Computation Histories

This is an approach to show that some languages are undecidable. For instance, the undecidability of the Hilbert's tenth problem uses this approach.

Computation History of a TM on an input is simply the sequence of configurations that the TM goes through while it processes the input.

It is represented as $\#C_1\#C_2\#\ldots\#C_l\#$ where $C_1, C_2, \ldots C_l$ are configurations.

Def 5.5: Let M be a TM and w be an input string. An accepting computation history of M on w is a sequence of configurations $C_1, C_2, \ldots C_l$ such that C_1 is the starting configuration of M on w, C_l is an accepting configuration, and each C_i legally follows from C_{i-1}, according to the rules of M.

A rejecting CH is defined the same way, but C_l is a rejecting configuration.

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

Theorem 5.13: ALL_{CFG} is undecidable.

Proof Sketch: We reduce $A_{TM} \leq_m \overline{ALL_{CFG}}$ using computation histories.

Given M and w, we will construct a CFG/PDA that generates all the strings that are not an accepting CH of M on w.

M accepts w $\Rightarrow$ CFG generates all but one string.

M does not accept w $\Rightarrow$ CFG generates all strings.

$ACH_{M,w} = \{z \in \Delta^* \mid z \text{ is an accepting CH of M on w}\}$

$\langle M, w \rangle \in A_{TM} \iff$ M accepts w

$\iff ACH_{M,w} \neq \emptyset$

$\iff \overline{ACH_{M,w}} \neq \Delta^*$

The grammar that generates $\overline{ACH_{M,w}}$ $\iff \overline{ACH_{M,w}}$ & ALL_{CFG}.

$\iff \overline{ACH_{M,w}} \in \overline{ALL_{CFG}}$

To complete the reduction $A_{TM} \leq_m \overline{ALL_{CFG}}$, we need to show that there is a CFG that generates $\overline{ACH_{M,w}}$.

Given $z \in \Delta^*$, there are four possibilities on why $z \in \overline{ACH_{M,w}}$ (or why z is not an accepting CH of M on w). The CFG can nondeterministically choose one of the following.

1. z is not well formed. Maybe does not start or end with #. Either no state or more than one state between two #'s.

   — This is a regular language and can be checked using a DFA.

2. z does not start correctly. The first config. is not the starting config. of M on w, i.e.,

$q_s w_1 w_2 \ldots w_n$.

– Regular. Can be checked using a DFA.

3. z does not end correctly. The state between the last two # symbols is not the accepting state.

– Regular. Can be checked using a DFA.

4. There is an i such that C_{i+1} is not a valid successor of C_i, i.e., $C_i \# C_{i+1}$ is not valid.

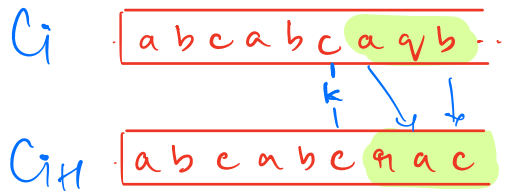
– Non deterministically guess i such that C_{i+1} is not a successor of C_i.

– Use a PDA to check that $C_i \# C_{i+1}$ is not valid.

Try problem 2.22. Show that C is context-free.

$$C = \{ x \# y \mid x, y \in \{0,1\}^* \text{ and } x \neq y \}.$$

Main ideas for checking that $C_i \# C_{i+1}$ is not valid.



$\delta(q,b) = (r,c,L)$

→ We have to nondeterministically choose the position where C_{i+1} makes an "error", say k

→ Use stack to keep track of k.

→ Need to remember the previous two symbols in case the state (tape head) is nearby.

Post Correspondence Problem (PCP)

This is a simple undecidable problem.

Given dominos $\left\{ \left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \cdots \left[\frac{t_k}{b_k}\right] \right\}$

can we arrange them (repeats allowed) so that top string = bottom string?

Example: $\left\{ \left[\frac{ab}{aba}\right], \left[\frac{ba}{abb}\right], \left[\frac{b}{ab}\right], \left[\frac{abb}{b}\right], \left[\frac{a}{bab}\right] \right\}$

$\left[\frac{ab}{aba}\right], \left[\frac{a}{bab}\right], \left[\frac{ba}{abb}\right], \left[\frac{b}{ab}\right], \left[\frac{abb}{b}\right], \left[\frac{abb}{b}\right], \left[\frac{b}{ab}\right], \left[\frac{abb}{b}\right]$

↑

This is a match.

PCP is an undecidable problem. More about it next week.