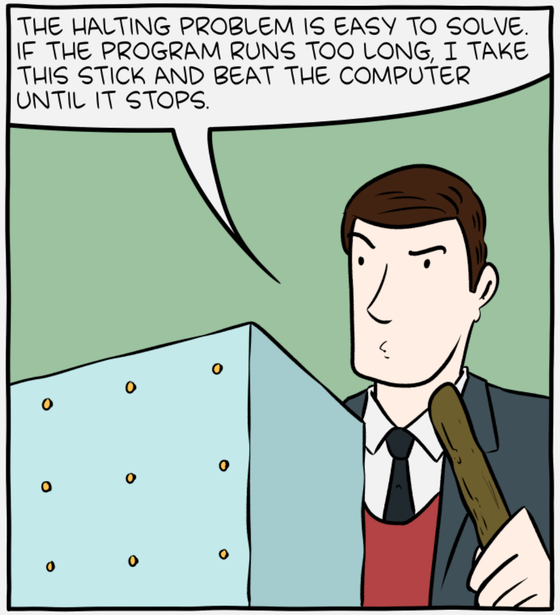
CS622

Reducibility by “Modifying the TM”

Friday, April 26, 2024

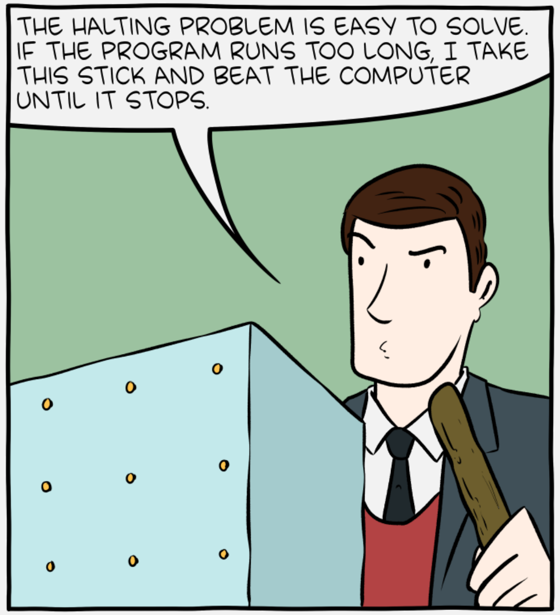


What if Alan Turing had been an engineer?

Announcements

- HW 10 out
  - Due Wed 5/1 12pm noon

- 5/1: HW 11 out
- 5/8: HW 11 in, HW 12 out
- 5/8: last lecture
- 5/15: HW 12 in (no exceptions)



What if Alan Turing had been an engineer?

Summary: The Limits of Algorithms

- $A_{\text{DFA}} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts input string } w\}$ — Decidable
- $A_{\text{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}$ — Decidable
- $A_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$ — **Undecidable**
- $HALT_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$ — **Undecidable**

Similar languages

It's straightforward to use hypothetical $HALT_{\text{TM}}$ decider to create A_{TM} decider

Summary: The Limits of Algorithms

- $A_{\text{DFA}} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts input string } w\}$ — Decidable
- $A_{\text{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}$ — Decidable
- $A_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$ — **Undecidable**
- $HALT_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$ — **Undecidable**
- $E_{\text{DFA}} = \{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$ — Decidable
- $E_{\text{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}$ — Decidable
- next: $E_{\text{TM}} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$ — **Undecidable**

Not as similar languages

How can we use a hypothetical E_{TM} decider to create A_{TM} or $HALT_{\text{TM}}$ decider?

Examples Table(s) are critical here!

Undecidability Proof Technique #2

Reducibility: Modifying the TM

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

Thm: E_{TM} is undecidable

Proof, by **contradiction**:

- Assume E_{TM} has ***decider*** R; use it to create *decider* for A_{TM}:

S = "On input $\langle M, w \rangle$, an encoding of a TM M and a string w:

- Run R on input $\langle M \rangle$
- If R accepts, *reject* (because it means $\langle M \rangle$ doesn't accept anything)
- if R rejects, then **???** ($\langle M \rangle$ accepts something, but is it *w*???)

