Reducibility, part 2

April 15, 2020
Homework 7 is out (yay!),
Due by Friday April 24

- **Problem 1** (20 pts)
  - **Topic**: show that collection of decidable languages is closed under the operations concat., star, compl.
  - **Lecture(s)**: 3/25
  - **Book**: sec 3.1
- **Problem 2** (20 pts)
  - **Topic**: Hilbert’s tenth problem, decidability of the problem in the single-variable polynomial case
  - **Lecture(s)**: 3/30
  - **Book**: sec 3.3
- **Problem 3** (20 pts)
  - **Topic**: show that a given language on DFA (\(L(A) = \Sigma^*\)) is decidable
  - **Lecture(s)**: 4/1, 4/3
  - **Book**: sec 4.1

- **Problem 4** (20 pts)
  - **Topic**: uncountable sets, diagonalization
  - **Lecture(s)**: 4/6
  - **Book**: sec 4.2
- **Problem 5** (10 pts)
  - **Topic**: co-Turing-recognizable languages
  - **Lecture(s)**: 4/8
  - **Book**: sec 4.2
- **Problem 6** (10 pts)
  - **Topic**: Magic: The Gathering
  - **Lecture(s)**: 4/1, 4/3, 4/6, 4/8
  - **Book**: sec 4.1, 4.2
  - **+ real-life**
In lectures from last week and before (chapter 4), we saw...

Decidable:

$A_{DFA}$ (acceptance)
$A_{NFA}$
$A_{REX}$
$E_{DFA}$ (emptiness)
$E_{Q_{DFA}}$ (equivalence)
$A_{CFG}$
$E_{CFG}$
$E_{Q_{CFG}}$

Undecidable:

$A_{TM}$
This week (chapter 5) we are…

- Examining several additional (besides $A_{TM}$) unsolvable problems

- Learning about the primary method for proving that problems are computationally unsolvable – that method is called reducibility

- Examples we saw in last lecture (4/13)
  - Ex 1: Halting Problem
  - Ex 2: $E_{TM}$
  - Ex 3: $\text{REGULAR}_{TM}$ (began)
Example 3: \( \text{REGULAR}_{\text{TM}} \)

Let:

\[
\text{REGULAR}_{\text{TM}} = \{ <M> \mid M \text{ is a TM and } L(M) \text{ is a regular language} \}
\]

Theorem:

\( \text{REGULAR}_{\text{TM}} \) is undecidable
Proof idea

- We assume that $\text{REGULAR}_{\text{TM}}$ is decidable by a TM $R$ and use this assumption to construct a TM $S$ that decides $A_{\text{TM}}$.
- The idea is for $S$ to take its input $<M,w>$ and modify $M$ so that the resulting TM recognizes a regular language iff $M$ accepts $w$.
- We call the modified language $M_2$.
- We design $M_2$ to recognize the nonregular language $\{0^n1^n \mid n \geq 0\}$ if $M$ does not accept $w$, and to recognize the regular language $\Sigma^*$ if $M$ accepts $w$.
- We must specify how $S$ can construct such an $M_2$ from $M$ and $w$.
- Here, $M_2$ works by automatically accepting all strings in $\{0^n1^n \mid n \geq 0\}$.
- In addition, if $M$ accepts $w$, $M_2$ accepts all other strings.
Proof

• We let $R$ be a TM that decides $\text{REGULAR}_{\text{TM}}$, and Construct TM $S$ to decide $A_{\text{TM}}$.
• Then $S$ works in the following manner.

