Reduction & Closure Properties

Tools for Proving Undecidability

Thursday, December 4, 2008
Reading: Sipser 5.1; Kozen 32-33; Stoughton 5.2

CS235 Languages and Automata

Department of Computer Science
Wellesley College

Two Ways to Show a Language L Undecidable

1. Use **diagonalization** argument like that for $HALT_{TM}$. This is cumbersome.
2. Transform an existing undecidable language to L via a technique called **reduction**. Much easier in practice.

Reduction *or* The Blue Elephant Gun

Q: How do you shoot a blue elephant?

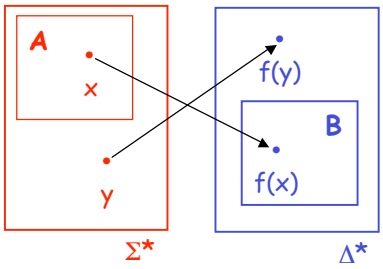
A: With a blue elephant gun, of course!

Q: How do you shoot a white elephant?

A: Hold its trunk until it turns blue, and then shoot it with a blue elephant gun!

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 computable by a terminating Turing Machine



How To Use Reduction

In **proofs by construction**:

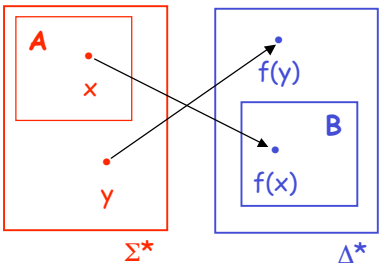
Given a B that is known to be solvable, use it to solve A.

E.g. A = sorting the lines of a file
B = sorting the elts of an array.

In **proofs by contradiction**:

Given an A that is known to be unsolvable, show that if B existed, it could be used to solve A. So B must be unsolvable too!

E.g. A = $HALT_{TM}$
B = $ACCEPT_{TM}$



What About $ACCEPT_{TM}$?

Remember $ACCEPT_{TM}$?

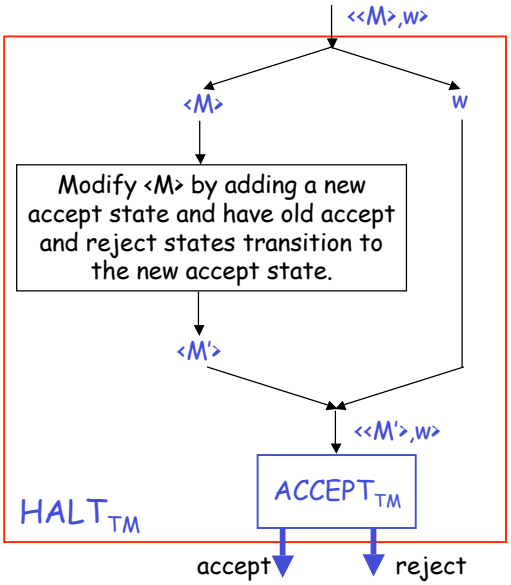
$ACCEPT_{TM}$ = {<<M>, w> | w is in L(M) (i.e. M accepts w)}

We can show that $ACCEPT_{TM}$ is undecidable using a **diagonalization** argument similar to that for $HALT_{TM}$.

However, there's an easier technique = show $ACCEPT_{TM}$ is undecidable using a **reduction** from $HALT_{TM}$:

$HALT_{TM}$ = {<<M>, w> | M halts on w (i.e. M decides w)}

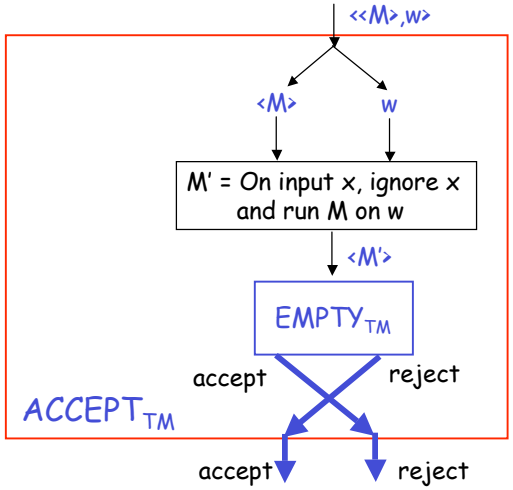
Reducing $HALT_{TM}$ to $ACCEPT_{TM}$



Key fact:
M halts on w
iff
M' accepts w

$EMPTY_{TM}$: Undecidable by Reduction from $ACCEPT_{TM}$

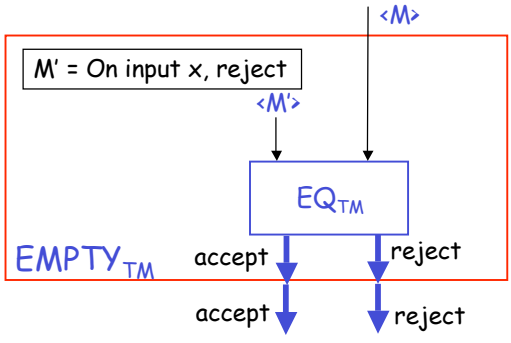
$EMPTY_{TM}$ = {<M> | M is a TM and L(M) = ∅}