R doesn't help all cases

"expected" result

Let $\langle M, w \rangle$ be a string where:
- M is some TM and
- w is some string

Example Table for A_{TM} decider S

String	M on w	R on $\langle M \rangle$	S on $\langle M, w \rangle$	In lang A_{TM}?
$\langle M, w \rangle$	Accept	Reject, $L(M)$=??	??	Yes
$\langle M, w \rangle$	Reject	Accept, $L(M)=\{\}$	Reject	No
$\langle M, w \rangle$	Loop	Accept, $L(M)=\{\}$	Reject	No

"Problem" case, use R to help

no w

Reducibility: Modifying the TM

$E_{TM} = \{\langle M \rangle | \ M \text{ is a TM and } L(M) = \emptyset\}$

Thm: E_{TM} is undecidable

Proof, by **contradiction**:

- Assume E_{TM} has *decider* R; use it to create *decider* for A_{TM}:

S = "On input $\langle M, w \rangle$, an encoding of a TM M and a string w:

  - Run R on input $\langle M \rangle$
  - If R accepts, *reject* (because it means $\langle M \rangle$ doesn't accept anything)
  - if R rejects, then **???** ($\langle M \rangle$ accepts something, but is it w???)

- Idea: Wrap $\langle M \rangle$ in a new TM that can *only* (maybe) accept w:

$L(M_1)$ depends on M and w!
If M accepts w, $L(M_1) = \{w\}$
else $L(M_1) = \{\}$

M_1 = "On input x:

1. If $x \neq w$, *reject*.
2. If $x = w$, run M on input w and *accept* if M does."

Input not w, always reject

Input is w, maybe accept

M_1 accepts w if M does

Reducibility: Modifying the TM

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

Thm: E_{TM} is undecidable

Proof, by contradiction:

- Assume E_{TM} has *decider* R; use it to create *decider* for A_{TM}:

  $S =$ "On input $\langle M, w \rangle$, an encoding of a TM M and a string w:

String x	M on w	M_1 on x	In lang $\{w\} \cap L(M)$?
w	Accept	Accept	Yes (lang = $\{w\}$)
w	Reject	Reject	No (lang = $\{\}$)
not w	-	Reject	No (lang = $\{\}$ or $\{w\}$)

Example Table for M_1

$L(M_1)$ depends on M and w!
If M accepts w, $L(M_1) = \{w\}$
else $L(M_1) = \{\}$

- Idea: Wrap $\langle M \rangle$ in a new TM that can *only* (maybe) accept w:

$M_1 =$ "On input x:
1. If $x \neq w$, *reject*.
2. If $x = w$, run M on input w and *accept* if M does."

Reducibility: Modifying the TM

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

Thm: E_{TM} is undecidable

Proof, by contradiction:

- Assume E_{TM} has *decider* R; use it to create *decider* for A_{TM}:

$S =$ "On input $\langle M, w \rangle$, an encoding of a TM M and a string w:

Example Table for M_1

String x	M on w	M_1 on x	In lang $\{w\} \cap L(M)$?
w	Accept	Accept	Yes (lang = $\{w\}$)
w	Reject	Reject	No (lang = $\{\}$)
not w	-	Reject	No (lang = $\{\}$ or $\{w\}$)

$L(M_1)$ depends on M and w!
If M accepts w, $L(M_1) = \{w\}$
else $L(M_1) = \{\}$

- Idea: W

Example Table for A_{TM} decider S

String	M on w	R on $\langle M \rangle$	S on $\langle M, w \rangle$	In lang A_{TM}?
$\langle M, w \rangle$	Accept	Reject, $L(M)$=??	??	Yes
$\langle M, w \rangle$	Reject	Accept, $L(M)=\{\}$	Reject	No
$\langle M, w \rangle$	Loop	Accept, $L(M)=\{\}$	Reject	No

