Decidable Languages

$A_{\mathrm{DFA}} = \{\langle B, w\rangle |\ B \text{ is a DFA that accepts } w\ \}$

is decidable: on input $\langle B, w\rangle$, *simulate* B on w and accept iff B accepts.

A_{NFA}

A_{REX}

$E_{\mathrm{DFA}} = \{\langle A\rangle |\ A \text{ is a DFA and } L(A) = \emptyset\ \}$

is decidable: on input $\langle A\rangle$ accept iff A has an accept state that can be reached from the initial state.

$\mathrm{EQ}_{\mathrm{DFA}} = \{\langle A, B\rangle |\ A \text{ and } B \text{ are DFAs s.t. } L(A) = L(B)\ \}$

is decidable: $L(C) = (L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B))$, and test if $L(C) = \emptyset$.

$A_{\text{CFG}} = \{\langle G, w\rangle \mid G$ is a CFG that generates w $\}$

is decidable: Chomsky Normal Form derivations have length exactly $2n - 1$.

$E_{\text{CFG}} = \{\langle G\rangle \mid G$ is a CFG and $L(G) = \emptyset$ $\}$

is decidable: (1) mark all terminal symbols, (2) repeat: mark A where $A \longrightarrow \alpha$, and every symbol in α is marked, (3) accept iff the start symbol is marked.

$\text{EQ}_{\text{CFG}} = \{\langle G, H\rangle \mid G, H$ are CFGs & $L(G) = L(H)$ $\}$

is **not** decidable.

Theorem: Every CFL is decidable.

Proof: Chomsky Normal Form.

The Halting Problem

$A_{\mathrm{TM}} = \{\langle M, w\rangle |\ M \text{ is a TM and } M \text{ accepts } w\ \}$

Theorem: A_{TM} is undecidable, but it is Turing-recognizable.

Proof: Turing-recognizable: simulation.

Suppose it is decidable; then $\exists$ TM H that decides it.

Define a new TM D ("D" for Diagonal) as follows:

$$D(\langle M\rangle) = \begin{cases} \text{accept} & \text{if } H(\langle M, \langle M\rangle\rangle) = \text{reject} \\ \text{reject} & \text{if } H(\langle M, \langle M\rangle\rangle) = \text{accept} \end{cases}$$

Consider $D(\langle D\rangle)$. It turns out that D accepts $\langle D\rangle$ iff D does not accept $\langle D\rangle$ — contradiction.

A language is **co-Turing-recognizable** if it is the complement of a Turing-recognizable language.

Theorem: A language is decidable iff it is both Turing and co-Turing recognizable.

Proof: $[\Longrightarrow]$ is easy. For the other direction, run in parallel the TMs that recognize a language and its complement.

Corollary: $\overline{A_{\mathrm{TM}}}$ is not Turing-recognizable.

Reducibility

Theorem:
$\mathrm{HALT}_{\mathrm{TM}} = \{\langle M, w\rangle |\ M$ is a TM and M halts on input w $\}$ is undecidable.

Proof: Suppose it is, and R is its decider. We use R to construct a decider S for A_{TM}, and hence get a contradiction.

On input $\langle M, w\rangle$, S simulates R on the same input. If R accepts, then S simulates M on w, and accepts iff M accepts. If R rejects, so does S.

We reduced A_{TM} to $\mathrm{HALT}_{\mathrm{TM}}$, thereby showing that $\mathrm{HALT}_{\mathrm{TM}}$ is not decidable, because if it were, so would be A_{TM}—which we know isn't.

Theorem: $\text{REGULAR}_{\text{TM}} =$
$\{\langle M\rangle \mid M$ is a TM and $L(M)$ is a regular language $\}$ is undecidable.

Proof: Suppose it has a decider R.

We build S based on R to decide A_{TM}.

On input $\langle M, w\rangle$, S constructs $\langle M_2\rangle$ where M_2 works as follows:

> on input x, if $x = 0^n 1^n$ for some n, accept; otherwise run M on input w and accept iff M accepts w.

Finally, run R on $\langle M_2\rangle$.

If R accepts, accept; if R rejects, reject.

