Reductions via Computation Histories

- An accepting computation history for a TM is a sequence of configurations

$$C_1, C_2, \ldots, C_l$$

  such that
  - C_1 is the start configuration for input w
  - C_l is an accepting configuration, and
  - each C_i follows legally from the preceding configuration.
- A rejecting computation history is defined similarly.
- Computation histories are finite sequences – if M does not halt on w, there is no computation history.
- Deterministic v.s nondeterministic computation histories.

Linear Bounded Automaton

- Suppose we cripple a TM so that the head never moves outside the boundaries of the input string.
- Such a TM is called a linear bounded automaton (LBA)
- Despite their memory limitation, LBAs are quite powerful.

Lemma

Let M be a LBA with q states, g symbols in the tape alphabet. There are exactly qng^n distinct configurations for a tape of length n.

Proof.

- The machine can be in one of q states.
- The head can be on one of the n cells.
- At most g^n distinct strings can occur on the tape.

□

Decidability of LBA Problems

Theorem 5.9

$A_{LBA} = \{\langle M, w \rangle \mid M$ is an LBA that accepts string $w\}$ is decidable.

Proof Idea

- We simulate LBA M on w with a TM L (which is NOT an LBA!)
- If during simulation M accepts or rejects, we accept or reject accordingly.
- What happens if the LBA M loops?
  - Can we detect if it loops?
- M has a finite number of configurations.
  - If it repeats any configuration during simulation, it is in a loop.
  - If M is in a loop, we will know this after a finite number of steps.
  - So if the LBA M has not halted by then, it is looping.

Decidability of LBA Problems

Theorem 5.9

$A_{LBA} = \{\langle M, w \rangle \mid M$ is an LBA that accepts string $w\}$ is decidable.

Proof

- The following TM decides A_{LBA}.
- $L =$ "On input $\langle M, w \rangle$
  1. Simulate M on 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*."
- LBAs and TMs differ in one important way. A_{LBA} is decidable.

COMPUTATION OVER "COMPUTATION HISTORIES"

- Now for a really wild and crazy idea!
- Consider an accepting computation history of a TM M, $C_1, C_2, \ldots, C_l$
- Note that each C_i is a string.
- Consider the string

$$\# \underbrace{\qquad\qquad}_{C_1} \# \underbrace{\qquad\qquad}_{C_2} \# \underbrace{\qquad\qquad}_{C_3} \# \cdots \# \underbrace{\qquad\qquad}_{C_l} \#$$

- The set of all valid accepting histories is also a language!!
- This string has length m and an LBA B can check if this is a valid computation history for a TM M accepting w.
  - Check if $C_1 = q_0 w_1 w_2 \cdots w_n$
  - Check if $C_l = \cdots q_{accept} \cdots$
  - Check if each C_{i+1} follows from C_i legally.
- Note that B is not constructed for the purpose of running it on any input!
- If $L(B) \neq \Phi$ then M accepts w

DECIDABILITY OF LBA PROBLEMS

THEOREM 5.10

$E_{LBA} = \{\langle M \rangle \mid M$ is an LBA and $L(M) = \Phi\}$ is undecidable.

PROOF.

- Suppose TM R decides E_{LBA}, we can construct a TM S which decides A_{TM}
- $S =$ "On input $\langle M, w \rangle$, where M is a TM and w is a string
  1. Construct LBA B from M and w as described earlier.
  2. Run R on $\langle B \rangle$.
  3. If R rejects, *accept*; if R accepts, *reject*."
- So if R says $L(B) = \Phi$, the M does NOT accept w.
- If R says $L(B) \neq \Phi$, the M accepts w.
- But, A_{TM} is undecidable – contradiction.

□

Another example reduction

Theorem

$ALL_{\mathsf{CFG}} = \{\langle G\rangle \mid G$ is a CFG and $\mathscr{L}(G) = \Sigma^*\}$ is undecidable.

Proof idea: reduction from A_{TM} to ALL_{CFG} via computation histories

Assuming that ALL_{CFG} is decidable we can devise the following decision procedure for A_{TM}:

- ▷ For a TM M and input w construct CFG G that generates all strings iff M does not accept w.
- ▷ If M does accept w then G does *not* generate some particular string. This will correspond to the accepting computation history for M 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:
  $S =$"On input $\langle M, w\rangle$ in which a M is a TM and w a string:
  1. Construct CFG G from M and w as described above.
  2. Run R on input $\langle G\rangle$
  3. If R rejects, *accept*; if R accepts, *reject*"

- If R accepts $\langle G\rangle$ then $\mathscr{L}(\langle G\rangle) = \Sigma^*$, 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 $\mathscr{L}(\langle G\rangle) \neq \Sigma^*$. 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

$EQ_{\mathsf{CFG}} = \{\langle G,H\rangle \mid G$ *and* H *are CFGs and* $\mathscr{L}(G) = \mathscr{L}(H)\}$ *is undecidable.*

Proof idea: reduction from ALL_{CFG} to EQ_{CFG}

A decider M for ALL_{CFG} can be built as follows:

$M =$"On input $\langle G\rangle$ in which a G is a CFG:

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

Since this reduction leads to a contradiction, EQ_{CFG} is undecidable.