Reducibility: Modifying the TM

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

Thm: E_{TM} is undecidable

Proof, by contradiction:

- Assume E_{TM} has *decider* R; use it to create *decider* for A_{TM}:

$S =$ "On input $\langle M, w \rangle$, an encoding of a TM M and a string w:

String x	M on w	M_1 on x	In lang $\{w\} \cap L(M)$?
w	Accept	Accept	Yes (lang = $\{w\}$)
w	Reject	Reject	No (lang = {})
not w	-	Reject	No (lang = {} or $\{w\}$)

Example Table for M_1

$L(M_1)$ depends on M and w!
If M accepts w, $L(M_1) = \{w\}$
else $L(M_1) = \{\}$

- Idea: W

String	M on w	R on $\langle M_1 \rangle$	S on $\langle M, w \rangle$	In lang A_{TM}?
$\langle M, w \rangle$	Accept	**Reject**, $L(M_1)$={w}	**Accept**	Yes
$\langle M, w \rangle$	Reject	Accept, $L(M_1)$={}	Reject	No
$\langle M, w \rangle$	Loop	Accept, $L(M_1)$={}	Reject	No

Example Table for A_{TM} decider S

Reducibility: Modifying the TM

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

Thm: E_{TM} is undecidable

Proof, by **contradiction**:

- Assume E_{TM} has *decider* R; use it to create *decider* for A_{TM}:

S = "On input $\langle M, w \rangle$, an encoding of a TM M and a string w:

First, construct M_1

- Run R on input $\langle M_1 \rangle$

Note: M_1 is only used as arg to R; it's never run (avoiding loop)!

- If R accepts, *reject* (because it means $\langle M \rangle$ doesn't accept w)
- if R rejects, then *accept* ($\langle M \rangle$ accepts w

$L(M_1)$ depends on M and w!
If M accepts w, $L(M_1) = \{w\}$
else $L(M_1) = \{\}$

- Idea: Wrap $\langle M \rangle$ in a new TM that can *only* (maybe) accept w:

M_1 = "On input x:
1. If $x \neq w$, *reject*.
2. If $x = w$, run M on input w and *accept* if M does."

Reducibility: Modifying the TM

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

Thm: E_{TM} is undecidable

Proof, by contradiction:

This decider for A_{TM} cannot exist!

- Assume E_{TM} has *decider* R; use it to create *decider* for A_{TM}:

S = "On input $\langle M, w \rangle$, an encoding of a TM M and a string w:

First, construct M_1

- Run R on input $\langle M_1 \rangle$
- If R accepts, *reject* (because it means $\langle M \rangle$ doesn't accept w)
- if R rejects, then *accept* ($\langle M \rangle$ accepts w)

- Idea: Wrap $\langle M \rangle$ in a new TM that can *only* (maybe) accept w:

M_1 = "On input x:
1. If $x \neq w$, *reject*.
2. If $x = w$, run M on input w and *accept* if M does."

Summary: The Limits of Algorithms

- $A_{\text{DFA}} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts input string } w\}$ — Decidable
- $A_{\text{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}$ — Decidable
- $A_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$ — **Undecidable**
- $E_{\text{DFA}} = \{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$ — Decidable
- $E_{\text{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}$ — Decidable
- $E_{\text{TM}} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$ — **Undecidable**
- $EQ_{\text{DFA}} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$ — Decidable
- $EQ_{\text{CFG}} = \{\langle G, H\rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$ — **Undecidable**
- next: $EQ_{\text{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$ — **Undecidable**

needs

Reduce to something else: EQ_{TM} is undecidable

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

Proof, by **contradiction**:

- Assume: EQ_{TM} has *decider* R; use it to create *decider* for ~~A_{TM}~~ E_{TM}:

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

S = "On input $\langle M\rangle$, where M is a TM:

1. Run R on input $\langle M, M_1\rangle$, where M_1 is a TM that rejects all inputs.
2. If R accepts, *accept*; if R rejects, *reject*."

