Reducibility

The Halting Problem

$HALT_{\mathrm{TM}} = \{\langle M, w\rangle \mid M$ is a TM and halts on input $w\}$.

Theorem. *$HALT_{\mathrm{TM}}$ is undecidable.*

Proof We could construct a machine S for A_{TM} from a machine R for $HALT_{\mathrm{TM}}$. On input x:

1. **Check that x is of form $\langle M, w\rangle$.** If not, reject x.
2. **Simulate R on $\langle M, w\rangle$.** Reject x if R has rejected.
3. M is *guaranteed to halt on w*.

   simulate M on w. Accept x if M has accepted and reject x otherwise. ∎

∎

The Emptiness Problem

$E_{\mathrm{TM}} = \{\langle M \rangle \mid M$ is a TM and $L(M) = \emptyset\}$.

Theorem. *E_{TM} is undecidable.*

Proof A TM R deciding $E_{\mathrm{TM}} \rightarrow$ a TM S deciding A_{TM}.
S's algorithm: On input x,

1. **Check that x is of form $\langle M, w \rangle$.** If not, reject x.
2. Construct a machine M_1 such that for each input y,
   M_1 simulates M on w and accepts y if M has accepted.
3. Check $L(M_1)$ for emptiness, i.e., **simulate R on input $\langle M_1 \rangle$.**
   Accept x if R has rejected and reject x otherwise. ∎

Testing Whether a TM Accepts a Regular Language

$REGULAR_{\mathrm{TM}} = \{\langle M \rangle \mid M$ is a TM and $L(M)$ is regular $\}$.

Theorem. *$REGULAR_{\mathrm{TM}}$ is undecidable.*

Proof A TM R deciding $REGULAR_{\mathrm{TM}} \rightarrow$ a TM S deciding A_{TM}.
S's algorithm: on input x,

1. **Check that x is of form $\langle M, w \rangle$.** If not, reject x.
2. Construct a machine M_1: Let a, b be two distinct symbols in the input alphabet Σ of M. On input y,
   (a) **Accept y if $y = a^n b^n$ for some n.**
   (b) Otherwise, simulate M on w, then **accept y if and only if M on w has accepted.**

Proof (cont'd)

The machine M_1 satisfies:

(*) $L(M_1) = \{a^n b^n \mid n \geq 1\}$ **if M doesn't accept w and Σ^* otherwise,** so

M accepts w if and only if $L(M_1)$ is regular.

3. Simulate R on $\langle M_1 \rangle$. **Accept x if R has accepted and reject x otherwise.** ∎

Testing Equivalence Between TMs

$EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle \mid M_1$ and M_2 are TMs and $L(M_1) = L(M_2)\}$.

Theorem. *EQ_{TM} is undecidable.*

Proof A TM R deciding $EQ_{\mathrm{TM}} \to$ a TM S deciding A_{TM}.

S's algorithm: on input x,

1. **Check that x is of form $\langle M, w\rangle$.** If not, reject x.
2. Construct a machine M_1 as in the previous proof.

   Construct a machine M_2 that accepts Σ^*.

   Then **$L(M_1) = L(M_2)$ if and only if M accepts w.**
3. **Simulate R on $\langle M_1, M_2\rangle$.** Accept x if R has accepted and reject x otherwise. ∎

Linear Bounded Automata

A **linear bounded automaton** is a Turing machine wherein the head is not permitted to move beyond the region in which the input was written. If the head attempts to move beyond the region it is kept at the same position.

Lemma. *Let M be an LBA with q states and with a tape alphabet of size s. For every $n \geq 1$, for every input of length n, there are* **precisely** qns^n **possible configurations***.*

The Acceptance Problem for LBA

$A_{\mathrm{LBA}} = \{\langle M, w\rangle \mid M$ is a TM and accepts w when restricted to be an LBA $\}$.

Theorem. *A_{LBA} is decidable.*

Proof Let M be a TM with q states and s symbols in the tape alphabet and w be an input to M of length n. By the previous lemma, there are only qns^n configurations, so if M on w accepts, it should do so within qns^n steps. So we have only to simulate M on w for qns^n steps. ∎

The Emptiness Problem for LBA

$E_{\text{LBA}} = \{\langle M \rangle \mid M$ is a TM and accepts no input viewed as an LBA $\}$.

Theorem. *E_{LBA} is undecidable.*

Proof A TM R deciding $E_{\text{LBA}} \longrightarrow$ a TM S deciding A_{TM}.
For a string $x = \langle M, w \rangle$ such that M is a Turing machine and w is an input to M, let L_x to be the set of all strings of the form $\#C_1\#C_2\# \cdots \#C_m\#$ such that

1. $C_1, \ldots, C_m$ **are configurations of** M**,**
2. C_1 **is the initial configuration** of M on w,
3. C_m **is an accepting configuration** of M on w, and
4. for every i, $1 \leq i \leq m-1$, C_{i+1} **is the next configuration of** C_i**.**

Proof (cont'd)

Then L_x can be decided by an LBA.

$L_x \neq \emptyset$ if and only M accepts w.

S's algorithm: on input x,

1. **Check that x is of form $\langle M, w \rangle$.** If not, reject x.
2. Construct a TM B that accepts L_x as an LBA.
3. Simulate R on $\langle B \rangle$. Reject x if R has accepted $\langle B \rangle$ and accept x otherwise. ∎

The Equivalence Problem for CFGs

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

Theorem. *ALL_{CFG} is undecidable.*

Proof For a string $x = \langle M, w \rangle$ such that M is a Turing machine and w is an input to M, let L_x be the set of all $\#D_1\# \cdots \#D_m\#$ for which there exist $C_1, \ldots, C_m$ such that:

1. **$C_1, \ldots, C_m$ are configurations of M,**
2. **C_1 is the initial configuration** of M on w,
3. **C_m is an accepting configuration** of M on w,
4. for every i, $2 \le i \le m$, **C_i is the next configuration of C_{i-1}**, and
5. for every i, $1 \le i \le m$, **$D_i = C_i$ if i is odd and $D_i = C_i^R$ otherwise.**

Proof (cont'd)

Then L_x is empty if and only if M does not accept w.

So, $\overline{L_x} = \Sigma^*$ if and only if M does not accept w.

$\overline{L_x}$ is a CFL. (Use a PDA that nondeterministically checks the **falsity** of 2, 3, or 4, accepting all prefixes & extensions.)

a TM R deciding $ALL_{\mathrm{CFG}} \rightarrow$ a TM S deciding A_{TM}
S's algorithm: on input x,

1. **Check that x is of form $\langle M, w\rangle$.** If not, reject x.
2. Construct a CFG G for $\overline{L_x}$. (Convert the above PDA to a CFG.)
3. **Simulate R on $\langle G\rangle$.** Accept x if R has rejected $\langle G\rangle$ and reject x otherwise. ∎

The Equivalence Problem (cont'd)

Define $EQ_{\mathrm{CFG}} = \{\langle G, H\rangle \mid G$ and H are CFGs that generate the same language $\}$.

Corollary. *EQ_{CFG} is undecidable.*

Proof A TM R deciding $EQ_{\mathrm{CFG}} \rightarrow$ a TM S deciding ALL_{CFG}.
S's algorithm: On input x,

1. **Check that x is of form $\langle G\rangle$.** If not, reject x.
2. Construct a grammar H for Σ^*.
3. **Simulate R on $\langle G, H\rangle$.** Accept x if R has accepted and reject x otherwise. ∎