Reducibility

Lecture 17—May 30, 2006
Reducibility

Instructor: Neil Rhodes
Mapping Reduction

We say that a language \( A \) is *mapping-reducible* to \( B \) \((A \leq_M B)\) if:

- There exists a computable function \( f \) that maps \( \Sigma^* \) to \( \Sigma^* \) with:
  - \( w \in A \) iff \( f(w) \in B \)

We say that a function \( f : \Sigma^* \rightarrow \Sigma^* \) is *computable* if:

- There exists a Turing machine \( M \) which, on every input \( w \), halts with \( f(w) \) on its tape
What can be shown with a reduction

$A \leq B$

- If $B$ has a TM, one can construct a TM for $A$

- If $B$ is decidable, one can construct a TM that decides $A$
Mapping-reducible examples

$$\text{EMPTY}_{TM} \leq_M \text{EQ}_{TM}$$

$$\text{PCP} \leq_M \cap_{CFG}$$

$$\text{PCP} \leq_M \text{AMBIG}_{CFG}$$

$$\text{ALL}_{CFG} \leq_M \text{EQ}_{CFG/REG}$$

$$\text{EQ}_{CFG/REG} \leq_M \text{EQ}_{CFG}$$

$$A_{TM} \leq_T \text{REGULAR}_{TM}$$

$$A_{TM} \leq_T \text{CF}_{TM}$$
Not mapping-reducible

$A_{TM} \leq_{M} \text{EMPTY}_{TM}$
Turing reduction

We define an *oracle Turing machine* $M^L$ as a Turing machine with an *oracle* for a language $L$. An oracle machine can write a string onto a special oracle tape, and then get back an answer as to whether that string is in $L$.

- We say that $A$ is *Turing-reducible* to $B$ ($A \leq_T B$) if there exists an oracle machine (with oracle for $B$) that recognizes $A$. 
Examples of Turing reductions

$A_{TM} \leq_{M} \text{EMPTY}_{TM}$

$\cap_{CFG} \leq_{T} \text{ALL}_{CFG}$
Differences between Turing- and mapping- reductions

Number of times membership question can be asked
- Mapping reduction uses membership in B exactly once
- Turing reduction can make multiple calls to oracle

When membership question can be asked
- Mapping reduction can only ask at the end of computation
- Turing reduction can do computation after querying oracle

Membership in language A versus membership in language B
- Mapping reduction maps yes-to-yes and no-to-no
- Turing reduction can invert mapping, or do other computation

Easy to simulate mapping reduction with Turing reduction but not vice-versa
- Turing reductions are stronger than mapping reductions. Sometimes we don’t use them because of their strength (Turing reductions could take more time, for example).
Reducing $\cap_{\text{CFG}}$ to $\text{ALL}_{\text{CFG}}$

$\text{ALL}_{\text{CFG}} = \{ G | G \text{ is a CFG and } L(G) = \Sigma^* \}$

- Create $G' = (G_1 \cap G_2)^c$
- Check whether $G' \in \text{ALL}_{\text{CFG}}$. If yes, $G_1$ and $G_2$ don’t intersect. If no, they do intersect.

- Is this valid? Is $G_1 \cap G_2$ a CFG? Is the complement of a CFL a CFL?
  - No

- However, look at $\text{PCP} \leq \cap_{\text{CFG}} \leq \text{ALL}_{\text{CFG}}$
  - The grammars used in the reduction of PCP were deterministic!

  - So, actually, the reduction is:

  $G' = G_1^c \cup G_2^c = (G_1 \cap G_2)^c$
  - $G_1^c$ is DCFG
  - Union is CFG