# Mapping Reducibility and The Recursion Theorem

CS154

Chris Pollett

May 3, 2006.

# Outline

- More on Mapping Reducibility
- Rice's Theorem
- Start of the Recursion Theorem

# More on Mapping Reducibility

- We are now going to give some results about mapping reductions and give an example.

**Theorem (*).** If $A \leq_m B$ and $B$ is decidable, then $A$ is decidable.

**Proof.** Let $M$ be a decider for $B$ and let $f$ be the reduction from $A$ to $B$. A decider $N$ for $A$ can be defined as follows:

$N$ = "On input $w$:

1. Compute $f(w)$.
2. Run $M$ on input $f(w)$ and output whatever $M$ outputs."

- Taking the contrapositive…

**Corollary.** If $A \leq_m B$ and $A$ is undecidable, then B is undecidable.

# An Example of Mapping Reducibility

- Recall we used a reduction from $A_{TM}$ to show $HALT_{TM}$ is undecidable.
- We can rephrase this reduction as a mapping reducibility...
- What we need to do is come up with a function $F$ that takes inputs of the form $<M, w>$ and returns outputs of the form $<M', w'>$, so that $<M, w>$ is in $A_{TM}$ iff $<M', w'>$ is in $HALT_{TM}$.
- To do this, let $F$ compute:

F = "On input $<M, w>$ :

1. Construct the following machine $M'$.
    1. Run $M$ on $x$.
    2. If $M$ accepts, *accept*.
    3. If $M$ *rejects,* enter an infinite loop."
2. Output $<M', w>$."

# Mapping Reducibility and Turing Recognizability

**Theorem.**

(a) If $A \leq_m B$ and $B$ is Turing Recognizable, then $A$ is Turing Recognizable.

(b) If $A \leq_m B$ and $B$ is co-Turing Recognizable, then $A$ is co-Turing Recognizable.

**Proof.** The proof is essentially the same as Theorem (*) that we did earlier.

**Corollary.**

(a) If $A \leq_m B$ and $A$ is not Turing Recognizable, then $B$ is not Turing Recognizable.

(b) If $A \leq_m B$ and $A$ is not co-Turing Recognizable, then $B$ is not co-Turing Recognizable.

# Applications to $EQ_{\mathrm{TM}}$

**Theorem.** $EQ_{\mathrm{TM}}$ is neither Turing-recognizable nor co-Turing-recognizable.

**Proof.** First we show $EQ_{\mathrm{TM}}$ is not Turing recognizable. To do this we show that $A_{\mathrm{TM}}$ is mapping reducible to $\overline{EQ}_{\mathrm{TM}}$. The reducing function $F$ is as follows:

$F$ = " On input <*M, w*>:

1. Construct two machines: $M_1$ , which on any input *rejects* and $M_2$ which on any input erases the input and then simulates $M$ on $w$ and accepts if $M$ does.
2. Output < $M_1$, $M_2$>"

To see $EQ_{\mathrm{TM}}$ is not co-Turing recognizable we show $A_{\mathrm{TM}}$ is mapping reducible to $EQ_{\mathrm{TM}}$. To do this consider the reduction $G$:

$G$ = " On input <*M, w*>:

1. Construct two machines: $\mathrm{M}_1$ , which on any input *accepts* and $M_2$ which on any input erases the input and then simulates $M$ on $w$ and accepts if $M$ does.
2. Output < $M_1$, $M_2$>"

# Rice's Theorem

- This theorem shows that almost any problem one could come up with connected to Turing Machines is undecidable.

**Theorem.** Let $P$ be a language such that there exists TM descriptions $<M> \in P$ and $<M'> \notin P$. Further assume that whenever we have two machines $M_1$ and $M_2$ such that $L(M_1) = L(M_2)$, then we have $<M_1> \in P$ iff $<M_2> \in P$. Then $P$ is undecidable.

**Proof.** Suppose we had a decider $R$ for $P$. We show how to use $R$ to build a decider for $A_{TM}$. Let $T_\emptyset$ be a TM which always rejects, so $L(T_\emptyset) = \emptyset$. We may assume $T_\emptyset \notin P$; otherwise, we carry out our argument using $\overline{P}$. Because $P$ is not trivial there exists a TM $T$ with $T \in P$. Using these machines consider the following decider $S$ for $A_{TM}$:

$S$ = "On input $<M, w>$:

1. Use $M$ and $w$ to construct the following TM $M_w$ :

   $M_w$ = " On input $x$:

   1. Simulate $M$ on $w$. If it halts and reject, *reject*. If it accepts, proceed to stage 2.
   2. Simulate $T$ on $x$. If it accepts, *accept*."

2. Use TM $R$ to determine whether $< M_w > \in P$. If yes, *accept*. If no, *reject*."

# Example Use of Rice's Theorem

- Consider the language $L = \{<M> \mid M$ is a TM and $1011 \in L(M)\}$.
- This language contains some, but not all TM encodings.
- It further has the property that for any $M_1$ and $M_2$ such that $L(M_1) = L(M_2)$, then we have $<M_1> \in L$ iff $<M_2> \in L$.
- Therefore, by Rice's Theorem it is undecidable.

# The Recursion Theorem

- One interesting property of living things is that they can reproduce – that is, they can produce "exact" copies of themselves.
- Can machines do this? As a first step:

**Lemma.** There is a computable function $q:\Sigma^* \rightarrow \Sigma^*$ , where if w is any string $q(w)$ is the description of a Turing Machine $P_w$ that prints out w and then halts.

**Proof.**

$Q$ ="On input $w$:

1. Construct the following TM $P_w$.

    $P_w$ = " On any input:

    1. Erase the input.
    2. Write $w$ on the tape.
    3. Halt."

2. Output $< P_w >$."