Lecture 17: Proving Undecidability

Proofs of Decidability

How can you prove a language is \textit{decidable}?

What Decidable Means

A language \( L \) is \textbf{decidable} if there exists a TM \( M \) such that for all strings \( w \):

- If \( w \in L \), \( M \) enters \( q_{\text{Accept}} \).
- If \( w \notin L \), \( M \) enters \( q_{\text{Reject}} \).

To prove a language is decidable, we can show how to construct a TM that decides it.

For a correct proof, need a convincing argument that the TM always eventually accepts or rejects any input.

Proofs of Undecidability

How can you prove a language is \textbf{undecidable}?

Proofs of Undecidability

To prove a language is \textbf{undecidable}, need to show there is no Turing Machine that can decide the language.

This is hard: requires reasoning about all possible TMs.

Proof by Reduction

1. We know \( X \) does not exist. (e.g., \( X \) is a TM that can decide \( A_{\text{TM}} \))

2. Assume \( Y \) exists. (e.g., \( Y \) is a TM that can decide \( B \))

3. Show how to use \( Y \) to make \( X \).

4. Since \( X \) does not exist, but \( Y \) could be used to make \( X \), then \( Y \) must not exist.
Lecture 17: Proving Undecidability

Reduction Proofs

A reduces to B means that can solve B can be used to make that can solve A.

Hence, A is not a harder problem than B.

The name “reduces” is confusing: it is in the opposite direction of the making.

Converse?

A reduces to B that can solve B can be used to make that can solve A

A is not a harder problem than B.

Does this mean B is as hard as A?

No! Y can be any solver for B. X is one solver for A.

There might be easier solvers for A.

Reduction Pitfalls

- Be careful: the direction matters a great deal
  – Showing a machine that decides B can be used to build a machine that decides A shows that A is not harder than B.
  – To show equivalence, need reductions in both directions.
- The transformation must involve only things you know you can do: otherwise the contradiction might be because something else doesn’t exist.

What does can do mean here?

What “Can Do” Means

- The transformations in a reduction proof are limited by what you are proving
- For undecidability proofs, you are proving something about all TMs: the reduction transformations are anything that a TM can do that is guaranteed to terminate
- For complexity proofs (later), you are proving something about how long it takes: the time it takes to do the transformation is limited

The Halting Problem

\[ \text{HALT}_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM description and } M \text{ halts on input } w \} \]

Alternate statement as problem:
Input: A TM M and input w
Output: True if M halts on w, otherwise False.

Is \( \text{HALT}_{TM} \) Decidable?

- Possible “Yes” answer: Prove it is decidable
  - Design a TM that can decide \( \text{HALT}_{TM} \)
- Possible “No” answer: prove it is undecidable
  - Show that no TM can decide \( \text{HALT}_{TM} \)
  - Show that a TM that could decide \( \text{HALT}_{TM} \) could be used to decide \( A_{TM} \) which we already proved is undecidable.
Acceptance Language

\[ A_{TM} = \{ <M, w> | M \text{ is a TM description and } M \text{ accepts input } w \} \]

We proved \( A_{TM} \) is undecidable last class.

Since we know \( A_{TM} \) is undecidable, we can show a new language \( B \) is undecidable if a machine that can decide \( B \) could be used to build a machine that can decide \( A_{TM} \).

Reducing \( A_{TM} \) to \( HALT_{TM} \)

\[ HALT_{TM} = \{ <M, w> | M \text{ is a TM description and } M \text{ halts on input } w \} \]

\[ A_{TM} = \{ <M, w> | M \text{ is a TM description and } M \text{ accepts input } w \} \]

\( <M, w> \) is in \( A_{TM} \) if and only if:
- \( M \) halts on input \( w \)
- and when \( M \) halts it is in accepting state.

Deciding \( A_{TM} \)

- Assume \( HALT_{TM} \) is decidable.
- Then some TM \( R \) can decide \( HALT_{TM} \).
- We can use \( R \) to build a machine that decides \( A_{TM} \):
  - Simulate \( R \) on \( <M, w> \)
  - If \( R \) rejects, it means \( M \) doesn’t halt: reject.
  - If \( R \) accepts, it means \( M \) halts:
    - Simulate \( M \) on \( w \), accept/reject based on \( M \)’s accept/reject.

Since any TM that decides \( HALT_{TM} \) could be used to build a TM that decides \( A_{TM} \) (which we know is impossible) this proves that no TM exists that can decide \( HALT_{TM} \).

Equivalence of DFA \( D \) and TM \( M \)

\[ EQ_{DM} = \{ <D, T> | D \text{ is a DFA description, } T \text{ is a TM description and } L(T) = L(D) \} \]

Is \( EQ_{DM} \) decidable?

\( EQ_{DM} \) is Undecidable

- Suppose \( R \) decides \( EQ_{DM} \).
- Can we use \( R \) to decide \( HALT_{TM} \)?

\[ HALT_{TM} = \{ <M, w> | M \text{ is a TM description and } M \text{ halts on input } w \} \]

\[ EQ_{DM} = \{ <D, T> | D \text{ is a DFA description, } T \text{ is a TM description and } L(T) = L(D) \} \]

Given \( M \) and \( w \), how can you construct \( D \) and \( T \) so \( R(<D, T>) \) tells you if \( M \) halts on \( w \)?

\( EQ_{DM} \) is Undecidable

- Suppose \( R \) decides \( EQ_{DM} \).
- Can we use \( R \) to decide \( HALT_{TM} \)?

\[ EQ_{DM} = \{ <D, T> | D \text{ is a DFA description, } T \text{ is a TM description and } L(T) = L(D) \} \]

\( D \) = DFA that accepts all strings.

\( T \) = TM that ignores input and simulates \( M \) on \( w \), and if simulated \( M \) accepts or rejects, accept.
EQ_DM is Undecidable

\[ \text{HALT}_{TM} = \{ <M, w> \mid M \text{ is a TM description and } M \text{ halts on input } w \} \]

\[ \text{EQ}_{DM} = \{ <D, T> \mid D \text{ is a DFA description, } T \text{ is a TM description and } L(T) = L(D) \} \]

\[ D = \text{DFA that rejects all strings.} \]

\[ T = \text{TM that ignores input and simulates } M \text{ on } w, \text{ and if simulated } M \text{ accepts or rejects, reject.} \]

Rice’s Theorem
Henry Gordon Rice, 1951

Any nontrivial property about the language of a Turing machine is undecidable.

Nontrivial means the property is true for some TMs, but not for all TMs.

Which of these are Undecidable?

- Does TM \( M \) accept any strings? \( \text{Undecidable} \)
- Does TM \( M \) accept all strings? \( \text{Undecidable} \)
- Does TM \( M \) accept “Hello”? \( \text{Undecidable} \)
- Does TM \( M_1 \) accept more strings than TM \( M_2 \)? \( \text{Undecidable} \)
- Does TM \( M \) take more than 1000 steps to process input \( w \)? \( \text{Decidable} \)
- Does TM \( M_1 \) take more steps than TM \( M_2 \) to process input \( w \)? \( \text{Undecidable} \)

Next Class

- Examples of some problems we actually care about that are undecidable
- Are there any problems that we don’t know if they are decidable or undecidable?
- PS5 Due next Tuesday (April 1)
- Exam 2 in two weeks