CMPSC230
Lesson 14: Undecidable Problems
Prof. John Wenskovitch
10/30/2014
Last Time

• Undecidable Problems
  – They exist!
  – The starter example
  – Non-Turing recognizable languages
  – The Halting Problem
Today

• More Undecidable Problems!
  – The reduction process
  – Example problems
  – New machine: LBA (linear bounded automaton)
The Halting Problem

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

- **Theorem:** \( \text{HALT}_{TM} \) is undecidable.
- **Proof:**
  - Assume to reach a contradiction that we have a TM \( R \) that decides \( \text{HALT}_{TM} \).
  - We will use \( R \) to construct \( S \), a TM that decides \( A_{TM} \).

\[ S = \text{“On input } \langle M, w \rangle, \text{ an encoding of a TM } M \text{ and a string } w:} \]
\[ 1. \text{ Run TM } R \text{ on input } \langle M, w \rangle \]
\[ 2. \text{ If } R \text{ rejects, reject.} \]
\[ 3. \text{ If } R \text{ accepts, simulate } M \text{ on } w \text{ until it halts.} \]
\[ 4. \text{ If } M \text{ has accepted, accept; if } M \text{ has rejected, reject.} \]

- If \( R \) decides \( \text{HALT}_{TM} \), then \( S \) decides \( A_{TM} \). Because \( A_{TM} \) is undecidable, \( \text{HALT}_{TM} \) must also be undecidable. ■
Emptiness Problem for Turing Machines

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

- **Theorem:** \( E_{TM} \) is undecidable.
- **Proof:**
  - Assume to reach a contradiction that we have a TM \( R \) that decides \( E_{TM} \).
  - We will use \( R \) to construct \( S \), a TM that decides \( A_{TM} \).

\( S = \text{"On input } \langle M, w \rangle \text{, an encoding of a TM } M \text{ and a string } w:\)

1. Use the description of \( M \) and \( w \) to construct a new TM \( M_1 \), which will reject all strings but \( w \), and on input \( w \) will work as usual:
   
   - \( M_1 = \text{"On input } x:\)
     1. If \( x \neq w \), reject.
     2. If \( x = w \), run \( M \) on input \( w \) and accept if \( M \) does."

2. Run \( R \) on input \( \langle M_1 \rangle \) and do opposite.

- The only string that \( M_1 \) can possibly accept is \( w \). Therefore, its language will be nonempty iff it accepts \( w \).
- If \( R \) decides \( E_{TM} \), then \( S \) decides \( A_{TM} \). Because \( A_{TM} \) is undecidable, \( E_{TM} \) must also be undecidable. ■
Can we decide if a Turing machine has an equivalent DFA?

\[ REGULAR_{TM} = \{ \langle M \rangle | M \text{ is a TM and } L(M) \text{ is a regular language} \} \]

- **Theorem:** \( REGULAR_{TM} \) is undecidable.
- **Proof:**
  - Assume to reach a contradiction that we have a TM \( R \) that decides \( REGULAR_{TM} \).
  - We will use \( R \) to construct \( S \), a TM that decides \( A_{TM} \).

\( S = \) “On input \( \langle M, w \rangle \), an encoding of a TM \( M \) and a string \( w \):
  1. Use the description of \( M \) and \( w \) to construct a new TM \( M_2 \):
     \( M_2 = \) “On input \( x \):
       1. If \( x \) has the form \( 0^n1^n \), **accept**.
       2. If \( x \) doesn’t take this form, run \( M \) on input \( w \) and **accept** if \( M \) accepts \( w \).”
  2. Run \( R \) on input \( \langle M_2 \rangle \) and do same.

- \( M_2 \) will recognize the clearly non-regular language \( 0^n1^n \) if \( M \) doesn’t accept \( w \), and recognizes the regular language \( \Sigma^* \) if \( M \) accepts \( w \).
- Note: \( M_2 \) is not constructed with the purpose of running it on some input; rather, it is constructed simply to feed its description into the decider for \( REGULAR_{TM} \).
- If \( R \) decides \( REGULAR_{TM} \), then \( S \) decides \( A_{TM} \). Because \( A_{TM} \) is undecidable, \( E_{TM} \) must also be undecidable. ■
Equivalence Problem for Turing Machines

\[ EQ_{TM} = \{ (M_1, M_2) | M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2) \} \]

- **Theorem:** \( EQ_{TM} \) is undecidable.
- **Proof:**
  - Assume to reach a contradiction that we have a TM \( R \) that decides \( EQ_{TM} \).
  - We will use \( R \) to construct \( S \), a TM that decides \( E_{TM} \).