Reduce to something else: EQ_{TM} is undecidable

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

Proof, by **contradiction**:

- Assume: EQ_{TM} has *decider* R; use it to create *decider* for E_{TM}:

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

~~S = "On input $\langle M\rangle$, where M is a TM:~~

~~1. Run R on input $\langle M, M_1\rangle$, where M_1 is a TM that rejects all inputs.~~
~~2. If R accepts, *accept*; if R rejects, *reject*."~~

- But E_{TM} is undecidable!

Summary: Undecidability Proof Techniques

- Proof Technique #1:
  - Use hypothetical decider to implement impossible A_{TM} decider
  - Example Proof: $HALT_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$

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

Reduce

- Proof Technique #2:
  - Use hypothetical decider to implement impossible A_{TM} decider
  - But first modify the input M
  - Example Proof: $E_{\text{TM}} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$

Reduce

Can also combine these techniques

- Proof Technique #3:
  - Use hypothetical decider to implement non-A_{TM} impossible decider
  - Example Proof: $EQ_{\text{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$

Summary: Decidability and Undecidability

- $A_{\text{DFA}} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts input string } w\}$ — Decidable
- $A_{\text{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}$ — Decidable
- $A_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$ — **Undecidable**
- $E_{\text{DFA}} = \{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$ — Decidable
- $E_{\text{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}$ — Decidable
- $E_{\text{TM}} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$ — **Undecidable**
- $EQ_{\text{DFA}} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$ — Decidable
- $EQ_{\text{CFG}} = \{\langle G, H\rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$ — **Undecidable**
- $EQ_{\text{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$ — **Undecidable**

Also Undecidable …

next

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

Undecidability Proof Technique #2:
Modify input TM M

Thm: $REGULAR_{\text{TM}}$ is undecidable

$REGULAR_{\text{TM}} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$

Proof, by **contradiction**:

- Assume: $REGULAR_{\text{TM}}$ has *decider* R; use it to create *decider* for A_{TM}:

S = "On input $\langle M, w\rangle$, an encoding of a TM M and a string w:

- First, construct M_2 (??)
- Run R on input $\langle M_2\rangle$
- If R accepts, *accept*; if R rejects, *reject*

Want: $L(M_2)$ =
- **regular**, if M accepts w
- **nonregular**, if M does not accept w

Thm: $REGULAR_{\text{TM}}$ is undecidable (continued)

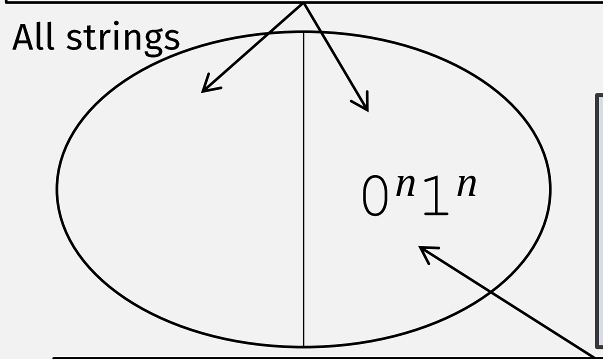
$REGULAR_{\text{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$

M_2 = "On input x:
1. If x has the form 0^n1^n, *accept*.
2. If x does not have this form, run M on input w and *accept* if M accepts w."

