PCP and Mapping Reducibility

Post Correspondence Problem (PCP)

Given a finite collection of dominos, each containing a string on each half, decide whether the dominos can be placed (with repetition) in line so that **the upper halves and the lower halves read the same from left to right.** Such a placement of dominos is called a **match.**

Example: Given a collection

$$\left\{ \left[\frac{\mathsf{b}}{\mathsf{ca}}\right], \left[\frac{\mathsf{a}}{\mathsf{ab}}\right], \left[\frac{\mathsf{ca}}{\mathsf{a}}\right], \left[\frac{\mathsf{abc}}{\mathsf{c}}\right] \right\}$$

the list

$$\left\{ \left[\frac{\mathsf{a}}{\mathsf{ab}}\right] \left[\frac{\mathsf{b}}{\mathsf{ca}}\right] \left[\frac{\mathsf{ca}}{\mathsf{a}}\right] \left[\frac{\mathsf{a}}{\mathsf{ab}}\right] \left[\frac{\mathsf{abc}}{\mathsf{c}}\right] \right\}$$

yields the string $\mathsf{abcaaabc}$ on both halves.

PCP is undecidable

$PCP = \{\langle P \rangle \mid P$ **is an instance of the Post correspondence problem with a match** $\}$**.**

We deal with a modified version of the problem

$MPCP = \{\langle P \rangle \mid P$ **is an instance of the Post correspondence problem with a match starting with the first domino** $\}$**.**

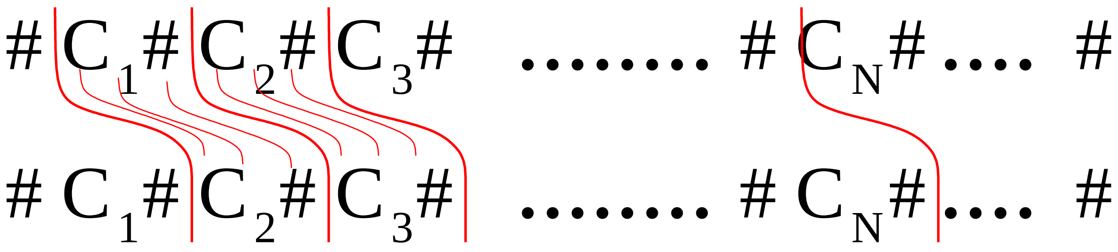
Then we transform A_{TM} to $MPCP$ in such a way that, for each $x = \langle M, w \rangle$:

(*) the matched string generated by the dominos for x will encode an accepting computation of M on w (with a "trailer").

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$.

We’ll need dominos $\left[\frac{\#}{\#}\right]$ and $\left[\frac{\mathsf{a}}{\mathsf{a}}\right]$ for each a in Γ.

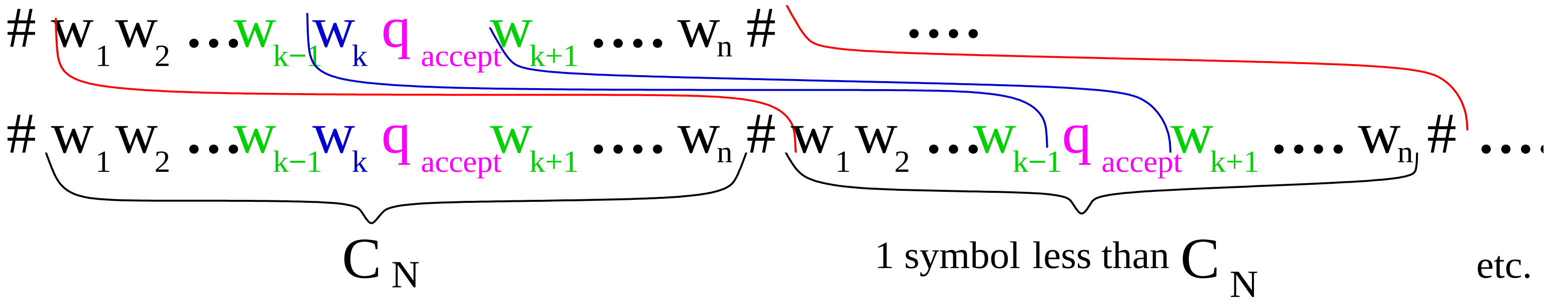
Problem: How do we constrain the (doubled up) sequence of configurations to be an accepting history?



