Halting Problem
Part 3

Adam Wyner

CS3518, Spring 2017

University of Aberdeen

Back to the Halting Problem

$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$

Recall:

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

Theorem: $HALT_{TM}$ is undecidable

Proof strategy (called proof by **reduction**):

- We use the undecidability of A_{TM} to prove the undecidability of $HALT_{TM}$

Halting Problem (Cont'd)

- Assume (to obtain contradiction) R decides $HALT_{TM}$
- S = On input $\langle M, w \rangle$ where M is a TM and w is a string:
  1. Run R on input $\langle M, w \rangle$.
  2. If R rejects, reject. i.e. {$\langle M, w \rangle$ does not halt}
  3. If R accepts, i.e. {$\langle M, w \rangle$ halts},
     then simulate M on w until it halts.
  4. If M accepts w, accept;
     If M rejects w, reject.
- By assumption, R decides $HALT_{TM}$ so S decides A_{TM}. But A_{TM} is undecidable so R doesn't decide $HALT_{TM}$
- It follows that $HALT_{TM}$ is undecidable!

Halting Problem (Cont'd)

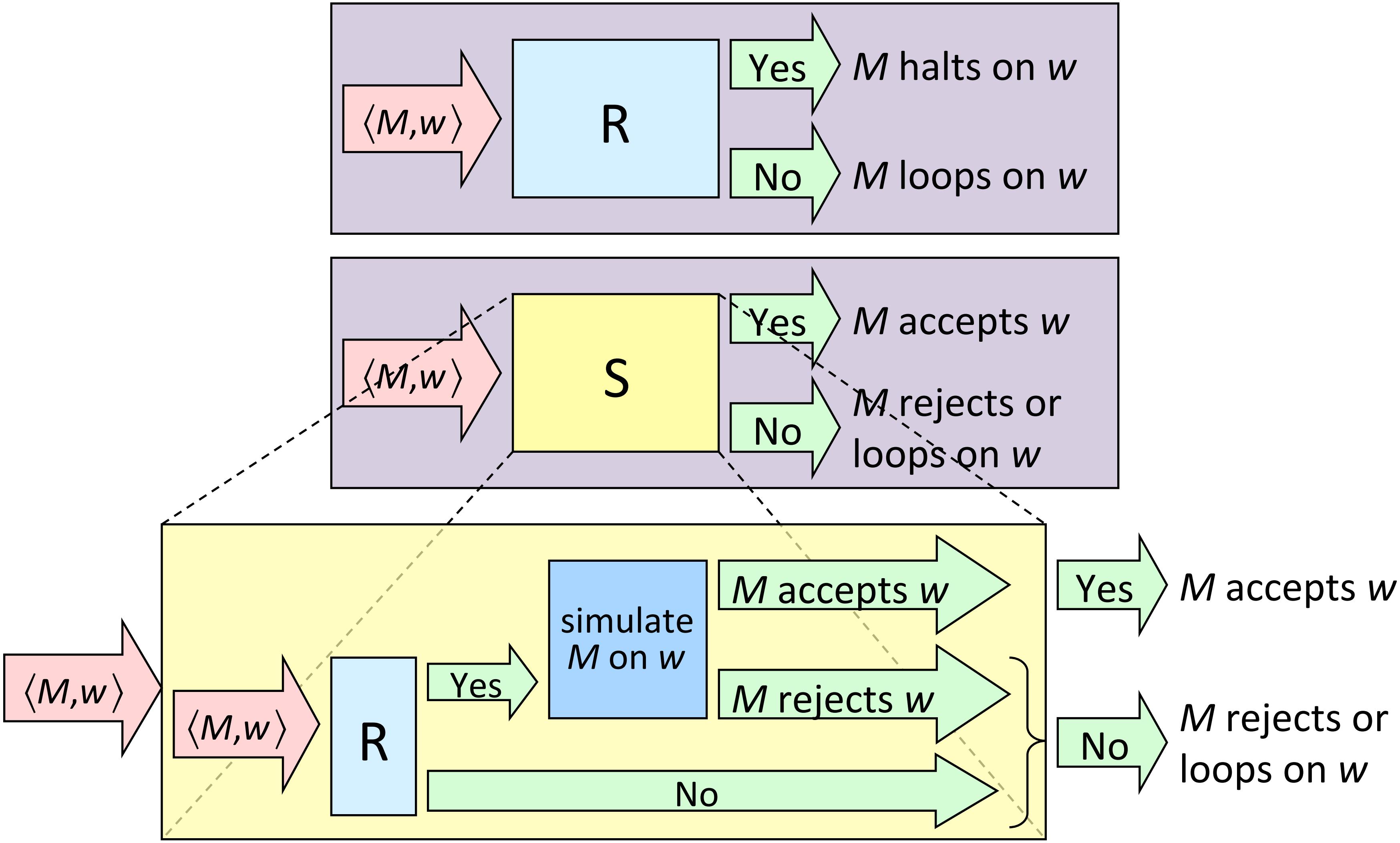
A_{TM} can be seen as consisting of two problems:

Given $\langle M, w \rangle$,

1. Decide whether $\langle M, w \rangle$ halts (yes/no)
2. a. If yes then decide whether M accepts w.

   b. If no then reject.

