Today's learning goals

- Define and explain core example of decision problems: $A_{DFA}$, $E_{DFA}$, $EQ_{DFA}$, $A_{TM}$, $HALT_{TM}$
- Define reductions from one problem to another.
- Use reductions to prove undecidability.

Exam 3 grades published today
HW 6 grades published this week
HW 7 due in 1 week.
No discussion section this week
No class on Thursday

Great examples!
(SKIP: Reductions via computation histories)
A proof that the Halting Problem is undecidable
Geoffrey K. Pullum
(http://www.lel.ed.ac.uk/~gpullum/loopsnoop.html)

No general procedure for bug checks will do. Now, I won’t just assert that, I’ll prove it to you. I will prove that although you might work till you drop, you cannot tell if computation will stop.

For imagine we have a procedure called P that for specified input permits you to see whether specified source code, with all of its faults, defines a routine that eventually halts. You feed in your program, with suitable data, and P gets to work, and a little while later (in finite compute time) correctly infers whether infinite looping behavior occurs.

If there will be no looping, then P prints out ‘Good.’ That means work on this input will halt, as it should. But if it detects an unstoppable loop, then P reports ‘Bad!’ — which means you’re in the soup.

Well, the truth is that P cannot possibly be, because if you wrote it and gave it to me, I could use it to set up a logical bind that would shatter your reason and scramble your mind.

Here’s the trick that I’ll use — and it’s simple to do. I’ll define a procedure, which I will call Q, that will use P’s predictions of halting success to stir up a terrible logical mess.

For a specified program, say A, one supplies, the first step of this program called Q I devise is to find out from P what’s the right thing to say of the looping behavior of A run on A.

If P’s answer is ‘Bad!’, Q will suddenly stop. But otherwise, Q will go back to the top, and start off again, looping endlessly back, till the universe dies and turns frozen and black.

And this program called Q wouldn’t stay on the shelf; I would ask it to forecast its run on itself. When it reads its own source code, just what will it do? What’s the looping behavior of Q run on Q?

If P warns of infinite loops, Q will quit; yet P is supposed to speak truly of it! And if Q’s going to quit, then P should say ‘Good.’ Which makes Q start to loop! (P denied that it would.)

No matter how P might perform, Q will scoop it: Q uses P’s output to make P look stupid. Whatever P says, it cannot predict Q: P is right when it’s wrong, and is false when it’s true!

I’ve created a paradox, neat as can be — and simply by using your putative P. When you posited P you stepped into a snare; Your assumption has led you right into my lair.

So where can this argument possibly go? I don’t have to tell you; I’m sure you must know. A reductio: There cannot possibly be a procedure that acts like the mythical P.

You can never find general mechanical means for predicting the acts of computing machines; it’s something that cannot be done. So we users must find our own bugs. Our computers are losers!
Reduction?

A problem $P_1$ **reduces to** a problem $P_2$ if any solution for $P_2$ can be used to solve $P_1$.

In other words: using a solution for $P_2$ as a subroutine gives a solution for $P_1$.

In our example: we used a solution for $\text{HALT}_{\text{TM}}$ to get a solution for $A_{\text{TM}}$. This means that $A_{\text{TM}}$ **reduces to** $\text{HALT}_{\text{TM}}$. 

$$\text{HALT}_{\text{TM}} = \{ <M,w> \mid M \text{ halts on } w \}$$

$$A_{\text{TM}} = \{ <M,w> \mid w \in L(M) \}$$
Reduction?

A problem $P_1$ reduces to a problem $P_2$ if any solution for $P_2$ can be used to solve $P_1$.

If $P_1$ reduces to $P_2$ and

A. $P_1$ is decidable, then $P_2$ is also decidable.
B. $P_2$ is decidable, then $P_1$ is also decidable.
C. Both of the above.
D. None of the above.
E. I don't know.
Reduction?

A problem $P_1$ reduces to a problem $P_2$ if any solution for $P_2$ can be used to solve $P_1$.

If $P_1$ reduces to $P_2$ and

A. $P_1$ is undecidable, then $P_2$ is also undecidable.
B. $P_2$ is undecidable, then $P_1$ is also undecidable.
C. Both of the above.
D. None of the above.
E. I don't know.
Reduction?

A problem $P_1$ **reduces to** a problem $P_2$ if any solution for $P_2$ can be used to solve $P_1$.

