

Post Correspondence Problem, PCP

Teodor Rus

`rus@cs.uiowa.edu`

The University of Iowa, Department of Computer Science

A puzzle kind description

Puzzle: consider a collection of dominos like:

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

A match in this puzzle is a list of these dominos (repetitions permitted) so that the string we get by reading off the symbols on the top is the same as the string of symbols on the bottom

Example: $\left\{ \left[\frac{a}{ab} \right] \left[\frac{b}{ca} \right] \left[\frac{ca}{a} \right] \left[\frac{a}{ab} \right] \left[\frac{abc}{c} \right] \right\}$ is a match because reading off the top string we get `abcaaabc` and reading off the bottom string we get `abcaaabc`, which is the same.

Note

- We can also depict a match by deforming the dominos so that the corresponding symbols from top and bottom line up, Figure 1

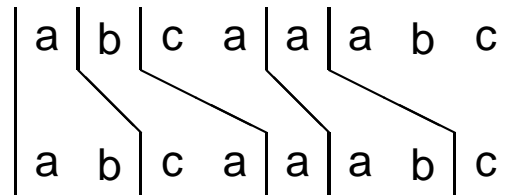


Figure 1: Deforming dominos

Observation

For some puzzle finding a match may not be possible.

For example, the puzzle $\left\{ \left[\frac{abc}{ab} \right], \left[\frac{ca}{a} \right], \left[\frac{acc}{ba} \right] \right\}$ cannot contain a match

Reason: top strings are longer than the corresponding bottom strings

Post correspondence problem:

Determine whether a collection of dominos has a match.

Note: this problem is unsolvable by algorithms

Mathematical formulation

An instance of PCP is a collection of dominos

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

A match is a sequence i_1, i_2, \dots, i_l of P components where
 $t_{i_1} t_{i_2} \dots t_{i_l} = b_{i_1} b_{i_2} \dots b_{i_l}$

PCP: determine whether P has a match.

Language:

$$PCP = \{ \langle P \rangle \mid P \text{ instance of PCP with a match} \}$$

Other formulations

(see Fleck and Hopcroft for example)

An instance of PCP over Σ consists of:

1. Two lists: $A = x_1, \dots, x_n$ and $B = y_1, \dots, y_n$, $x_i, y_i \in \Sigma^*$, $1 \leq i \leq n$;
2. The question: **do there exists a sequence i_1, \dots, i_m of integers such that $x_{i_1} \dots x_{i_m} = y_{i_1} \dots y_{i_m}$?**

Note: To show the correspondence, the lists A and B can be written: $\{[\frac{x_1}{y_1}], \dots, [\frac{x_n}{y_n}]\}$, $x_i, y_i \in \Sigma^*$, $1 \leq i \leq n$.

Theorem 5.15

PCP is undecidable

Proof idea: by reduction from A_{TM} via accepting computation histories.

- Show that from a TM M and input w we can construct an instance P of PCP where a match is an accepting computation history for M on w ;
- If we could determine whether this instance of PCP has a match, we would be able to determine whether M accepts w ;
- Since A_{TM} is undecidable we cannot determine whether P has a match.

Constructing P from (M, w)

- Choose dominos in P so that making a match forces a simulation of M on w to occur;
- The match must start with the initial configuration of M on w , i.e., with $C_1 \models C_2$ where $C_1 = q_0 w$;
- In the match construction, each domino links a position or positions in one configuration with the corresponding one(s) in the next configuration, following the transition function of M , $C_i \stackrel{\delta}{\models} C_{i+1}$.

Technical details

- For convenience in the construction of P we assume that M on w never attempts to move its head off the left-hand end of the tape (we need to alter M to prevent such behavior);
- Alter PCP to require that a match starts with the first domino, $[\frac{t_1}{b_1}]$ where $t_1 = \#$ and $b_1 = \#q_0w$. This is called modified PCP, $MPCP$;
- **Formally:**
 $MPCP = \{ \langle P \rangle \mid P \text{ instance of PCP with a match that starts with first domino} \}$

Note: we will show later how to remove this requirement.