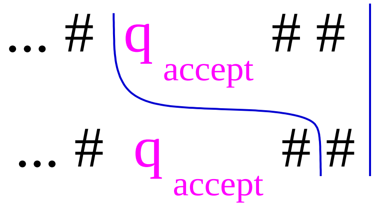
Solution: Use a “shifted” correspondence (red lines). (Simple vertical correspondence $\left[\frac{\#}{\#}\right]$, $\left[\frac{\mathsf{a}}{\mathsf{a}}\right]$, ..., would give *all* strings.) Also add dominos that make the right changes in the vicinity of the state (head position) symbol (details later).

Further problem: We'll end up with one "extra" configuration on the bottom, the accepting configuration.

Solution: Use dominos that delete a symbol adjacent to q_{accept}, and finally one that adds q_{accept}## to the top and # to the bottom:



Final portion:



Question: Why is there a match *only if* there is an accepting computation?

Answer: Because only the initial and final domino have an unequal number of #'s in the top and bottom. So by "forcing" the initial inequality, we force the correspondence to remain shifted by one configuration till the end. And the shifted correspondences must comply with the transition function, by our choice of dominos.

Details: Three Kinds of Dominos

1. The Initial Domino: $[\frac{\#}{\# q_0 x_1 \cdots x_n \#}]$.

   The lower part is one computational step ahead of the upper part.

2. The Computation Dominos:

   Correspond to configuration rewriting rules. **Filling the upper part that is lagging behind with the computational dominos will advance the lower part by a single step of M.**

3. The Cleaning Dominos:

   Dominos that gradually "eat up" the lower part while keeping the state in q_{accept}.

The Computation Dominos

- $[\frac{\#}{\#}]$, and $[\frac{a}{a}]$ for each $a \in \Gamma$. (This includes $[\frac{\sqcup}{\sqcup}]$.)

- For each $p, q \in Q$ and $a, b, c \in \Gamma$ such that $\delta(p, a) = (q, b, L)$, $[\frac{\#pa}{\#qb}]$ and $[\frac{cpa}{qcb}]$.

- For each $p, q \in Q$ and $a, b, c \in \Gamma$ such that $\delta(p, a) = (q, b, R)$, $[\frac{pa\#}{bq\sqcup\#}]$ and $[\frac{pac}{bqc}]$.

Note: Sipser uses $[\frac{pa}{bq}]$ and $[\frac{\#}{\sqcup\#}]$ instead of the 2 types above.

The Cleaning Dominos

- For each $a \in \Sigma$, $[\frac{a q_{\text{accept}}}{q_{\text{accept}}}]$ and $[\frac{q_{\text{accept}} a}{q_{\text{accept}}}]$.

- The end domino: $[\frac{q_{\text{accept}} \# \#}{\#}]$.

From MPCP to PCP

For a string $u = u_1 u_2 \cdots u_m$, let $\star u = \star u_1 \star u_2 \star \cdots \star u_m$, $u\star = u_1 \star u_2 \star \cdots \star u_m \star$, $\star u \star = \star u_1 \star u_2 \star \cdots \star u_m \star$, where $\star$ is a new symbol.

Modify the start domino $[\frac{t}{b}]$ to $[\frac{\star t}{\star b \star}]$ and each other domino $[\frac{\mathsf{u}}{\mathsf{v}}]$ to $[\frac{\star u}{v \star}]$; add a new domino $[\frac{\star \diamond}{\diamond}]$, where $\diamond$ is a new symbol.

This will force the start domino to be the first one and the newly added one to be the last one.

Computable Functions

A function $f : \Sigma^* \to \Sigma^*$ is **computable** if there exists a Turing machine M such that for every $x \in \Sigma^*$, M on x halts with just $f(x)$ on its tape.