**New strategy:** to prove that a problem is undecidable, prove that a problem we know to be undecidable reduces to it.

**Idea:** Reductions relate the difficulty of problems.
Reduction?

A problem \( P_1 \) reduces to a problem \( P_2 \) if any solution for \( P_2 \) can be used to solve \( P_1 \).

Which of the following is false?

A. \( A_{TM} \) reduces to \( A_{TM} \).
B. \( A_{TM} \) reduces to the complement of \( A_{TM} \).
C. \( A_{TM} \) reduces to \( \{0,1\}^* \).
D. \( \{0,1\}^* \) reduces to \( A_{TM} \).
E. I don't know.
Reminder: \( \text{HALT}_\text{TM} \) is undecidable \((\text{Theorem 5.1})\)

**Proof (using reductions):** We will show that \( A_{\text{TM}} \) reduces to \( \text{HALT}_\text{TM} \), and therefore (since \( A_{\text{TM}} \) is undecidable), \( \text{HALT}_\text{TM} \) must be undecidable.
Reminder: $\text{HALT}_{TM}$ is undecidable

Proof (using reductions): We will show that $A_{TM}$ reduces to $\text{HALT}_{TM}$, and therefore (since $A_{TM}$ is undecidable), $\text{HALT}_{TM}$ must be undecidable.

Assume that $M_{\text{HALT}}$ is a machine that decides $\text{HALT}_{TM}$.

Goal: Define decider for $A_{TM}$ using $M_{\text{HALT}}$ as subroutine.

"On input $<M,w>$ … Want to accept if $w$ in $L(M)$, reject o.w."
Reminder: \(\text{HALT}_{\text{TM}}\) is undecidable

**Proof (using reductions):** We will show that \(A_{\text{TM}}\) reduces to \(\text{HALT}_{\text{TM}}\), and therefore (since \(A_{\text{TM}}\) is undecidable), \(\text{HALT}_{\text{TM}}\) must be undecidable.

Assume that \(M_{\text{HALT}}\) is a machine that decides \(\text{HALT}_{\text{TM}}\).

**Goal:** Define decider for \(A_{\text{TM}}\) using \(M_{\text{HALT}}\) as subroutine.

"On input \(<M,w>\) \hspace{1cm} \text{Want to accept if } w \text{ in } L(M), \text{ reject o.w.}
1. Run \(M_{\text{HALT}}\) on \(<M,w>\). If rejects, reject.
2. If accepts, run \(M\) on \(w\).
3. If accepts, accept; if rejects, reject."

**Claim:** this is a decider for \(A_{\text{TM}}\) so \(A_{\text{TM}}\) reduces to \(\text{HALT}_{\text{TM}}\).
Claim: $E_{TM}$ is undecidable. \hspace{2cm} (Theorem 5.2)

$E_{TM} = \{ <M> \mid M \text{ is a TM and } L(M) \text{ is empty} \}$

i.e. want to recognize codes of TMs that always reject/loop

Proof by reduction?

To use proof by reduction to prove that $E_{TM}$ is undecidable, we must reduce an undecidable set to $E_{TM}$
Claim: \( E_{TM} \) is undecidable.

Proof by reduction

- **Goal**: show that \( A_{TM} \) reduces to \( E_{TM} \).
  - i.e. Build an algorithm that uses a decider for \( E_{TM} \) as a subroutine and that decides \( A_{TM} \)

- **Assume**: have a TM, \( R \), that decides \( E_{TM} \)
- **Build**: new TM, \( M_{ATM} \), that decides \( A_{TM} \)
  - Always halts
  - Accepts iff input \(<M,w>\) and \( w \) is in \( L(M) \).
**Claim:** $E_{TM}$ is undecidable.

**Proof by reduction**

- **Goal:** show that $A_{TM}$ reduces to $E_{TM}$.

  - i.e. Build an algorithm that uses a decider for $E_{TM}$ as a subroutine and that decides $A_{TM}$

  **Assume:** have a TM, $R$, that decides $E_{TM}$

  **Build:** new TM, $M_{ATM}$, that decides $A_{TM}$

  - Always halts
  - Accepts iff input $<M,w>$ and $w$ is in $L(M)$.

