PCP and Mapping Reducibility

Post Correspondence Problem (PCP)

We have a collection of domino pieces, each of which has a string in the top half and a string in the bottom half. Suppose we have infinitely many supplies of each piece. Can we produce a sequence of these domino pieces so that the string that emerges in the top half is identical to that in the bottom half?

We call such a placement of domino pieces a **match**.

Example: Given a collection

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

the list

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

yields the string abcaaabc in both halves.

PCP is undecidable

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

Our goal is to show that PCP is undecidable.

MPCP

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 domino pieces for** x **will encode accepting computation of** M **on** w**.**

Three Kinds of Domino Pieces

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ be the machine of our interest. We will use three types of domino pieces: the initial domino, the computation domino pieces, and the clearing domino pieces.

Three Kinds of Domino Pieces

The idea behind matching is the following:

1. The **initial domino creates the initial configuration** of M on both sides, with an overhang on the bottom side.

Three Kinds of Domino Pieces

The idea behind matching is the following:

1. The **initial domino creates the initial configuration** of M on both sides, with an overhang on the bottom side.
2. The computation pieces **extend the domino sequence and append the next configuration**, while maintaining the existence of an overhang on the bottom side.

Three Kinds of Domino Pieces

The idea behind matching is the following:

1. The **initial domino creates the initial configuration** of M on both sides, with an overhang on the bottom side.
2. The computation pieces **extend the domino sequence and append the next configuration**, while maintaining the existence of an overhang on the bottom side.
3. Once the configuration becomes an accepting one, the clearing pieces **enable to the top part to catch up with the bottom part.**

The Initial Domino

$$[\frac{\#}{\#q_0x_1 \cdots x_n\#}]$$

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

The Computation Domino Pieces

What we want to do is to produce from

$$[\frac{\#C_1\#C_2\#\cdots\#C_{k-1}\#}{\#C_1\#C_2\#\cdots\#C_{k-1}\#C_k\#}]$$

such that $C_1, C_2, \cdots, C_k$ are configurations of M and C_k is not an accepting configuration,

$$[\frac{\#C_1\#C_2\#\cdots\#C_{k-1}\#C_k\#}{\#C_1\#C_2\#\cdots\#C_{k-1}\#C_k\#C_{k+1}\#}]$$

where C_{k+1} is the next configuration of C_k.

The Computation Domino Pieces

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

Use of Computation Pieces

Suppose $C_k = ababpcd$ and $\delta(p, c) = (q, e, R)$. Then the following extension occurs:

$$\left[\frac{\cdots\#}{\cdots\#ababpcd\#}\right] \Longrightarrow \left[\frac{\cdots\#a}{\cdots\#ababpcd\#a}\right] \Longrightarrow$$

$$\left[\frac{\cdots\#ab}{\cdots\#ababpcd\#ab}\right] \Longrightarrow \left[\frac{\cdots\#aba}{\cdots\#ababpcd\#aba}\right] \Longrightarrow$$

$$\left[\frac{\cdots\#abab}{\cdots\#ababpcd\#abab}\right] \Longrightarrow \left[\frac{\cdots\#ababpc}{\cdots\#ababpcd\#ababeq}\right] \Longrightarrow$$

$$\left[\frac{\cdots\#ababpcd}{\cdots\#ababpcd\#ababeqd}\right] \Longrightarrow \left[\frac{\cdots\#ababpcd\#}{\cdots\#ababpcd\#ababeqd\#}\right]$$

$C_{k+1} = ababeqd$ is the next configuration.

The Cleaning Domino Pieces

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

These domino pieces are used to shorten the overhang configuration.

Use of Cleaning Pieces

Suppose the current overhang is $abq_{\text{accept}}cde\#$. We have:

$$\left[\frac{\cdots\#}{\cdots\#abq_{\text{accept}}cde\#}\right] \Longrightarrow \left[\frac{\cdots\#a}{\cdots\#abq_{\text{accept}}cde\#a}\right] \Longrightarrow$$

$$\left[\frac{\cdots\#abq_{\text{accept}}}{\cdots\#abq_{\text{accept}}cde\#aq_{\text{accept}}}\right] \Longrightarrow \left[\frac{\cdots\#abq_{\text{accept}}c}{\cdots\#abq_{\text{accept}}cde\#aq_{\text{accept}}c}\right]$$

$$\Longrightarrow \left[\frac{\cdots\#abq_{\text{accept}}cd}{\cdots\#abq_{\text{accept}}cde\#aq_{\text{accept}}cd}\right]$$

$$\Longrightarrow \left[\frac{\cdots\#abq_{\text{accept}}cde}{\cdots\#abq_{\text{accept}}cde\#aq_{\text{accept}}cde}\right] \Longrightarrow$$

$$\left[\frac{\cdots\#abq_{\text{accept}}cde\#}{\cdots\#abq_{\text{accept}}cde\#aq_{\text{accept}}cde\#}\right]$$

The bottom overhang has lost the b!

From MPCP to PCP

Let $\star$ be a new symbol. For a string $u = u_1 u_2 \cdots u_m$ not containing a $\star$, define

- $\star u = \star u_1 \star u_2 \star \cdots \star u_m$,
- $u\star = u_1 \star u_2 \star \cdots \star u_m \star$, and
- $\star u \star = \star u_1 \star u_2 \star \cdots \star u_m \star$,

String Modification

- Change the start domino $[\frac{t}{b}]$ to $[\frac{\star t}{\star b \star}]$.
- Change each of the remaining domino pieces uv to $[\frac{\star u}{v \star}]$.
- Add a new "last" domino $[\frac{\star \diamond}{\diamond}]$, where $\diamond$ is a yet another new symbol.

This will force the start domino to be the first one and the "last" domino 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 $x \in \Sigma^*$,

$x \in A$ **if and only if** $f(x) \in B$**.**

Namely, the function f maps **members of** A **to members of** B and **non-members of** A **to non-members of** B.

Properties About 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 R that computes a mapping reduction f from A to B.

Suppose B is decided by a Turing machine M. Construct a new Turing machine N:

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

Then N 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.** ∎

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

$EQTM$ Goes Beyond the Turing-Recognizable Languages

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

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 $\overline{EQ_{\text{TM}}}$. Let $s \in EQ_{\text{TM}}$ and $t \in \overline{EQ_{\text{TM}}}$ 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; and
  - M_2 first simulates M on w and accepts *its own input* if M accepts.
- Otherwise, $f(x) = t$.

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

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; and
  - M_2 first simulates M on w and accepts *its own input* if M accepts.
- Otherwise, $f(x) = s$.

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

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