An **accepting computation history** for M on w is a sequence of configurations $C_1, C_2, \ldots, C_l$, where $C_1 = q_0 w$, $C_i \vdash C_{i+1}$, and C_l is accepting.

A **Linear Bounded Automaton (LBA)** is a TM wherein the head cannot move off the input.

LBAs decide: $A_{\mathrm{DFA}}, A_{\mathrm{CFG}}, E_{\mathrm{DFA}}, E_{\mathrm{CFG}}$.

Theorem: Every CFL can be decided by an LBA.

Proof: Given w, any derivation of w has at most $|w|$ many symbols in an intermediate step. Try out all possible derivations, keeping track of them by having a single square represent a constant number of different symbols.

Theorem: $A_{\mathrm{LBA}} = \{\langle M, w\rangle |\ M$ is an LBA that accepts string w $\}$ is decidable.

Proof: First observation:

> There are $n \cdot |Q| \cdot |\Gamma|^n$ distinct configurations of an LBA on inputs of size n.

On input $\langle M, w\rangle$ simulate M on w for at most $|w| \cdot |Q| \cdot |\Gamma|^{|w|}$ many steps.

If within these many steps, it halted, accept iff it accepted.

Otherwise, it must have entered an infinite loop, so reject if it has not halted and the number of steps exceeded the above quantity.

Theorem: $E_{\mathrm{LBA}} = \{\langle M\rangle \mid M$ is an LBA where $L(M) = \emptyset\ \}$ is undecidable.

Proof: Suppose it has a decider.

We build a decider S for A_{TM} as follows:

on input $\langle M, w\rangle$, S constructs $\langle B_{\langle M,w\rangle}\rangle$ where $B_{\langle M,w\rangle}$ is an LBA that accepts x iff x is an accepting computation history of M on w, i.e.,

$$x = C_1 \ \texttt{\#}\ C_2 \ \texttt{\#}\ \ldots \ \texttt{\#}\ C_l$$

Then, S uses the decider for E_{LBA} to test if $B_{\langle M,w\rangle}$ is empty.

If $L(B_{\langle M,w\rangle}) = \emptyset$ reject, otherwise accept.

Post's Correspondence Problem (PCP)

An instance of PCP consists of two finite lists of strings over some alphabet Σ. The two lists must be of equal length:

$A = w_1, w_2, \ldots, w_k$
$B = x_1, x_2, \ldots, x_k$

For each i, the pair (w_i, x_i) is said to be a *corresponding pair*. We say that this instance of PCP has a solution if there is a sequence of one or more indices:

$$i_1, i_2, \ldots, i_m \qquad m \geq 1$$

such that:

$$w_{i_1} w_{i_2} \ldots w_{i_m} = x_{i_1} x_{i_2} \ldots x_{i_m}$$

The PCP is: given (A, B), tell whether there is a solution.

Aside: To express PCP as a language, we let L_{PCP} be the language: $\{\langle A, B\rangle | (A, B)$ instance of PCP with solution$\}$

Ex. $A = 1, 10111, 10$; $B = 111, 10, 0$

Then $2, 1, 1, 3$ is a solution as:

$$\underbrace{10111}_{w_2}\underbrace{1}_{w_1}\underbrace{1}_{w_1}\underbrace{10}_{w_3} = \underbrace{10}_{x_2}\underbrace{111}_{x_1}\underbrace{111}_{x_1}\underbrace{0}_{x_3}$$

Note that $2, 1, 1, 3, 2, 1, 1, 3$ is another solution.

On the other hand:

$A = 10, 011, 101$; $B = 101, 11, 011$

Does not have a solution.

Modified PCP (MPCP)

The MPCP has an additional requirement that the first pair in the solution must be the first pair of (A, B).

So $i_1, i_2, \ldots, i_m$, $m \geq 0$, is a solution to the (A, B) instance of MPCP if:

$$w_1 w_{i_1} w_{i_2} \ldots w_{i_m} = x_1 x_{i_1} x_{i_2} \ldots x_{i_m}$$

$i_1, i_2, \ldots, i_r$ is a *partial solution* of PCP if one of the following is the prefix of the other:

$$w_{i_1} w_{i_2} \ldots w_{i_r} \qquad x_{i_1} x_{i_2} \ldots x_{i_r}$$

Same def holds for MPCP, but w_1, x_1 must be at the beginning.