Key fact:
M accepts w
iff
M' accepts every x

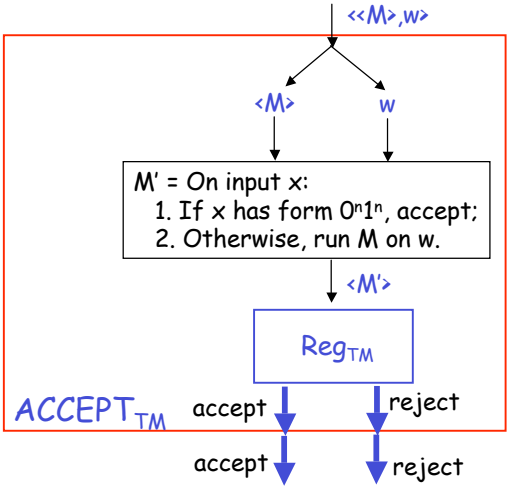
EQ_{TM}: Undecidable by Reduction from $EMPTY_{TM}$

EQ_{TM} = {<M_1, M_2> | M_1 and M_2 are TMs and $L(M_1) = L(M_2)$}



REG_{TM}: Undecidable by Reduction from $ACCEPT_{TM}$

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



Key fact:
M accepts w
iff
M' accepts a
regular language
({0,1}*)

Closure Properties of **Dec** and **RE**

Dec is closed under:
- union
- intersection
- concatenation
- Kleene star
- **complement**

RE is closed under:
- union
- intersection
- concatenation
- Kleene star

We've already seen the complement story:
- L in **Dec** implies $\overline{L}$ in **Dec**.
- L and $\overline{L}$ are in **RE** implies L and $\overline{L}$ in **Dec**.
- L is semi-decidable+ (in **RE** — **Dec**)
  and $\overline{L}$ semi-decidable- (in **co-RE** — **Dec**)

Now we'll study the other properties

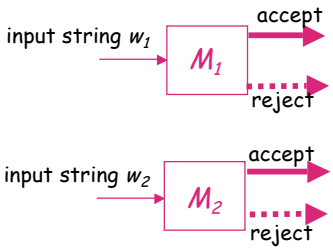
Digression: How to Run Two TMs in Parallel?

We often want to run two accepting machines M_1 and M_2 on the same or different inputs.

The machines should not be run *sequentially* (say M_1 before M_2) because if M_1 loops, M_2 will never run.

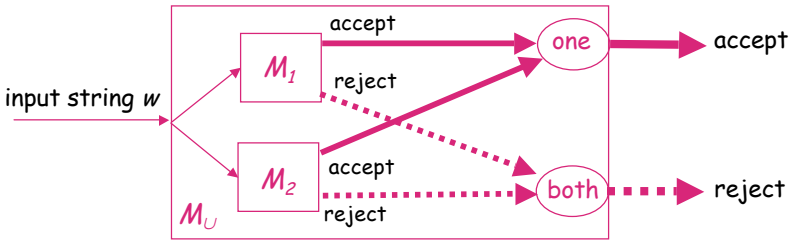
Instead, the machines are run *in parallel* by performing one step of M_1 followed by one step of M_2, alternating between the two machines.

After each step, we can check whether either machine is in its accept state or reject state. So it's possible to run M_1 and M_2 in parallel until one accepts, both accept, one rejects, or both reject.



Example: RE is Closed Under Union

Suppose L_1, L_2 are accepted by machines M_1, M_2, respectively. $L_1 \cup L_2$ is accepted by the following machine $M_\cup$, which runs M_1 and M_2 in parallel until one accepts or both reject.



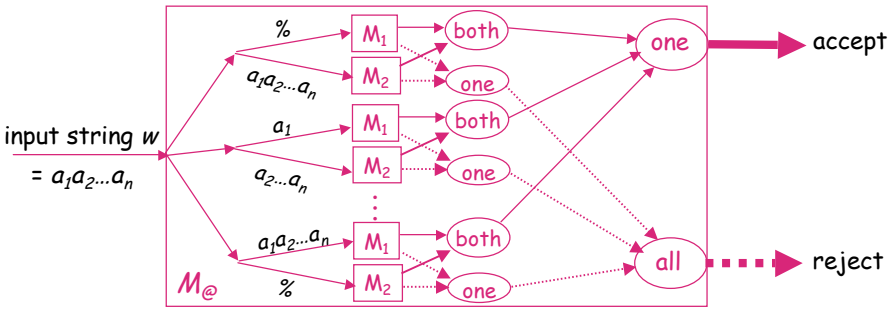
It is essential to run M_1 and M_2 in *parallel*. Why?

A similar diagram shows **Dec** is closed under union, but in that case M_1 and M_2 can be run *sequentially*. Why?

Similar arguments show **RE**, **Dec** are closed under intersection.

Example: **RE** is Closed Under Concatenation

Suppose L_1, L_2 are accepted by machines M_1, M_2, respectively. L_1 @ L_2 is accepted by the following machine $M_@$, which runs copies of M_1 and M_2 in parallel on all possible decompositions of w into pairs of substrings until one pair accepts or all reject.



This diagram can be implemented by a Turing Machine program that loops over all possible decompositions, interleaving the steps for each.

Similar ideas show that **Dec** is closed under concatenation and that both **RE** and **Dec** are closed under Kleene star.