\( S = \) “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 and do same.”

  - We know that \( L(M_1) \) is empty. Therefore, if we can decide that \( M \) and \( M_1 \) are equal, we can decide that \( M \) is empty.
  - If \( R \) decides \( EQ_{TM} \), then \( S \) decides \( E_{TM} \). Because \( E_{TM} \) is undecidable, \( EQ_{TM} \) must also be undecidable. ■
New Machine: LBA

• **Linear Bounded Automaton** – a restricted type of Turing machine, on which the tape head is not permitted to move off the portion of the tape containing the input.
  
  – If the machine tries to move its head off either end of the input, the head stays where it is, similar to the left end of an ordinary TM’s tape.
  
  – Despite the tape limitation, LBAs are still powerful:
    
    • The deciders for $A_{DFA}$, $A_{CFG}$, $E_{DFA}$, and $E_{CFG}$ are all LBAs.
    
    • Every CFL can be decided by an LBA.
$A_{LBA}$ is decidable!

- **Lemma:** Let $M$ be an LBA with $q$ states and $g$ symbols in the tape alphabet. There are exactly $qng^n$ distinct configurations of $M$ for a tape of length $n$.

- **Proof:** Remember that a configuration of $M$ is like a snapshot in the middle of a computation, consisting of the current state, the head position, and the contents of the tape. $M$ has $q$ states, the head can be in one of $n$ positions, and $g^n$ possible strings of tape symbols appear on the tape. The product of these three values is the total number of distinct configurations of $M$ with a tape of length $n$. 
$A_{LBA}$ is decidable!

• **Theorem:** $A_{LBA}$ is decidable.

• **Proof:**
  
  – We construct a machine $L$ to decide $A_{LBA}$:
    
    $L = \text{"On input } \langle M, w \rangle, \text{ where } M \text{ is an LBA and } w \text{ is a string:} $
    
    1. Simulate $M$ on $w$ for $qng^n$ steps, or until it halts.
    2. If $M$ has halted, do same. If $M$ has not halted, reject.”

  – If $M$ on $w$ has not halted within $qng^n$ steps, it must be repeating configurations, and therefore looping. Therefore, our algorithm can reject. ■
Accepting Computation History

• Let $M$ be a Turing machine and $w$ an input string. An accepting computation history for $M$ on $w$ is a sequence of configurations, $C_1, C_2, \ldots, C_l$, where $C_1$ is the start configuration of $M$ on $w$, $C_l$ is an accepting configuration of $M$, and each $C_i$ legally follows from $C_{i-1}$ according to the rules of $M$.

• A rejecting computation history for $M$ on $w$ is defined similarly, except that $C_l$ is a rejecting configuration.

• Computation histories are finite sequences – if $M$ does not halt on $w$, then no accepting or rejecting computation history exists for $M$ on $w$. 
Emptiness Problem for LBAs

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

- **Theorem:** \( E_{LBA} \) is undecidable.
- **Proof:**
  - Assume to reach a contradiction that we have a TM \( R \) that decides \( E_{LBA} \).
  - We will use \( R \) to construct \( S \), a TM that decides \( A_{TM} \).

\( S = \) “On input \( \langle M, w \rangle \), where \( M \) is a TM and \( w \) is a string:
  1. Construct an LBA \( B \) and string \( x \) from \( M \) and \( w \) such that \( B \) accepts \( x \) iff \( x \) is an accepting computation history for \( M \) on \( w \). (Assume that the input \( x \) is provided as ‘#’-delimited configurations on the tape.)
  2. Run \( R \) on input \( \langle B \rangle \) and do opposite.”

- If \( R \) accepts \( B \), then \( L(B) = \emptyset \); therefore, \( M \) has no accepting computation history of \( w \), so \( M \) doesn’t accept \( w \).
- If \( R \) decides \( E_{LBA} \), then \( S \) decides \( A_{TM} \). Because \( A_{TM} \) is undecidable, \( E_{LBA} \) must also be undecidable. ■
Any Questions?

NO HOMEWORK