Proof

Let TM R decide the PCP and construct S deciding A_{TM} .

- Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ and $w \in \Sigma^*$;
- The TM S that decides A_{TM} constructs an instance of the PCP that has a match iff M accepts w ;
- S constructs first an instance P' of MPCP. This construction has seven parts to be carried out shortly;
- To convert P' to P , an instance of PCP, build the requirement of starting with first domino directly in the problem, so no need to state it explicitly.

Trick for mapping P' into P

- Let $u = u_1 u_2 \dots u_n$ a string of length n . Define the following three strings:

$$\star u = \star u_1 \star u_2 \star u_3 \star \dots \star u_n$$

$$u \star = u_1 \star u_2 \star u_3 \star \dots u_n \star$$

$$\star u \star = \star u_1 \star u_2 \star u_3 \dots \star u_n \star$$

- $\star u_{i_1} \star \dots \star u_{i_k} \star = \star u_{j_1} \star \dots \star u_{j_k} \star$ iff $u_{i_1} \dots u_{i_k} = u_{j_1} \dots u_{j_k}$

Mapping P' into P

If P' were $P' = \{[\frac{t_1}{b_1}], [\frac{t_2}{b_2}], [\frac{t_3}{b_3}], \dots, [\frac{t_k}{b_k}]\}$ where we require that a match must start with the puzzle $[\frac{t_1}{b_1}]$ then

The PCP instance of P' which does not require the match to start with the puzzle $[\frac{t_1}{b_1}]$ can be obtained from P' by the following transformation:

$$P = \{[\frac{*t_1}{*b_1*}], [\frac{*t_1}{b_1*}], [\frac{*t_2}{b_2*}], [\frac{*t_3}{b_3*}], \dots, [\frac{*t_k}{b_k*}], [\frac{* \diamond}{\diamond}]\}$$

Observations

1. Considering P an instance of the PCP, we see that only domino that could possibly start a match is the first one, $\left[\frac{*t_1}{*b_1*} \right]$. This is because it is the only domino where both the top and the bottom start with the same symbol, namely $*$;
2. Beside forcing the match to start with first domino, the presence of $*$ -s doesn't affect possible matches because they simply interleave with the original symbols, which occur in the even positions of the match;
3. The domino $\left[\frac{* \diamond}{\diamond} \right]$ is there to allow the top to add the extra $*$ at the end of the match.

Construction of S : Part 1

Put $\left[\frac{\#}{\#q_0w_1w_2\dots w_n\#} \right]$ into P' as its first domino $\left[\frac{t_1}{b_1} \right]$

Note: because P' is an instance of MPCP the match must begin with this domino. The bottom string begins correctly with $C_1 = q_0w_1w_2\dots w_n$, the first configuration of the accepting computation history for M on w , as seen in Figure 2

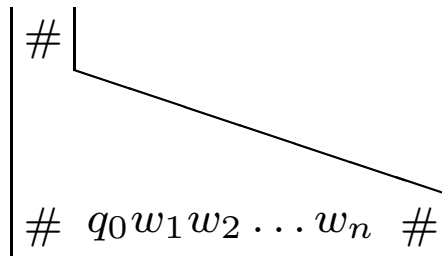


Figure 2: Beginning of the MPCP match

Observations

- In the depiction in Figure 2 of the partial match, bottom string consists of $\#q_0w_1w_2 \dots w_n\#$ and top string only of $\#$.
- To get a match we need to extend the top string to match the bottom. This is achieved by dominos $\left[\frac{c}{c}\right]$, for all $c \in \Gamma \cup Q$

Constructing a solution

- A solution to PCP is obtained by extending the previous match with the next configurations of TM M ;
- The next configuration of M will appear at the extension of bottom string of previous configuration by forcing a single step simulation of M ;
- Then by dominos $[\frac{c}{c}]$, $c \in \Gamma \cup Q$ the top and bottom match is constructed.