Always accept strings 0^n1^n
$L(M_2)$ = **nonregular,** so far

If M accepts w,
accept everything else,
so $L(M_2) = \Sigma^* =$ **regular**

if M does not accept w, M_2 accepts all strings (**regular** lang)



if M accepts w, M_2 accepts this **nonregular** lang

Want: $L(M_2)$ =
- **regular,** if M accepts w
- **nonregular,** if M does not accept w

Also Undecidable …

Seems like no algorithm can compute
anything about
the language of a Turing Machine,
i.e., about the runtime behavior of programs …

- $REGULAR_{\text{TM}}$ = {<M> | M is a TM and $L(M)$ is a regular language}
- $CONTEXTFREE_{\text{TM}}$ = {<M> | M is a TM and $L(M)$ is a CFL}
- $DECIDABLE_{\text{TM}}$ = {<M> | M is a TM and $L(M)$ is a decidable language}
- $FINITE_{\text{TM}}$ = {<M> | M is a TM and $L(M)$ is a finite language}

An Algorithm About Program Behavior?

```
main()
{
    printf("hello, world\n");
}
```

Write a program that,
given another program as its argument,
returns TRUE if that argument prints
“**`Hello, World!`**”

TRUE

Fermat's Last Theorem
(unknown for ~350 years,
solved in 1990s)

```
main()
{
  If x^n + y^n = z^n, for any integer n > 2
    printf("hello, world\n");
}
```

Write a program that,
given another program as its argument,
returns TRUE if that argument prints
"`Hello, World!`"

?????

Also Undecidable …

Seems like no algorithm can compute
anything about
the language of a Turing Machine,
i.e., about the runtime behavior of programs …

- $REGULAR_{TM} = \{<M> \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$
- $CONTEXTFREE_{TM} = \{<M> \mid M \text{ is a TM and } L(M) \text{ is a CFL}\}$
- $DECIDABLE_{TM} = \{<M> \mid M \text{ is a TM and } L(M) \text{ is a decidable language}\}$
- $FINITE_{TM} = \{<M> \mid M \text{ is a TM and } L(M) \text{ is a finite language}\}$
- …

Rice's Theorem

- $ANYTHING_{TM} = \{<M> \mid M \text{ is a TM and "… } \textbf{anything} \text{ …" about } L(M)\}$

Rice's Theorem: $ANYTHING_{\text{TM}}$ is Undecidable

$ANYTHING_{\text{TM}}$ = {<M> | M is a TM and … **anything** … about $L(M)$}

- "… **Anything** …", more precisely:

  For any M_1, M_2,
  - if $L(M_1) = L(M_2)$
  - then $M_1 \in ANYTHING_{\text{TM}} \Leftrightarrow M_2 \in ANYTHING_{\text{TM}}$

- Also, "… **Anything** …"must be "non-trivial":
  - $ANYTHING_{\text{TM}}$!= {}
  - $ANYTHING_{\text{TM}}$!= set of all TMs

Rice's Theorem: $ANYTHING_{\text{TM}}$ is Undecidable

$ANYTHING_{\text{TM}}$ = {<M> | M is a TM and … **anything** … about $L(M)$}

Proof by **contradiction**

- Assume some language satisfying $ANYTHING_{\text{TM}}$ has a decider R.
  - Since $ANYTHING_{\text{TM}}$ is non-trivial, then there exists $M_{ANY} \in ANYTHING_{\text{TM}}$
  - Where R accepts M_{ANY}
- Use R to create decider for A_{TM}:

On input <M, w>:

- Create M_w:

M_w = on input x:
- Run M on w
  - If M rejects w: reject x
  - If M accepts w:
    Run M_{ANY} on x and accept if it accepts, else reject

If M accepts w: $M_w = M_{\text{ANY}}$
If M doesn't accept w: M_w accepts nothing

These two cases must be different, (so R can distinguish when M accepts w)

Wait! What if the TM that accepts nothing is in $ANYTHING_{\text{TM}}$!

- Run R on M_w
  - If it accepts, then $M_w = M_{ANY}$, so M accepts w, so accept
  - Else reject

Proof still works! Just use the complement of $ANYTHING_{\text{TM}}$ instead!

Rice’s Theorem Implication

$\{<M> \mid M \text{ is a TM that installs malware}\}$

Undecidable!
(by Rice’s Theorem)