**What's the input to $R$?**

A. $w$
B. $<M>$
C. $<M,w>$
D. $<M, <M> >$
E. I don't know.
Claim: $E_{TM}$ is undecidable.

Proof by reduction

- **Assume**: have a TM, $R$, that decides $E_{TM}$
- **Build**: new TM, $M_{ATM}$, that decides $A_{TM}$
  - Always halts
  - Accepts iff input $<M, w>$ and $w$ is in $L(M)$.
- **Define** "On input $<M, w>$:
  1. Run $R$ on input $<M>$. If rejects, reject.
  2. If accepts, run $M$ on input $w$.
     a. If accepts, accept; if reject, reject."

Does this machine work? Always halt? Recognize $A_{TM}$?
Claim: $E_{TM}$ is undecidable.

Proof by reduction

- **Assume**: have a TM, $R$, that decides $E_{TM}$
- **Build**: new TM, $M_{ATM}$, that decides $A_{TM}$
  - Always halts
  - Accepts iff input $<M,w>$ and $w$ is in $L(M)$.
- **Define** "On input $<M,w>$:
  1. Run $R$ on ??

Need to ignore what $M$ does on inputs other than $w$ ... use an AUXILIARY MACHINE
Claim: $E_{TM}$ is undecidable.

Proof by reduction

- **Assume**: have a TM, $R$, that decides $E_{TM}$
- **Build**: new TM, $M_{ATM}$, that decides $A_{TM}$
  - Always halts
  - Accepts iff input $<M,w>$ and $w$ is in $L(M)$.

- **Define** "On input $<M,w>$:
  1. First, build TM $X$ = "On input $x$, ignore $x$ and simulate $M$ on $w$."
  2. Run $R$ on $<X>$.
     a. If accepts, reject; if rejects; accept."

Need to ignore what $M$ does on inputs other than $w$ ... use an AUXILIARY MACHINE
Claim: \( E_{\text{TM}} \) is undecidable.

Proof by reduction

- **Assume**: have a TM, R, that decides \( E_{\text{TM}} \)
- **Build**: new TM, \( M_{\text{ATM}} \), that decides \( A_{\text{TM}} \)
  - Always halts
  - Accepts iff input \(<M,w>\) and \( w \) is in \( L(M) \).

- **Define** "On input \(<M,w>\)"
  1. First, build TM \( X = \) "On input \( x \), ignore \( x \) and simulate \( M \) on \( w \)."
  2. Run R on \(<X>\).
     a. If accepts, reject; if rejects; accept.

For a given \(<M,w>\), what's \( L(X) \)?

A. \( \{ w \} \)

B. \( w \)

C. \( \{ x \mid x \neq w \} \)

D. \( \Sigma^* \)

E. The empty set.
Claim: $E_{TM}$ is undecidable.

**Proof by reduction**

- **Assume**: have a TM, R, that decides $E_{TM}$
- **Build**: new TM, $M_{ATM}$, that decides $A_{TM}$
  - Always halts
  - Accepts iff input $<M,w>$ and $w$ is in $L(M)$.
- **Define** "On input $<M,w>$:
  1. First, build TM $X$ = "On input $x$, ignore $x$ and simulate $M$ on $w$."
  2. Run R on $<X>$.
     a. If accepts, reject; if rejects; accept."
- **Correctness**: ….
So far

<table>
<thead>
<tr>
<th>Decidable</th>
<th>Undecidable</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_{DFA}$</td>
<td>$A_{TM}$</td>
</tr>
<tr>
<td>$E_{DFA}$</td>
<td>$\text{HALT}_{TM}$</td>
</tr>
<tr>
<td>$\text{EQ}_{DFA}$</td>
<td>$E_{TM}$</td>
</tr>
</tbody>
</table>
General approach

To prove that \( \{<M> \mid M \text{ is a TM and } L(M) \text{ has property } P\} \) is undecidable

- Assume **towards a contradiction** that \( R \) is a decider for \( \{<M> \mid M \text{ is a TM and } L(M) \text{ has } P\} \).
- Build decider for \( A_{\text{TM}} \) by: "On input \( <M,w> \)
  1. Construct a new TM \( X \) such that \( X \) has \( P \) iff \( w \) in \( L(M) \)
  2. Run \( R \) on \( <X> \): if accepts, accept; if rejects, reject."

Note: sometimes easier to build \( X \) so that \( X \) has \( P \) iff \( w \) not in \( L(M) \)
Puzzle

Claim: Exactly one of $E_{TM}$ and its complement is recognizable.

Proof:

Why not both?
Which is?