# Rice's Theorem Meets Program Analysis

## Bad News and How We Deal with it In Practice

Monday, December 10, 2007
Reading: Sipser 5.1, 5.3; Kozen 32-34

**CS235 Languages and Automata**

Department of Computer Science
Wellesley College

## Overview of Today's Lecture

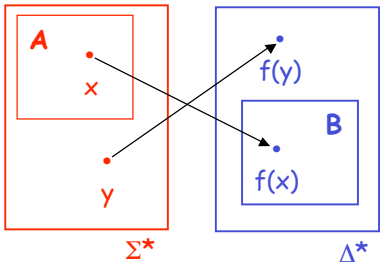
- More examples of undecidability via reduction
- Reduction arguments generalize to *Rice's Theorem*: any nontrivial program property is undecidable.
- How program analysis deals with Rice's Theorem in practice: *restricted systems* and *conservative approximations*.

# Review of Reduction

A (many-to-one) **reduction of A to B** is a function **f**: $\Sigma^* \rightarrow \Delta^*$ such that x in A iff f(x) in B.

**f** must be a **computable function** = one that is describable by some Turing Machine that, on input **w**, halts with just f(w) on its tape.

We write $A \leq_m B$ if A is reducible to B by a computable function.



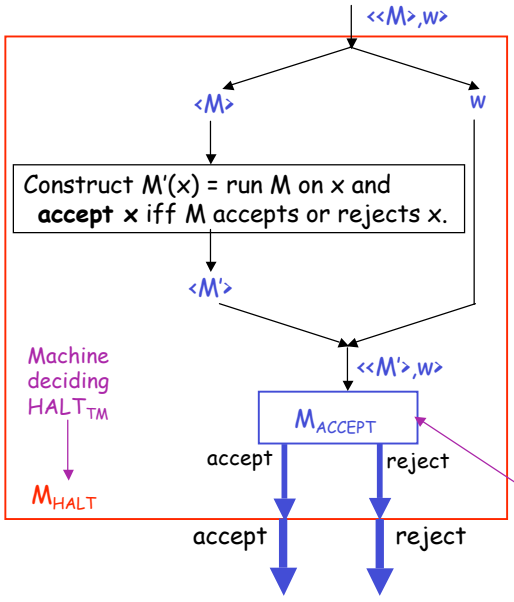

To use reduction in **undecidability proofs**:

Given an A that is known to be undecidable, show that if B were decidable, it could be used to decide A. So B must be undecidable too!

E.g. $A = HALT_{TM}$ (known to be undecidable)
$B = ACCEPT_{TM}$ (want to show undecidable)