Show: (1) If PCP is decidable, then so is MPCP, (2) if MPCP is decidable, then so is A_{TM}, Since A_{TM} is *not* decidable, neither is (M)PCP.

$\boxed{\text{PCP decidable} \Longrightarrow \text{MPCP decidable}}$

We show that given an instance (A, B) of MPCP, we can construct an instance (A', B') of PCP such that:

$$(A, B) \text{ has solution } \iff (A', B') \text{ has solution}$$

Let (A, B) be an instance of MPCP over the alphabet Σ. Then (A', B') is an instance of PCP over the alphabet $\Sigma' = \Sigma \cup \{*, \$\}$.

If $A = w_1, w_2, w_3, \ldots, w_k$, then
$A' = *\mathbf{w}_1*, \mathbf{w}_1*, \mathbf{w}_2*, \mathbf{w}_3*, \ldots, \mathbf{w}_k*, \$$.

If $B = x_1, x_2, x_3, \ldots, x_k$, then $B' = *\mathbf{x}_1, *\mathbf{x}_1, *\mathbf{x}_2, *\mathbf{x}_3, \ldots, *\mathbf{x}_k, *\$$.

where if $x = a_1 a_2 a_3 \ldots a_n \in \Sigma^*$, then $\mathbf{x} = a_1 * a_2 * a_3 * \ldots * a_n$.

Ex. If (A, B) is an instance if MPCP given as:

$A = 1, 10111, 10$
$B = 111, 10, 0$

Then (A', B') is an instance of PCP given as follows:

$A' = *1*, 1*, 1*0*1*1*1*, 1*0*, \$$
$B' = *1*1*1, *1*1*1, *1*0, *0, *\$$

$\boxed{\text{MPCP decidable} \Longrightarrow A_{\mathrm{TM}} \text{ decidable}}$

Given a pair (M, w) we construct an instance (A, B) of MPCP such that:

TM M accepts $w \iff (A, B)$ has a solution.

The MPCP instance (A, B) simulates, in its partial solutions, the computation of M on w.

That is, partial solutions will be of the form:

$$\#\alpha_1\#\alpha_2\#\alpha_3\#\ldots$$

where α_1 is the initial configuration of M on w, and for all i, $\alpha_i \vdash \alpha_{i+1}$.

The string from the B list will always be one configuration ahead of the A list; the A list will be allowed to "catch-up" only when M accepts w.

We construct an instance (A, B) of MPCP as follows:

1. A: $\#$
   B: $\#q_0w\#$

2. A: $X_1, X_2, \ldots, X_n, \#$
   B: $X_1, X_2, \ldots, X_n, \#$
   where the X_i are all the tape symbols.

3. To simulate a move of M, for all non-accepting $q \in Q$:

list A	list B	
qX	Yp	if $\delta(q, X) = (p, Y, \rightarrow)$
ZqX	pZY	if $\delta(q, X) = (p, Y, \leftarrow)$
$q\#$	$Yp\#$	if $\delta(q, B) = (p, Y, \rightarrow)$
$Zq\#$	$pZY\#$	if $\delta(q, B) = (p, Y, \leftarrow)$

4. If the configuration at the end of B has an accepting state, then we need to allow A to catch up with B. So we need for all accepting states q, and all symbols X, Y:

list A	list B
XqY	q
Xq	q
qY	q

5. Finally, after using 4 and 3 above, we end up with $x\#$ and $x\#q\#$, where x is a long string. Thus we need $q\#\#$ in A and $\#$ in B to complete the catching up.

$f : \Sigma^* \longrightarrow \Sigma^*$ is a **computable function** if some TM M computes $f(x)$ on input x; i.e., there exists a TM M such that $f(x) = M(x)$, and M *always* halts.

A is **mapping reducible (many-one reducible)** to B, $A \leq_m B$, if there is an $f : \Sigma^* \longrightarrow \Sigma^*$ s.t. $w \in A \iff f(w) \in B$.

f is called the **reduction** of A to B.

Theorem: $A \leq_m B$, and B is decidable (Turing-recognizable), then so is A.

Corollary: If $A \leq_m B$ and A is undecidable (not Turing-recognizable), then so is B.