Halting Problem (Cont'd)



Halting Problem (summary)

- A_{TM} "reduces" to $HALT_{TM}$
- Since A_{TM} is undecidable, $HALT_{TM}$ is also undecidable
- Reduction (restating an unknown problem in terms of a known problem and solution) is a key strategy in
  - the theory of computability
  - the theory of computational complexity

A Turing-Unrecognisable Language

- Definition: A language is co-Turing-recognisable iff it is the complement of a Turing-recognisable language
- Theorem: A language is decidable if and only if it is Turing-recognisable <u>and</u> co-Turing-recognisable

A Turing-Unrecognisable Language

Some things that follow:

- The complement of A_{TM} is not Turing-recognisable
  - We have seen: A_{TM} is Turing-recognisable
  - If the complement of A_{TM} were also Turing-recognisable, A_{TM} would be decidable (and it isn't)

- Review question:

  Consider L = {<M,w> | M loops on w}.

  Does there exists a TM for L that is a recogniser?

A Turing-Unrecognisable Language

- Review question:
  Consider L = {<M,w> | M loops on w}

  Does there exists a TM for L that is a recogniser?

- No!

  Suppose X was a recogniser for L.
  Consider any M and any w. If M loops on w then X accepts <M,w> (in finitely many steps). If M does not loop on w then M either accepts or rejects w (in finitely many steps). Hence, a decider for A_{TM} can be constructed as follows:

A Turing-Unrecognisable Language

Suppose X recognised L. Then K would decide A_{TM}:

K = On input ⟨M, w⟩ where M is a TM and w is a string:

1a. Run X on ⟨M, w⟩. (a recogniser!)

1b. Simultaneously, Run U on ⟨M, w⟩. (a recogniser!)

Either M loops on w or M does not loop on w.

If M loops on w then X accepts ⟨M,w⟩ (in finite time)

so K rejects ⟨M,w⟩

If M doesn't loop on w, U accepts or rejects ⟨M,w⟩

If U accepts ⟨M,w⟩ then K accepts ⟨M,w⟩

If U rejects ⟨M,w⟩ then K rejects ⟨M,w⟩

A Turing-Unrecognisable Language

K = On input ⟨M, w⟩ where M is a TM and w is a string:

1a. Run X on ⟨M, w⟩. (a recogniser!)

1b. Simultaneously, Run U on ⟨M, w⟩. (a recogniser!)

Either M loops on w or M does not loop on w.

If M loops on w then X accepts ⟨M,w⟩ (in finite time)

so K rejects ⟨M,w⟩

If M doesn't loop on w, U accepts or rejects ⟨M,w⟩

If U accepts ⟨M,w⟩ then K accepts ⟨M,w⟩

If U rejects ⟨M,w⟩ then K rejects ⟨M,w⟩

- But A_{TM} is undecidable. It follows that *X* cannot recognise L. So, L cannot be recognised

Reduction

- Reduction is an important technique, not just in proving decidability (and other computability) results, but also in proving complexity results (e.g., “solving problem X takes exponential time”)
- Another example of using reduction to prove the undecidability of a problem:

Problem No. 2

- Problem: determine if a Turing machine does not accept any input, that is, its language is empty (Compare the old E_{DFA})
- Let $E_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$
- Theorem: E_{TM} is undecidable

- Proof strategy: proof by reduction again
  – Assume that E_{TM} is decidable and
  – Show that A_{TM} is decidable: a contradiction
- Let R be a TM that decides E_{TM}
- We use R to build S that decides A_{TM}

Problem No. 2 (Cont'd)

- We run R on a modification of $\langle M \rangle$:
  - We modify $\langle M \rangle$ to ensure M rejects all strings except w
  - On input w, M works as usual

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.

- The only string M_1 may now accept is w :

$M_1 = \{w\}$ or $M_1 = \Phi$

$L(M_1)$ is non-empty $\Leftrightarrow$ M_1 accepts w

$\Leftrightarrow$ M accepts w

Problem No. 2 (Cont'd)

Proof that E_{TM} is undecible. Assume TM R decides E_{TM}

Build TM S (using R) that decides A_{TM}:

S = On input $\langle M, w \rangle$ where M is a TM and w is a string:

1. Use M and w to build M_1 as explained
2. Run R on input $\langle M_1 \rangle$.
3. If R accepts (so L(M1) = Φ), reject (M rejects w)
   If R rejects (so L(M1) ≠ Φ), accept (M accepts w).

- S would decide A_{TM} but that's not possible
- Hence, E_{TM} must be undecidable

Problem No. 2 (Cont'd)