# Reducing $HALT_{TM}$ to $ACCEPT_{TM}$




$HALT_{TM}$ = {<<M>, w> | M is a TM and M halts on w}

$ACCEPT_{TM}$ = {<<M>, w> | M is a TM and M accepts w}

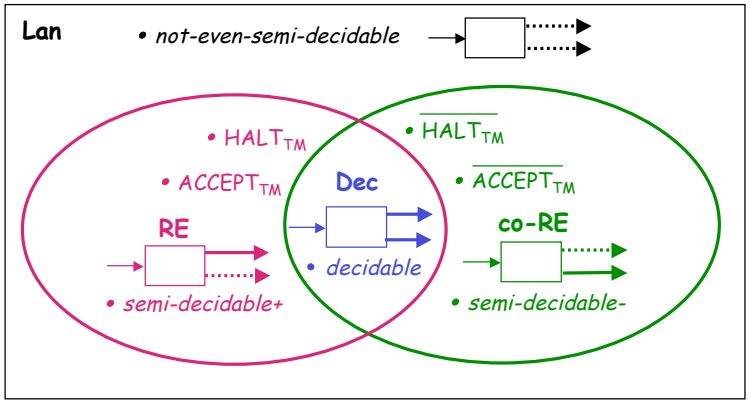
*Essence of Reduction:*

<<M>,w> $\in HALT_{TM}$

iff

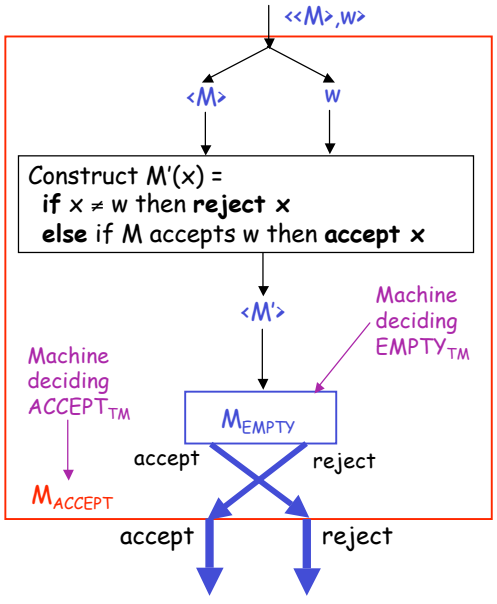
<<M'>,w> $\in ACCEPT_{TM}$

# Positioning $ACCEPT_{TM}$ & $HALT_{TM}$ On the Map






# $EMPTY_{TM}$ Is Undecidable



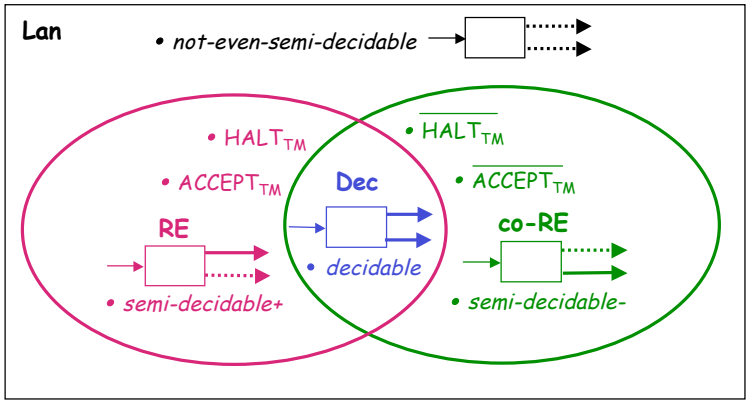

$EMPTY_{TM} = \{\langle M\rangle \mid$ M is a TM and $L(M) = \emptyset\}$

*Essence of Reduction:*

$\langle\langle M\rangle, w\rangle \in ACCEPT_{TM}$ iff $\langle M'\rangle \notin EMPTY_{TM}$

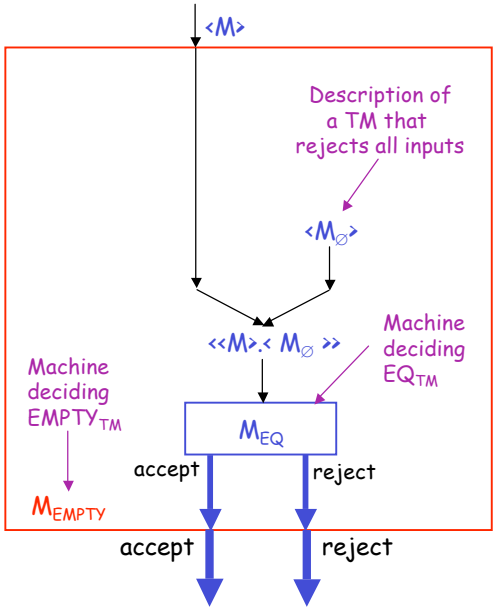
# Where Does $EMPTY_{TM}$ Belong?

Please place $EMPTY_{TM}$ and $\overline{EMPTY_{TM}}$ on the map:






# $EQ_{TM}$ Is Undecidable




$EQ_{TM} = \{\langle\langle M_1\rangle,\langle M_2\rangle\rangle \mid M_1 \text{ \& } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$

*Essence of Reduction:*

$\langle M\rangle \in EMPTY_{TM}$

iff

$\langle\langle M\rangle,\langle M_\emptyset\rangle\rangle \in EQ_{TM}$

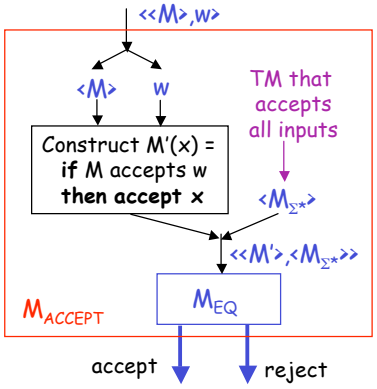
# Relationships between Reducible Languages

- If $A \leq_m B$ then $\overline{A} \leq_m \overline{B}$.
- If $A \leq_m B$ and $B \in$ **Dec** then $A \in$ **Dec**.
- If $A \leq_m B$ and $B \in$ **RE** then $A \in$ **RE**.
- If $A \leq_m B$ and $A \notin$ **Dec** then $B \notin$ **Dec**.
- If $A \leq_m B$ and $A \notin$ **RE** then $B \notin$ **RE**.
- If $A \leq_m B$ and $A \in$ **RE** - **Dec** then $\overline{B} \notin$ **RE**.

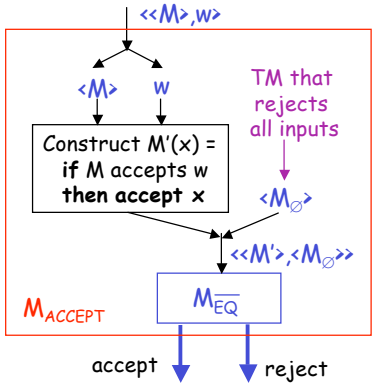


# Understanding $EQ_{TM}$

- If $A \leq_m B$ and $A \in$ **RE** - **Dec** then $\overline{B} \notin$ **RE**.




$ACCEPT_{TM} \leq_m EQ_{TM}$, so $\overline{EQ_{TM}} \notin$ **RE**.

