The Post Correspondence Problem

Mark Greenstreet, CpSc 421, Term 1, 2008/09

- The Post Correspondence Problem (PCP)
  - Definition
  - Examples
  - Demonstrating undecidability of PCP

- Reductions summary
The Post Correspondence Problem

- Given a set, \( P \) of pairs of strings:

\[
P = \left\{ \left[ \frac{t_1}{b_1} \right], \left[ \frac{t_2}{b_2} \right], \ldots, \left[ \frac{t_k}{b_k} \right] \right\}
\]

where each \( t_i, b_i \in \Sigma^* \),

- Question: Does there exist a sequence \( i_1, i_2, \ldots, i_n \) such that:

\[
 t_{i_1} t_{i_2} \cdots t_{i_n} = b_{i_1} b_{i_2} \cdots b_{i_n}
\]

Note: the same pair can occur multiple times, i.e. there can be \( j \neq m \) s.t. \( i_j = i_m \).
A PCP Example

Let \( P = \left\{ \begin{array}{c}
\begin{array}{c}
a \\
ab
\end{array}
\end{array},
\begin{array}{c}
ab \\
bb
\end{array},
\begin{array}{c}
ba \\
aa
\end{array},
\begin{array}{c}
bc \\
cc
\end{array},
\begin{array}{c}
ca \\
aa
\end{array},
\begin{array}{c}
\text{cd} \\
\text{d}
\end{array}\right\} \). (I’ve numbered the tiles to make it easier to talk about them.)

Does the PCP problem \( P \) have a solution?
Another PCP Example

Let $P = \left\{ \begin{array}{cccc}{ \text{a}} & {\text{b}} & {\text{c}} & {\text{c}} \\ {\text{ab}} & {\text{cc}} & {\text{b}} & {\text{d}} \\ \end{array} \right\}$.

Does the PCP problem $P$ have a solution?
Another PCP Example

Let \( P = \left\{ \begin{array}{c} \begin{array}{c} a \\ ab \\ 1 \end{array} , \begin{array}{c} b \\ cc \\ 2 \end{array} , \begin{array}{c} c \\ 3 \end{array} , \begin{array}{c} d \\ ddddd \\ 4 \end{array} , \begin{array}{c} ddest \\ e \\ 6 \end{array} \end{array} \right\} \).  

Does the PCP problem \( P \) have a solution?

- \( P \) has a solution iff \( \exists n. (2^n \mod 5) = 3 \).

- Yes (let \( n = 3 \)).
PCP is undecidable

- Proof by computational histories.

- Sketch:
  - Start with a pair that has the initial configuration for a TM on the bottom and an empty string on top.
  - Include pairs in $P$ whose top strings match the current configuration, and whose bottom strings build the next configuration.
  - A bunch of details to:
    - Account for moving the tape head.
    - Extend the tape with blanks when needed.
    - Force the first pair of a solution to be the one that gives the initial configuration.
    - ...

- A Simplifying Assumption:
  - We’ll assume that any solution must start with tile 1 — we’ll call this the “Modified Post Correspondence Problem” (MPCP).
  - (Don’t worry.) We’ll remove this assumption later.
Tile 1

- We’ll reduce $A_{TM}$ to MPCP.
- Let $M\#w$ be a string where $M$ describes a TM and $w$ describes an input string to $M$.
- The first tile will give the initial TM configuration as the bottom string, and an empty string on top. We’ll use $\#$ (with $\# \notin \Gamma$) as the end marker for configurations.

\[
\begin{array}{c}
# \\
\#q_0w\# \\
\end{array} \quad \in \quad P
\]
From one configuration to the next

At each step, we copy the current configuration from the bottom string to the upper string, and build the next configuration on the lower string:

\[
\begin{array}{c}
\#C_0 \#C_1 \# \ldots \#C_{k-1} \\
\#C_0 \#C_1 \# \ldots \#C_{k-1} \#C_k \\
\end{array}
\rightarrow
\begin{array}{c}
\#C_0 \#C_1 \# \ldots \#C_{k-1} \#C_k \# \\
\#C_0 \#C_1 \# \ldots \#C_{k-1} \#C_k \#C_{k+1} \\
\end{array}
\]

A configuration looks like \( \alpha bqc \beta \).

To calculate the next configuration, we

- Copy \( \alpha \) to the upper and lower strings.
- Copy \( \alpha bqc \) to the upper string and write its successor to the lower string.
- Copy \( \beta \) to the upper and lower strings.

To copy \( \alpha \) and \( \beta \) we include the following tile in \( P \) for each \( c \in \Gamma \):

\[
\begin{array}{c}
c \\
c \\
\end{array}
\]

The next two slides describe how to handle transitions.
For each transition \( \delta(q, c) = (q', c', R) \):

- We add the tile \( \begin{array}{c} \text{qc} \\ \text{c'}q' \end{array} \) to \( P \). This enables the move:

\[
\begin{array}{c}
\# \ldots \# \alpha \\
\# \ldots \# \alpha qc \beta \# \alpha \\
\end{array} \rightarrow \begin{array}{c}
\# \ldots \# \alpha qc \beta \# \alpha c' q' \\
\end{array}
\]

- If \( c = \square \), we also add the tile \( \begin{array}{c} \text{q#} \\ \text{c'q'#} \end{array} \) to handle the case when the head is moving further into the infinite string of blanks at the end of the tape.
All the Left Moves

For each transition $\delta(q, c) = (q', c', L)$:

- for each $b \in \Gamma$ we add the tile $\frac{bqc}{q'bc'}$ to $P$. This enables the move:

$$
\begin{array}{c}
\# \ldots \# \alpha \\
\# \ldots \# \alpha bc \beta \# \alpha \\
\end{array}
\rightarrow
\begin{array}{c}
\# \ldots \# \alpha bqc \\
\# \ldots \# \alpha bc \beta \# \alpha q'bc' \\
\end{array}
$$

