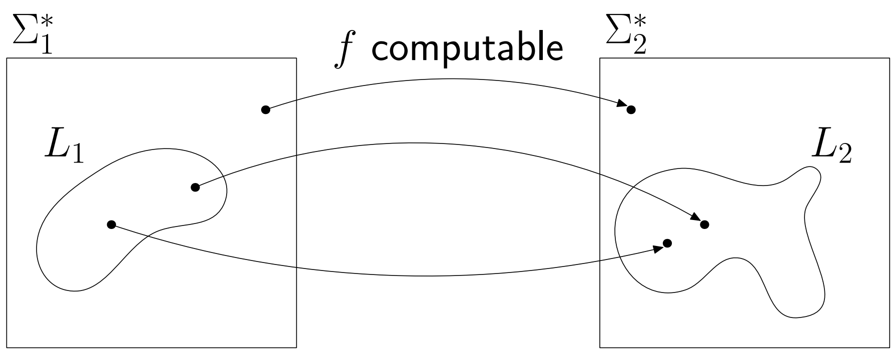
- A <u>(mapping) reduction</u> of $L_1 \subseteq \Sigma_1^*$ to $L_2 \subseteq \Sigma_2^*$ is a computable function
  $f : \Sigma_1^* \to \Sigma_2^*$ such that, for any $w \in \Sigma^*$,
  $w \in L_1$ iff $f(w) \in L_2$

  We write $L_1 \leq_m L_2$.

Properties of Reducibility

Lemma: If $L_1 \leq_m L_2$, then

- if L_2 is decidable (resp., r.e.), then so is L_1;
- if L_1 is undecidable (resp., non-r.e.), then so is L_2.



Examples of Reductions from Last Lecture

- For every r.e. L, $L \leq_m \mathsf{A}_{\mathsf{TM}}$.

- $\mathsf{A}_{\mathsf{TM}} \leq_m \mathsf{HALT}_{\mathsf{TM}}$.

- $\mathsf{HALT}_{\mathsf{TM}} \leq_m \mathsf{HALT}^{\varepsilon}_{\mathsf{TM}}$.

Rice's Theorem

Informally: every (nontrivial) property of r.e. languages is undecidable.

Rice's Theorem: Let $\mathcal{P}$ be any subset of the class of r.e. languages such that $\mathcal{P}$ and its complement are both nonempty. Then the language $L_{\mathcal{P}} = \{\langle M \rangle : L(M) \in \mathcal{P}\}$ is undecidable.

Thus, given a TM M, it is undecidable to tell if

- $L(M) = \emptyset$,
- $L(M)$ is regular,
- $|L(M)| = \infty$, etc.

Proof of Rice's Theorem

- We will reduce L_ε to $L_\mathcal{P}$.
- Suppose without loss of generality that $\emptyset \notin \mathcal{P}$.
- Pick any $L_0 \in \mathcal{P}$ and say $L_0 = L(M_0)$.
- Define $f(\langle M \rangle) = \langle M' \rangle$, where

  M' is TM that on input w,

  · first simulates M on input ε

  · then simulates M_0 on input w

- **Claim:** f is a mapping reduction from L_ε to $L_\mathcal{P}$.
- Since L_ε is undecidable, so is $L_\mathcal{P}$.

Recursion Theory over $\mathcal{N}$

- We have presented this theory as a theory of languages
- Classically it is treated as a theory of sets of numbers
- The two are equivalent since strings can be converted to numbers (treating strings as numerals, for example) and v.v.
- So it makes sense to say "The set of primes is recursive"

Recursive functions

- A function f is recursive if f is computable
- e.g. if there is a TM that always leaves $f(w)$ on the tape when started with input w
- Similarly we can speak of a recursive function from numbers to numbers
- Thm: A set is r.e. iff it is the range of a recursive function

Recursive functions

- A function f is recursive if f is computable
- e.g. if there is a TM that always leaves $f(w)$ on the tape when started with input w
- Similarly we can speak of a recursive function from numbers to numbers
- Thm: A set is r.e. iff it is the range of a recursive function or is $\emptyset$