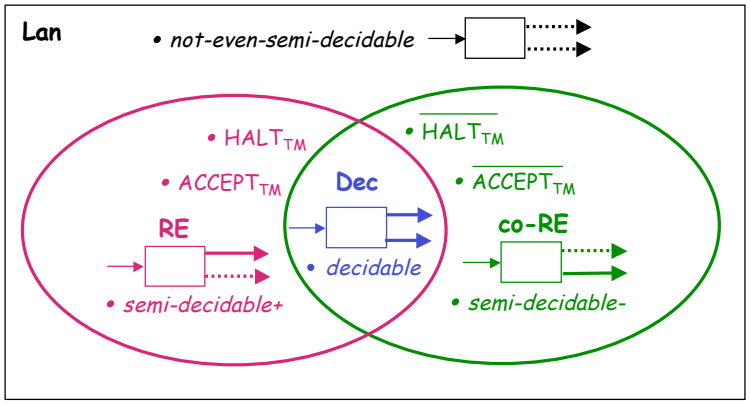



$ACCEPT_{TM} \leq_m \overline{EQ_{TM}}$, so $EQ_{TM} \notin$ **RE**.

# Where Does $EQ_{TM}$ Belong?

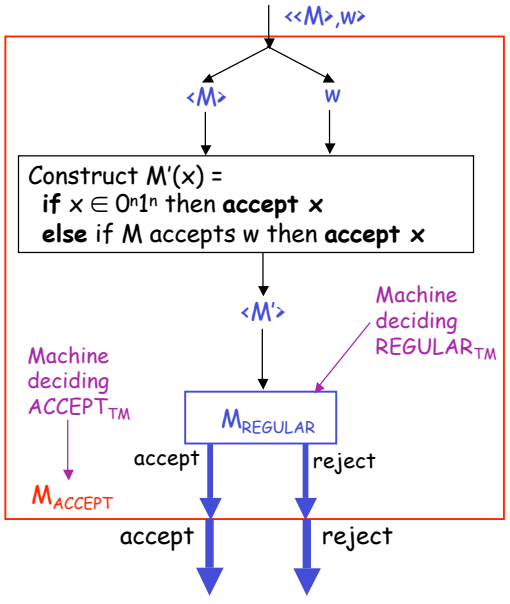
We now know that both $EQ_{TM}$ and $\overline{EQ_{TM}}$ are not in **RE**.

Please place $EQ_{TM}$ and $\overline{EQ_{TM}}$ on the map:






# $REGULAR_{TM}$ Is Undecidable




$REGULAR_{TM}$
= {<M> | M is a TM and L(M) is regular

*Essence of Reduction:*

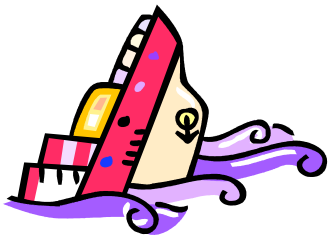
If $w \in L(M)$, then $L(M') = \Sigma^*$
If $w \notin L(M)$, then $L(M') = 0^n1^n$

So, $\langle\langle M\rangle,w\rangle \in ACCEPT_{TM}$

iff

$\langle M'\rangle \in REGULAR_{TM}$

## Rice's Theorem



The results for $EMPTY_{TM}$ and $REGULAR_{TM}$ can be generalized to **Rice's Theorem**:

> *Any nontrivial property of the language of a Turing Machine is undecidable.*

- A property is nontrivial if it contains some, but not all, TM descriptions.
- The properties must be of the languages of the machines, not the machines themselves: for any property P, $L(M_1) = L(M_2)$ implies $\langle M_1 \rangle \in P$ iff $\langle M_2 \rangle \in P$.
- For a proof of Rice's Theorem, see Sipser Ex. 5.28 or Kozen Lec. 34.



## Some Applications of Rice's Theorem

The following languages are undecidable:

- $CFL_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a context-free language}\}$
- $DEC_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is } \in \textbf{Dec}\}$
- $INCLUDES\_ONE_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } 1 \in L(M)\}$
- $ALL_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \Sigma^*\}$
- $FINITE_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is finite}\}$
- $INFINITE_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is infinite}\}$

## Rice's Theorem and Program Analysis

A consequence of Rice's theorem is that any nontrivial property of a program is undecidable. E.g.

- Does evaluation of expression E terminate?
- Does evaluation of expression E have a side effect?
- Is local variable x in C/C++/Pascal initialized?
- Does an expression E have a type in the polymorphic type system?
- Does a program satisfy a given specification?
- Will this object ever be referenced again? (if not, it can be garbage collected).



## Two Ways to Deal with Rice's Theorem

1. **Restrict the system** so that properties are decidable. E.g. SML type system is not full polymorphic type system.
2. Use **conservative approximations** to properties in program analysis tools (e.g., type checkers, compilers). E.g.:
   - termination analysis
   - side-effect analysis
   - initialization analysis
   - return analysis
   - reachability analysis