Let $S = \text{"On input } \langle M, w \rangle, \text{ where } M \text{ is a TM and } w \text{ is a string:} \text{\[}
\begin{enumerate}
\item Construct the following TM $M_2$. \[M_2 = \text{"On input } x:\]
\hspace{1cm} \begin{enumerate}
\item If $x$ has the form $0^n1^n$, accept.
\item If $x$ does not have this form, run $M$ on input $w$ and accept if $M$ accepts $w$.\]
\item Run $R$ on input $\langle M_2 \rangle$. \[\]
\item If $R$ accepts, accept; if $R$ rejects, reject.\[\]
\end{enumerate}
\end{enumerate}
Comment

- Using similar proof as in \( \text{REGULAR}_{\text{TM}} \), the problems of testing whether the language of a TM is one of the following is undecidable:
  - Context-free language
  - Decidable language
  - Finite language

- **Rice’s theorem** generalizes this
  - Determining *any property* of the languages recognized by Turing machines is undecidable
Example 4: $EQ_{TM}$

Let:

$EQ_{TM} = \{<M_1, M_2> \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$

Theorem:

$EQ_{TM}$ is undecidable
Proof idea

- Show that if $EQ_{TM}$ were decidable, $E_{TM}$ also would be decidable by giving a reduction from $E_{TM}$ to $EQ_{TM}$.

- The idea is simple. $E_{TM}$ is the problem of determining whether the language of a TM is empty. $EQ_{TM}$ is the problem of determining whether the languages of two TMs are the same.

- If one of these languages happens to be empty, we end up with the problem of determining whether the language of the other language is empty— that is, the $E_{TM}$ problem.

- So in a sense, the $E_{TM}$ problem is a special case of the $EQ_{TM}$ problem wherein one of the machines is fixed to recognize the empty language.
Proof

We let TM $R$ decide $EQ_{TM}$ and construct TM $S$ to decide $E_{TM}$ as follows

$$S = \text{“On input $\langle M \rangle$, where $M$ is a TM:}$$

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.”

If $R$ decides $EQ_{TM}$, $S$ decides $E_{TM}$. But $E_{TM}$ is undecidable, so $EQ_{TM}$ also must be undecidable.
Another look at this list (thanks to our growing body of knowledge)

Decidable:

\( A_{\text{DFA}} \) (acceptance)
\( A_{\text{NFA}} \)
\( A_{\text{REX}} \)
\( E_{\text{DFA}} \) (emptiness)
\( EQ_{\text{DFA}} \) (equivalence)
\( A_{\text{CFG}} \)
\( E_{\text{CFG}} \)
\( EQ_{\text{CFG}} \)

Undecidable:

\( A_{\text{TM}} \)
\( \text{HALT}_{\text{TM}} \)
\( E_{\text{TM}} \)
\( EQ_{\text{TM}} \)
\( \text{REGULAR}_{\text{TM}} \)
\( \text{CFL}_{\text{TM}} \)
\( \text{DECIDABLE}_{\text{TM}} \)

Rice’s theorem
Reductions via computation histories

- The computation history method is an important technique for proving that $A_{TM}$ is reducible to certain languages.

- Often useful when the problem to be shown undecidable involves testing for the existence of something.
  - E.g. undecidability of Hilbert’s tenth problem, testing for the existence of integral roots in a polynomial.

- We will define computation history today and use it to show a decidable problem; in next lecture we will use it to show examples of undecidable problems.
Definition – 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.
A *linear bounded automaton* is a restricted type of Turing machine wherein the tape head isn’t 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—in the same way that the head will not move off the left-hand end of an ordinary Turing machine’s tape.

An LBA is a TM with a limited amount of memory.
Let:

\[ A_{\text{LBA}} = \{ (M, w) | M \text{ is an lBA that accepts string } w \}. \]

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 \( n \).

Theorem:

\( A_{\text{LBA}} \) is decidable.
Proof – lemma on LBA

**Proof** Recall that a configuration of $M$ is like a snapshot in the middle of its computation. A configuration consists of the state of the control, position of the head, and contents of the tape. Here, $M$ has $q$ states. The length of its tape is $n$, so 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 quantities is the total number of different configurations of $M$ with a tape of length $n$. 
Proof – theorem on LBA

**PROOF** The algorithm that decides $A_{LBA}$ is as follows.

$L = "On \text{ 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, *accept* if it has accepted and *reject* if it has rejected. If it has not halted, *reject.*"

If $M$ on $w$ has not halted within $qng^n$ steps, it must be repeating a configuration according to Lemma 5.8 and therefore looping. That is why our algorithm rejects in this instance.