Facts

1. In part 2,3,4 of the construction we add to P' dominos that perform the main part of the simulation;
2. Part 2 handle head motions to the right, part 3 handles head motions to the left, and part 4 handles the tape cells not adjacent to the head.

Construction of S : Part 2

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ and $w \in \Sigma^*$. For every $a, b \in \Gamma$ and every $q, r \in Q, q \neq q_r$, if $\delta(q, a) = (r, b, R)$ put $\left[\frac{qa}{br}\right]$ into P' .

Pictorial: if $xqay \mapsto xbray$ then $\left[\frac{qa}{br}\right]$ must be placed in P' .

Construction of S : Part 3

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ and $w \in \Sigma^*$. For every $a, b, c \in \Gamma$ and every $q, r \in Q, q \neq q_r$, if $\delta(q, a) = (r, b, L)$ put $\left[\frac{cqa}{rcb}\right]$ into P' .

Pictorial: if $xcqay \mapsto xrcby$ then $\left[\frac{cqa}{rcb}\right]$ must be placed in P' .

Construction of S : Part 4

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ and $w \in \Sigma^*$. For every $a \in \Gamma$, put $\begin{bmatrix} a \\ a \end{bmatrix}$ into P' .

Illustration: to construct a hypothetical example showing what we did so far we choose:

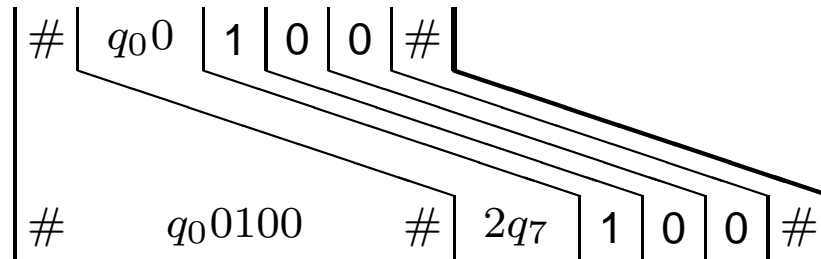
$$\Sigma = \{0, 1, 2\}, \Gamma = \{0, 1, 2, \sqcup\}, w = 0100, \delta(q_0, 0) = (q_7, 2, R)$$

Illustration, continuation

- Part 1 places $[\frac{\#}{\#q_00100\#}] = [\frac{t_1}{b_1}]$ in P' ;
- Part 2 places $[\frac{q_00}{2q_7}]$ in P' as $\delta(q_0, 0) = (q_7, 2, R)$;
- Part 4 places the dominos $[\frac{0}{0}], [\frac{1}{1}], [\frac{2}{2}], [\frac{\sqcup}{\sqcup}]$ in P' .

Thus, $P' = \{[\frac{\#}{\#q_00100\#}], [\frac{q_00}{2q_7}], [\frac{0}{0}], [\frac{1}{1}], [\frac{2}{2}], [\frac{\sqcup}{\sqcup}]\}$

The match constructed so far



Construction of S : Part 5

Put $\left[\frac{\#}{\#}\right]$ and $\left[\frac{\#}{\square\#}\right]$ into P' .

- The first of these dominos allows us to copy the symbol $\#$ that marks the separation of configurations;
- The second domino allows us to add blank symbol \square at the end of a configuration to simulate infinitely many blanks to the right that are suppressed when we write the configuration.

Continue the example

Assume that in state q_7 , upon reading 1, M goes into state q_5 , writes 0, and moves head to the right, i.e., $\delta(q_7, 1) = (q_5, 0, R)$, i.e., $[\frac{q_7 1}{0 q_5}] \in P'$.