Example: Let Σ be a fixed alphabet. Define $f : \Sigma^* \to \Sigma^*$ as follows:

- If $w = \langle M \rangle$ for some Turing machine, then $f(w) = \langle M' \rangle$ where M' is M with q_{accept} and q_{reject} swapped.
- Otherwise, $f(w) = w$.

Then f is computable.

Mapping Reducibility

A language $A \subseteq \Sigma^*$ is **mapping reducible** to $B \subseteq \Sigma^*$ (write $A \leq_{\mathrm{m}} B$) if there exists a computable function $f : \Sigma^* \to \Sigma^*$ such that for every $x \in \Sigma^*$,

$$x \in A \textbf{ if and only if } f(x) \in B.$$

In other words, the function f maps **members of A to members of B** and **nonmembers of A to nonmembers of B**.

Example: $A_{TM} \leq_{\mathrm{m}} HALT_{TM}$. We can use $f(x) = x$ if x is not of form $\langle M, w \rangle$. Otherwise, $f(\langle M, w \rangle) = \langle M_1, w \rangle$, where M_1, for input y, simulates M on y, and accepts y if M accepts y, and otherwise runs forever.

Then $x \in A_{TM}$ (which implies x is of form $\langle M, w \rangle$ and M accepts w) iff $f(x) \in$ *HALT*$_{TM}$.

Properties of Mapping Reducibility

Theorem. *If $A \leq_{\mathrm{m}} B$ and B is decidable then A is decidable.*

Proof Let $A \leq_{\mathrm{m}} B$ be witnessed by a Turing machine M_f that computes a mapping reduction f from A to B.

Suppose B is decided by a Turing machine M_B. Construct a new Turing machine M_A:

1. On input x, simulate M_f on x to compute $f(x)$.
2. **Simulate M_B on $f(x)$.** Accept if M_B accepts and reject if M_B rejects.

Then M_A decides A. ∎

Properties of Mapping Reducibility (cont'd)

Corollary. *If $A \leq_{\mathrm{m}} B$ and A is undecidable then B is undecidable.* ∎

Theorem. *If $A \leq_{\mathrm{m}} B$ and B is Turing-recognizable then A is Turing-recognizable.* ∎

Proof Same as for decidable case, but we conclude "Then M_A recognizes A".

Corollary. *If $A \leq_{\mathrm{m}} B$ and A is not Turing-recognizable then B is not Turing-recognizable.* ∎

EQ_{TM} Goes Beyond the Turing-Recognizable Languages

Theorem. *EQ_{TM} is neither Turing-recognizable nor co-Turing-recognizable.*

Proof Show that A_{TM} is mapping reducible to EQ_{TM} as well as to $\overline{EQ_{\mathrm{TM}}}$. Let $s \in EQ_{\mathrm{TM}}$ and $t \in \overline{EQ_{\mathrm{TM}}}$ be fixed.

Reduction to EQ_{TM}

- If x is of the form $\langle M, w \rangle$, then $f(x) = \langle M_1, M_2 \rangle$, where
  - M_1 accepts every input y; and
  - M_2 first simulates M on w and accepts *its own input* y if M has accepted w.
- Otherwise, $f(x) = t$.

f is computable, and for every x, $x \in A_{\mathrm{TM}}$ if and only if $f(x) \in EQ_{\mathrm{TM}}$.

Proof (cont'd)

Reduction to $\overline{EQ_{\mathrm{TM}}}$

- If x is of the form $\langle M, w \rangle$, then $g(x) = \langle M_1, M_2 \rangle$, where
  - M_1 rejects every input y; and
  - M_2 first simulates M on w and accepts *its own input* y if M has accepted w.
- Otherwise, $g(x) = s$.

g is computable and for every x, $x \in A_{\mathrm{TM}}$ if and only if $g(x) \notin EQ_{\mathrm{TM}}$.

Thus, $A_{\mathrm{TM}} \leq_{\mathrm{m}} EQ_{\mathrm{TM}}$ and $A_{\mathrm{TM}} \leq_{\mathrm{m}} \overline{EQ_{\mathrm{TM}}}$. ∎