Languages, Reductions, and Undecidability

November 28, 2012
CS 81: Computability and Logic

An engineer and a mathematician are out for a walk and spot a house on fire. There is a garden hose lying in the yard; the engineer hooks it up and puts out the fire. They continue walking, and see a home where a couple is washing their car in their driveway. The mathematician detaches their hose and sets their house on fire, thus reducing it to a previously solved problem.
Church-Turing Thesis

If it can be done at all, then (with suitably coded inputs and outputs) it can be done by

✓ A Turing Machine
✓ Lambda Calculus
✓ An Unrestricted Grammar
✓ A 2-register machine
✓ Java
✓ Scheme
✓ Python
✓ …

Consequently, in most cases where I say “TM,” you can think “Program.”
**TM Languages**

✓ A TM **accepts** a string if it halts saying “yes”.

✓ A language is **semidecidable** (a.k.a. recognizable, recursively enumerable) if there is a TM that accepts exactly the strings in the language.

✓ A language is **decidable** (a.k.a. recursive) if it is accepted by a TM that always halts (i.e., the TM always says “yes” or “no”).
Decidable vs. Semidecidable

✓ If a language is decidable, then its complement is decidable. Why?

✓ If a language is semidecidable, and its complement is semidecidable, then the language is decidable. Why?
Languages of Acceptance

Which are semidecidable (by a TM)? Decidable?

✓ $A_{DFA} = \{ \langle D, w \rangle | D \text{ a DFA, } D \text{ accepts } w \}$

✓ $A_{NFA} = \{ \langle N, w \rangle | N \text{ an NFA, } N \text{ accepts } w \}$

✓ $A_{RE} = \{ \langle R, w \rangle | R \text{ a regexp, } R \text{ matches } w \}$

✓ $A_{CFG} = \{ \langle G, w \rangle | G \text{ a CFG, } G \text{ produces } w \}$

✓ $A_{TM} = \{ \langle M, w \rangle | M \text{ a TM, } M \text{ accepts } w \}$
Semidecidability

Show that these languages are semidecidable.

✓ Accepts-s := \{ \langle M \rangle \mid M \text{ accepts } s \} 
✓ NE_{TM} := \{ \langle M \rangle \mid M \text{ accepts at least one } w \in \Sigma^* \} 

Showing a language not semidecidable requires a different approach. (See the Homework.)
Is There More?
Digression: Bootstrapping a Compiler

Lots of compilers are written in the same language they compile!

✓ Gnu C Compiler (used in CS 105) is written in C
✓ Glasgow Haskell Compiler (used in CS 131) is in Haskell

Practical reasons to run programs on their own source code!
$A_{TM}$ IS NOT DECIDABLE

$$A_{TM} = \{ \langle M, w \rangle \mid M \text{ a TM, } M \text{ accepts } w \}$$

<table>
<thead>
<tr>
<th></th>
<th>$\langle M_0 \rangle$</th>
<th>$\langle M_1 \rangle$</th>
<th>$\langle M_2 \rangle$</th>
<th>$\langle M_3 \rangle$</th>
<th>$\langle M_4 \rangle$</th>
<th>$\cdots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_0$</td>
<td>Acc.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_1$</td>
<td></td>
<td>Acc.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td></td>
<td></td>
<td>Acc.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_3$</td>
<td></td>
<td></td>
<td></td>
<td>Acc.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Acc.</td>
<td>Acc.</td>
</tr>
</tbody>
</table>
Is There More?

- Regular: $a^*b^*$
- Context Free: $a^n b^n$
- Decidable: $a^p b^p c^p$
  (p perfect)
- Semidecidable: $\text{ATM}$

?
What is the complement of $A_{\text{TM}}$?
We’ll Stop Here
**Obligatory Corollary**

Theorem

The language

\[ H = \{ \langle M, w \rangle \mid M \text{ a TM, } M \text{ halts on } w \} \]

is not decidable.

Proof.

Suppose there were a halt-checking TM…

\[ \square \]
Undecidability, So Far

1. Acceptance for TMs is semidecidable but not decidable.

\[ A_{TM} = \{ \langle M, w \rangle | M \text{ a TM, } M \text{ accepts } w \} \]

2. Non-Acceptance for TMs is not semidecidable (hence not decidable).

3. TM Halting is semidecidable but not decidable.

\[ H = \{ \langle M, w \rangle | M \text{ a TM, } M \text{ halts on } w \} \]
**Problem Hardness**

If I proved $P = NP$ then I would become rich and famous.

Therefore, being rich and famous can’t be harder than proving $P = NP$!

$Rich\ &\ Famous \leq Prove\ P = NP$
Reductions

Given problems $P$ and $Q$, we say that

$$P \leq Q$$

if a solution to $Q$ would let us solve $P$ as well.

✓ I.e., (P is “not fundamentally more difficult” Q.)

✓ We say “P reduces to Q.”

In Math, we often show we can solve a problem $X$ by taking advantage of previously-solved problem $Q$ (i.e., prove $X \leq Q$)

In Theocomp, we typically prove a problem $X$ hard by showing a solution to $X$ would also solve the hard problem $P$ (i.e., prove $P \leq X$).
Reductions

To prove that $P$ reduces to $Q$ ($P \leq Q$), it suffices to prove:

✓ If we have a solver for $Q$, then we can use it to solve any instance of $P$.
✓ I.e., show you could construct a $P$-solver if you could make calls to a $Q$-solving subroutine.

Commonly, we instead prove a “mapping reduction”:

✓ For every instance of $P$, we can construct an instance of $Q$ with the same yes/no answer.

Why is this enough?
**Warning**

It is easy to get the reduction backwards!

Correct form:

- ✓ Assume the unknown problem $X$ is decidable
- ✓ Show that it means that $H$ (or $A_{TM}$ or ...) is decidable.
- ✓ Contradiction.
- ✓ Therefore, $X$ is not decidable

Or:

- ✓ Assume the unknown problem $X$ is semidecidable
- ✓ Show that it means that $\neg H$ (or $\neg A_{TM}$ or ...) is semidecidable.
- ✓ Contradiction.
- ✓ Therefore, $X$ is not semidecidable.

If you ever find yourself assuming things previously proved impossible ("assume I had a way to decide halting... then $X$ is decidable.") you’re doing the reduction wrong!
Reduction Practice

Show the following are not decidable (e.g., by reducing $A_{TM}$ to each).

✓ $NE_{TM} := \{ \langle M \rangle \mid M \text{ accepts at least one } w \in \Sigma^* \}$
✓ $E_{TM} := \{ \langle M \rangle \mid L(M) = \emptyset \}$
✓ $ALL_{TM} := \{ \langle M \rangle \mid L(M) = \Sigma^* \}$
✓ $Accepts-s := \{ \langle M \rangle \mid M \text{ accepts } s \}$
✓ $Regular := \{ \langle M \rangle \mid L(M) \text{ is regular} \}$