- We also add the tile $\frac{qc}{q'c'}$ to $P$ to handle the case when the head is at the left end of the tape.
The End Game

- \( M \) accepts \( w \) iff we can reach a configuration for our MPCP

\[
\begin{array}{c}
#C_0 \ldots #C_{n-1} # \\
#C_0 \ldots #C_{n-1} # \alpha q_{\text{accept}} \beta #
\end{array}
\]

- Now we have to “fix” the problem that we’ve got one more configuration on the lower tape than the upper one. For each \( c \in \Gamma \) we add the tiles:

\[
\begin{array}{c|c}
q_{\text{accept}} c & q_{\text{accept}} c \\
q_{\text{accept}} & q_{\text{accept}}
\end{array}
\]

- These allow us to discard one tape symbol each time we copy the configurations until we get to:

\[
\begin{array}{c}
#C_0 \ldots #q_{\text{accept}} c # \\
#C_0 \ldots #q_{\text{accept}} c # q_{\text{accept}} #
\end{array}
\]

So, we add one more tile to our set:

\[
q_{\text{accept}} # #
\]

- Now, we have an instance of MPCP that has a solution iff \( M \) accepts \( w \).
We need to force our $\text{tile}_1$ (see slide 6) to be the first tile of any solution.

Let $\star$ be a new symbol (i.e. not in $\Gamma \cup \{\#\}$).

For any string, $s$, let $\star s$ be the string obtained by inserting a $\star$ before each symbol of $s$. For example, $\star (abc) = \star a \star b \star c$.

For any string, $s$, let $s\star$ be the string obtained by adding a $\star$ before each symbol of $s$. For example, $(abc)\star = a \star b \star c\star$.

Finally, $\star s \star$, puts on star between each pair of symbols of $s$ and one star at the beginning of $s$ and one at the end. For example, $\star (abc)\star = \star a \star b \star c\star$. 
From MPCP to PCP

- Given a set of tiles, $P$ for MPCP as described above:
  - Replace the initial tile, $\#q_0w\#\star$, with $\starq_0w\#\star$.
  - Replace the final tile, $q_{accept}##$, with $\starq_{accept}\#\star\#$.
  - For every other tile, $t\ b$, replace it with $\star t\ b\star$.

Now, $\starq_0w\#\star$ must be the first tile of any solution because it is the only tile that starts and ends with the same symbol.

- We have reduced computational histories for $A_{TM}$ to PCP.
  \[ \therefore \text{PCP is undecidable.} \]
Summarizing Reductions

- Turing computable functions.
- Mapping reductions.
- Using reductions to show non-decidability.
- Examples
Turing computable functions

- $f : \Sigma^* \rightarrow \Sigma^*$ be a function over strings of $\Sigma$, a finite alphabet.
- $f$ is Turing computable (henceforth, “computable”) iff there is some TM that on every input $w$ halts with $f(w)$ (and nothing but $f(w)$) on its tape.

Examples of computable functions:
- addition, subtract, multiplication of integers encoded as binary (or unary, or decimal, or any base you like) strings.
- Sorting a list of strings into lexigraphical order.
- solution of the Traveling Salesman Problem.

Examples we’ve seen in this class
- Transforming a description of a TM (and possibly its input) into the description of another TM (and possibly its input).
- Transforing the description of a TM (and possibly its input) into a string describing another kind of machine such as a PDA, CFG, PCP problem, etc.
Language $A$ is **mapping reducible** to language $B$ iff there is a computable function, $f$ such that for every $w$:

$$w \in A \iff f(w) \in B$$

We write $A \leq_M B$ to indicate that $A$ is mapping reducible to $B$.

Mapping reducibility is a reflexive and transitive relation:

$$A \leq_M A$$

$$(A \leq_M B) \land (B \leq_M C) \Rightarrow A \leq_M C$$
Mapping and Decidability

- If $A \leq_M B$ and $B$ is Turing decidable, then $A$ is decidable.
  - Likewise if $B$ is Turing recognizable so is $A$.
  - And so on for co-recognizable, and any other complexity class you want to name.

- If $A \leq_M B$ and $A$ is not Turing decidable, then $B$ is not Turing decidable either.
Mapping Examples

- We’ve shown $\overline{A_{TM}} \leq_M E_{TM}$ to show that $E_{TM}$ is undecidable (Oct. 31).

- We’ve shown $A_{TM} \leq_M REGULAR$ and $\overline{A_{TM}} \leq_M REGULAR$ to show that REGULAR is undecidable (in fact it is neither Turing recognizable nor Turing co-recognizable) (Nov. 7).

- We’ve shown $\overline{A_{TM}} \leq_M E_{LBA}$ (using computational histories) to show that $E_{LBA}$ is undecidable (Nov. 10).

- Let $CFALL = \{ G \mid G$ describes a CFG and $L(G) = \Sigma^* \}$. We’ve shown $\overline{A_{TM}} \leq_M CFALL$ (using computational histories) to show that $CFALL$ is undecidable (Nov. 10).

- We’ve shown $A_{TM} \leq_M PCP$ (using computational histories) to show that the Post Correspondence Problem is undecidable (today).
This coming week (and beyond)

- **Reading**
  - Today: Sipser 5.3
  - Nov. 14 (Friday): Sipser 7.1
  - Nov. 17 (Monday): Sipser 7.2
  - Nov. 19 (A week from today): Tutorial by Brad Bingham

- **Homework**
  - Nov. 14 (Friday): HW 10 goes out.
  - Nov. 17 (Monday): HW 9 due.