Hence, latest partial match in Figure 3

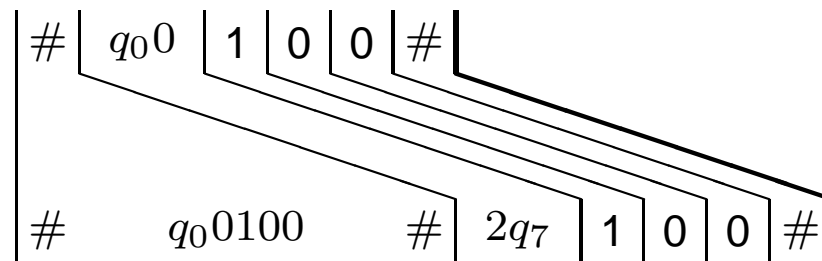


Figure 3: A partial match

extends with the match in Figure 4:

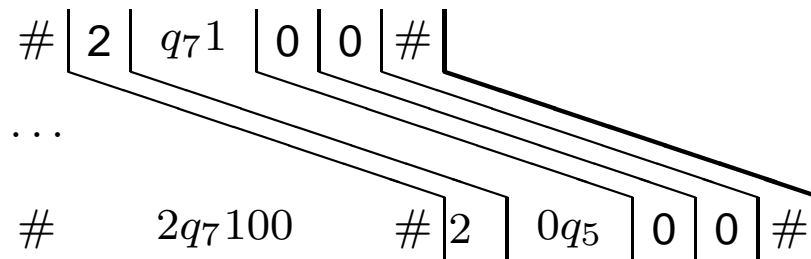


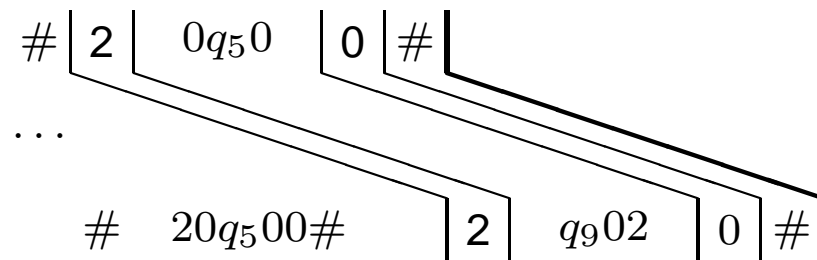
Figure 4: A partial match

Fact

If $\delta(q_5, 0) = (q_9, 2, L)$ then we have dominos

$$\left[\frac{0q_50}{q_902} \right], \left[\frac{1q_50}{q_912} \right], \left[\frac{2q_50}{q_922} \right], \left[\frac{\sqcup q_50}{q_9\sqcup 2} \right]$$

Note: the first domino is relevant because symbol to the left of the head is 0 and the preceding partial match extends to:



Observations

- As we construct the match we are forced to simulate M on w ;
- This process continues until M reaches a halting state;
- If an accept state occurs, we want to let the top of partial match catch up with the bottom. This is done by Part 6.

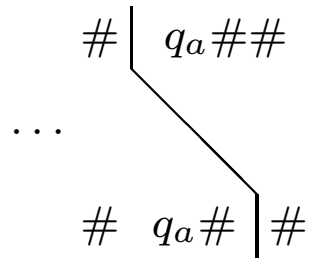
Construction of S : Part 6

For every $a \in \Gamma$, put $[\frac{aq_a}{q_a}]$, $[\frac{q_a a}{q_a}]$ into P' .

Part 6 has the effect of adding “pseudo-steps” to the TM M after it halted, where the head “eats” adjacent symbols until non are left.

Construction of S : Part 7

Finally, add the domino $\left[\frac{q_a \# \#}{\#} \right]$ and complete the match:



Application

Problem: is there an algorithm that decides if a context-free grammar is ambiguous?

