Computability Theory
More reductions

Haniel Barbosa

The University of Iowa
Readings for this lecture

Chapter 5 of [Sipser 1996], 3rd edition. Sections 5.1 and 5.3.
Linear Bounded Automata (LBA)

An LBA is a restricted type of TM 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.

▶ An LBA is a TM with a limited memory

▶ An LBA can only solve problems requiring memory that can fit within the segment of tape used as input.

▶ Using a tape alphabet larger than the input alphabet allows the available memory to be increased up to a constant factor.

▶ For an input of length $n$, the memory amount available is linear in $n$ (hence the name)
Linear Bounded Automata (LBA)

- Despite their memory constraint, LBAs are quite powerful.

- Deciders for $A_{DFA}$, $E_{DFA}$, $A_{CFG}$, $E_{CFG}$, all are LBAs.

- The majority of practical computing problems can be solved by an LBA.

- Every CFL for example can be decided by an LBA.

**Theorem**

$A_{LBA}$ is decidable.
Computation History

The computation history for a TM on an input in the sequence of configurations that the machine goes through as it processes the input.

Let $M$ be a TM and $w$ an input string. A computation history for $M$ on $w$ is a sequence of configurations $C_1, \ldots, C_k$ in which:

1. $C_1$ is the start configuration of $M$ on $w$
2. $C_{i+1}$ legally follows from $C_i$ according to the transition function of $M$

We distinguish two kind of histories:

- Accepting computation history: $C_k$ is an accepting configuration of $M$
- Rejecting computation history: $C_k$ is a rejecting configuration of $M$
Computation histories

- Computation histories are finite sequences.

- If $M$ does not halt on $w$, no accepting or rejecting computation history exists for $M$ on $w$.

- Deterministic machines have at most one computation history on a given input.

- Nondeterministic machines may have many computation histories on a single input, corresponding to various computation branches.
Reduction via computation histories

We show that $E_{LBA}$ is undecidable with a reduction using computation histories.

Under the assumption that $E_{LBA}$ is decidable:

- To decide $\langle M, w \rangle \in A_{TM}$ we construct an LBA $B$ and test if $L(B)$ is empty.
- The language recognized by $B$ consists of all accepting computation histories of $M$ on $w$.
- If $M$ accepts $w$ then $L(B) \neq \emptyset$, otherwise $L(B) = \emptyset$.
- $B$ operates by checking if inputs correspond to valid accepting computation histories of $M$ on $w$.
- Hardest to check is whether each intermediate configuration legally follows from the previous one.

If we can detect whether $L(B)$ is empty, we can determine whether $M$ accepts $w$. Therefore $E_{LBA}$ is undecidable.
Another example reduction

**Theorem**

\[ \text{ALL}_{\text{CFG}} = \{ \langle G \rangle \mid G \text{ is a CFG and } L(G) = \Sigma^* \} \text{ is undecidable.} \]

**Proof idea: reduction from \( A_{\text{TM}} \) to \( \text{ALL}_{\text{CFG}} \) via computation histories**

Assuming that \( \text{ALL}_{\text{CFG}} \) is decidable we can devise the following decision procedure for \( A_{\text{TM}} \):

\[
\triangleright \text{ For a TM } M \text{ and input } w \text{ construct CFG } G \text{ that generates all strings iff } M \text{ does not accept } w. \\
\triangleright \text{ If } M \text{ does accept } w \text{ then } G \text{ does not generate some particular string. This will correspond to the accepting computation history for } M \text{ on } w. \\
\]

This theorem the main result necessary for showing that the equivalence problem for CFGs is undecidable.
Strategy

- An accepting computation history for $M$ on $w$ has the form
  $\#C_1\#C_2\#\ldots\#C_k\#$.

- Therefore, $G$ generates all strings that
  1. do not start with $C_1$
  2. do not end with an accepting configuration
  3. for some $i$ and $C_i$, do not properly yield $C_{i+1}$ under the rules of $M$

- If $M$ does not accept $w$, no accepting history exists, so all strings fail in one way or another.
Strategy

- An accepting computation history for $M$ on $w$ has the form
  $\#C_1\#C_2\#\ldots\#C_k\#$.

- Therefore, $G$ generates all strings that
  1. do not start with $C_1$
  2. do not end with an accepting configuration
  3. for some $i$ and $C_i$, do not properly yield $C_{i+1}$ under the rules of $M$

- If $M$ does not accept $w$, no accepting history exists, so all strings fail in one way or another.

Since CFG and PDA are equivalent, we may use a PDA equivalent to $G$ to check the above conditions. It would operate on

$\#C_1\#C_2^R\#C_3\#C_4^R\#\ldots\#C_k\#$

to be able to check condition 3. (See textbook for construction.)
Proof

- Suppose that $TM \ R$ decides $ALL_{CFG}$. Construct $TM \ S$ that decides $A_{TM}$ as follows:

  $S = \text{"On input } \langle M, w \rangle \text{ in which a } M \text{ is a TM and } w \text{ a string:}
  \begin{enumerate}
    \item Construct CFG $G$ from $M$ and $w$ as described above.
    \item Run $R$ on input $\langle G \rangle$
    \item If $R$ rejects, accept; if $R$ accepts, reject"
  \end{enumerate}

- If $R$ accepts $\langle G \rangle$ then $L(\langle G \rangle) = \Sigma^*$, thus $M$ has no accepting computation on $w$, and $M$ does not accept $w$. Consequently $S$ rejects $\langle M, w \rangle$

- If $R$ rejects $\langle G \rangle$ then $L(\langle G \rangle) \neq \Sigma^*$. Since the only string that $G$ cannot generate is an accepting computation history for $W$ on $w$, it means that $M$ accepts $w$. Consequently $S$ accepts $\langle M, w \rangle$

This is a contradiction, so $R$ cannot exist, therefore $ALL_{CFG}$ is undecidable.
Testing equivalence of CFGs is undecidable

Theorem

\[ EQ_{\text{CFG}} = \{ \langle G, H \rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H) \} \] is undecidable.

Proof idea: reduction from \( ALL_{\text{CFG}} \) to \( EQ_{\text{CFG}} \)

A decider \( M \) for \( ALL_{\text{CFG}} \) can be built as follows:

\( M = \text{“On input } \langle G \rangle \text{ in which a } G \text{ is a CFG:} \)

1. Construct CFG \( H \) such that \( L(H) = \Sigma^* \)
2. Run the decider \( EQ_{\text{CFG}} \) on \( \langle G, H \rangle \)
3. If it accepts, accept; if it rejects, reject.”

Since this reduction leads to a contradiction, \( EQ_{\text{CFG}} \) is undecidable.
Mapping Reducibility

**Definition**

Language $A$ is *mapping reducible* to language $B$, written $A \leq_m B$, if there is a computable function $f : \Sigma^* \rightarrow \Sigma^*$, called a *reduction* from $A$ to $B$, such that for every $w \in \Sigma^*$,

$$w \in A \iff f(w) \in B$$

**Theorem**

*If $A \leq_m B$ and $B$ is decidable (Turing-recognizable), then $A$ is decidable (Turing-recognizable).*

**Corollary**

*If $A \leq_m B$ and $A$ is undecidable (not Turing-recognizable), then $B$ is decidable (not Turing-recognizable).*