CMPSC230
Lesson 13: Undecidability

Prof. John Wenskovitch
10/23/2014
Last Time

• Decidable Languages:
  – What is a decidable language?
  – What are some examples?
    • Acceptance problem for DFAs, NFAs, RegEx, CFGs
    • Emptiness problem for DFAs, NFAs, RegEx, CFGs
    • Equality problem for DFAs, NFAs, RegEx
Today

• Unsolvable Problems
  – They exist!
  – The starter example
  – Non-Turing recognizable languages
  – The Halting Problem
Acceptance Problem for Turing Machines

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

• We can definitely build a recognizer:
• Proof:

\[ U = \text{“On input } \langle M, w \rangle \text{, where } M \text{ is a TM and } w \text{ is a string:} \]
  1. Simulate \( M \) on \( w \).
  2. If \( M \) ever enters its accept state, accept; if \( M \) ever enters its reject state, reject.” ■

• Is it possible to build a decider?
Acceptance Problem for Turing Machines

• **Theorem:** $A_{TM}$ is undecidable.

• **Proof:**
  
  – Assume to reach a contradiction that Turing Machine $H$ is a decider for $A_{TM}$. On input $\langle M, w \rangle$, where $M$ is a TM and $w$ is a string, $H$ halts and **accepts** if $M$ accepts $w$. Furthermore, $H$ halts and **rejects** if $M$ rejects $w$.

  $$H(\langle M, w \rangle) = \begin{cases} 
  \text{accept} & \text{if } M \text{ accepts } w \\
  \text{reject} & \text{if } M \text{ doesn't accept } w 
  \end{cases}$$
Acceptance Problem for Turing Machines

• Construct a new TM $D$ with $H$ as a subroutine.
  
  – $D$ will call $H$ and do the opposite (reject if $H$ accepts, accept if $H$ rejects).

  – $D = \text{“On input } \langle M \rangle, \text{ where } M \text{ is a TM:}
    \begin{enumerate}
    \item Run $H$ on input $\langle M, \langle M \rangle \rangle$ and do opposite.”
    \end{enumerate}

  – Running a machine on its own description ($\langle M, \langle M \rangle \rangle$) is basically the same idea as running a program with itself as input.
Acceptance Problem for Turing Machines

\[ D(\langle M \rangle) = \begin{cases} 
  \text{accept} & \text{if } M \text{ doesn't accept } \langle M \rangle \\
  \text{reject} & \text{if } M \text{ accepts } \langle M \rangle 
\end{cases} \]

• OK, so what happens if we run D on its own description?

\[ D(\langle D \rangle) = \begin{cases} 
  \text{accept} & \text{if } D \text{ doesn't accept } \langle D \rangle \\
  \text{reject} & \text{if } D \text{ accepts } \langle D \rangle 
\end{cases} \]

• No matter what \( D \) does, it is forced to do the opposite, which is a contradiction. Thus, neither TM \( D \) nor TM \( H \) can exist. ■
A Quick Step-Through

• Assume that a TM $H$ decides $A_{TM}$.

• Use $H$ to build a TM $D$ that takes an input $\langle M \rangle$.
  – $D$ accepts its input $\langle M \rangle$ when $M$ does not accept its input $\langle M \rangle$.
  – $D$ rejects its input $\langle M \rangle$ when $M$ accepts its input $\langle M \rangle$.

• Finally, run $D$ on itself:
  – $H$ accepts $\langle M, w \rangle$ exactly when $M$ accepts $w$.
  – $D$ rejects $\langle M \rangle$ exactly when $M$ accepts $\langle M \rangle$.
  – $D$ rejects $\langle D \rangle$ exactly when $D$ accepts $\langle D \rangle$. 
Some languages are not Turing-recognizable

- **Proof:**
  - The set of all strings $\Sigma^*$ is countable for any alphabet $\Sigma$:
    - We can list all strings in $\Sigma^*$ by listing all strings of length 0, all strings of length 1, all strings of length 2, ...
  - The set of all Turing machines is countable, because each Turing machine $M$ has an encoding into a string $\langle M \rangle$. If we eliminate all strings that don’t produce legal Turing machines, we are left with the set of all legal Turing machines ($T$).
Some languages are not Turing-recognizable

– The set of all infinite binary sequences is uncountable:
  • Let $B$ be the set of all infinite binary sequences.
  • We can show that $B$ is uncountable with a diagonalization argument.

– Let $L$ be the set of all languages over alphabet $\Sigma$.

– $L$ has a one-to-one correspondence with $B$:
  • Each language $A$ in $L$ has a unique sequence in $B$.
  • If $B$ is uncountable, then $L$ is also uncountable.
  • If the number of languages ($L$) is uncountable, but the number of Turing machines ($T$) is countable, then there are more possible languages than possible Turing machines. ■
\( A_{\overline{TM}} \) is not Turing Recognizable

- **Theorem:** A language is Turing Decidable iff it is Turing Recognizable and its complement is Turing Recognizable.

- **Proof (\( \rightarrow \)):**
  - If language \( A \) is decidable, then it is recognizable.
  - If a TM \( M \) can decide whether or not a string is in \( A \), then it can also decide whether or not a string is in \( \overline{A} \).
$A_{TM}$ is not Turing Recognizable

• **Proof (←):**
  
  – If both $A$ and $\overline{A}$ are Turing-recognizable, let $M_1$ be the recognizer for $A$, and let $M_2$ be the recognizer for $\overline{A}$.
  
  – The following machine $M$ is a decider for $A$:

  \[
  M = \text{"On input } w:\n  \begin{align*}
  1. & \text{ Run both } M_1 \text{ and } M_2 \text{ on input } w \text{ in parallel.} \\
  2. & \text{ If } M_1 \text{ accepts, } \text{accept}; \text{ if } M_2 \text{ accepts, } \text{reject}.\n  \end{align*}
  \]

  – Every string $w$ is either in $A$ or $\overline{A}$; therefore either $M_1$ or $M_2$ must accept $w$. $M$ always halts, and hence is a decider. Therefore, $A$ is decidable. ■
\( A_{TM} \) is not Turing Recognizable

• **Theorem:** \( A_{TM} \) is not Turing recognizable

• **Proof:**
  
  – We know that \( A_{TM} \) is Turing recognizable.
  
  – If \( A_{TM} \) were also Turing recognizable, then the previous proof says that \( A_{TM} \) would become Turing decidable.
  
  – We have a proof that \( A_{TM} \) is not decidable; therefore \( A_{TM} \) must not be Turing recognizable.
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 \(A_{TM}\).

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

1. Run TM \(R\) on input \( \langle M, w \rangle \)
2. If \(R\) rejects, reject.
3. If \(R\) accepts, simulate \(M\) on \(w\) until it halts.
4. If \(M\) has accepted, accept; if \(M\) 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. \(\blacksquare\)
Any Questions?

HOMEWORK (due 10/30)
4.7, 4.8, 4.10, 4.13