

Harvard CS 121 and CSCI E-207

Lecture 16: Reductions

Harry Lewis

November 5, 2009

- Reading: Sipser Ch. 5

“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.:

- $\overline{A_{TM}} = \{\langle M, w \rangle : M \text{ does not accept } w\}$
- $\overline{HALT_{TM}} = \{\langle M, w \rangle : M \text{ does not halt on } w\}$
- $\overline{HALT_{TM}^\varepsilon} = \{\langle M \rangle : M \text{ does not halt on } \varepsilon\}$

Proof: If these languages were r.e., then A_{TM} , $HALT_{TM}$, and $HALT_{TM}^\varepsilon$ 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 $A_{\text{infinite}} = \{\langle M \rangle : L(M) \text{ is infinite}\}$. Is A_{infinite} recursive?

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

- Let $A_{\text{infinite}} = \{\langle M \rangle : L(M) \text{ is infinite}\}$. Is A_{infinite} recursive?
- Suppose M_2 decides A_{infinite} . 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 M_2 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_{infinite} ; since A_{TM} is undecidable, so is A_{infinite}

Formalizing the Notion of Reduction

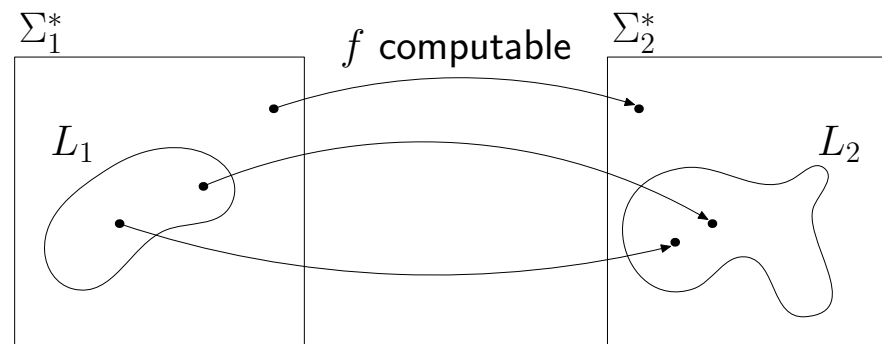
- L_1 “reduces” to L_2 if we can use a “black box” for L_2 to build an algorithm for L_1 .
- A function $f : \Sigma_1^* \rightarrow \Sigma_2^*$ is computable if there is a Turing machine that for every input $w \in \Sigma_1^*$, M halts with just $f(w)$ on its tape.
- A (mapping) reduction of $L_1 \subseteq \Sigma_1^*$ to $L_2 \subseteq \Sigma_2^*$ is a computable function $f : \Sigma_1^* \rightarrow \Sigma_2^*$ 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 Turing-recognizable L , $L \leq_m A_{\text{TM}}$.
- $A_{\text{TM}} \leq_m \text{HALT}_{\text{TM}}$.
- $\text{HALT}_{\text{TM}} \leq_m \text{HALT}_{\text{TM}}^{\varepsilon}$.

Rice's Theorem

Informally: every (nontrivial) property of Turing-recognizable 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