Language: $AMBIG_{CFG} = \{\langle G \rangle \mid G \text{ is an ambiguous CFG}\}$

Remember: A CFG $G = (N, \Sigma, P, S)$ is ambiguous if there exists a string $w \in L(G)$ which has at least two leftmost derivations from S

Solution: we will show that $AMBIG_{CFG}$ is undecidable

Proof idea

- Reduce PCP to $AMBIG_{CFG}$. Since PCP is undecidable this proves that $AMBIG_{CFG}$ is also undecidable
- To perform the reduction we assume that an instance P of PCP is given, $P = \{[\frac{t_1}{b_1}], [\frac{t_2}{b_2}], \dots, [\frac{t_k}{b_k}]\}$
- Construct the CFG $G = (\{T, B, S\}, \Sigma, P, S)$ where $\Sigma = \{t_1, \dots, t_k, b_1, \dots, b_k, a_1, \dots, a_k\}$ and P is the set of the rules:
 $S \rightarrow T|B$
 $T \rightarrow t_1Ta_1 | \dots | t_kTa_k | t_1a_1 | \dots | t_ka_k$
 $B \rightarrow b_1Ba_1 | \dots | b_kBa_k | b_1a_1 | \dots | b_ka_k$
Note: a_1, \dots, a_k are terminal symbols different from $t_i, b_i, 1 \leq i \leq k$.
- Show that this reduction works

Proof

P has a solution iff CFG G is ambiguous.

if: assume that P has a match and show that G is ambiguous

- If P has a match $t_{i_1}t_{i_2} \dots t_{i_l} = b_{i_1}b_{i_2} \dots b_{i_l}$
- Then the string $t_{i_1}t_{i_2} \dots t_{i_l}a_{i_l} \dots a_{i_2}a_{i_1} = b_{i_1}b_{i_2} \dots b_{i_l}a_{i_l} \dots a_{i_2}a_{i_1}$ has two different leftmost derivations, one from T and one from B , hence G is ambiguous

Derivations:

$$S \Rightarrow T \Rightarrow t_{i_1}Ta_{i_1} \Rightarrow^* t_{i_1} \dots t_{i_l}a_{i_l} \dots a_{i_1}$$

$$S \Rightarrow B \Rightarrow b_{i_1}Ta_{i_1} \Rightarrow^* b_{i_1} \dots b_{i_l}a_{i_l} \dots a_{i_1}$$

Proof, continuation

only if: assume that G is ambiguous and show that P has a match

- If G is ambiguous, some string w has multiple leftmost derivations
- But all strings generated by G have the form: $w_{start}a_{i_l} \dots a_{i_2}a_{i_1}$
where w_{start} contains only symbols from P 's alphabet
- In addition, once we choose the first rule of the derivation, $S \rightarrow T$ or $S \rightarrow B$, the following steps in the derivation are uniquely determined by the sequence $a_{i_l} \dots a_{i_2}a_{i_1}$

Proof, continuation

- **Consequently:** w_{start} has at most two leftmost derivations:
 1. $S \Rightarrow T \Rightarrow t_{i_1} T a_{i_1} \Rightarrow t_{i_1} t_{i_2} T a_{i_2} a_{i_1} \Rightarrow \dots \Rightarrow t_{i_1} t_{i_2} \dots t_{i_l} a_{i_l} \dots a_{i_2} a_{i_1}$
 2. $S \Rightarrow B \Rightarrow b_{i_1} B a_{i_1} \Rightarrow b_{i_1} b_{i_2} B a_{i_2} a_{i_1} \Rightarrow \dots \Rightarrow b_{i_1} b_{i_2} \dots b_{i_l} a_{i_l} \dots a_{i_2} a_{i_1}$
- Since $t_{i_1} t_{i_2} \dots t_{i_l} a_{i_l} \dots a_{i_2} a_{i_1} = b_{i_1} b_{i_2} \dots b_{i_l} a_{i_l} \dots a_{i_2} a_{i_1}$ it results that $t_{i_1} t_{i_2} \dots t_{i_l} = b_{i_1} b_{i_2} \dots b_{i_l}$.
- Hence, the initial PCP P has a match.

A decidable version of PCP

Problem: PCP over the alphabet $\Sigma = \{1\}$ is decidable

Proof: we describe a TM M that decide the unary PCP

• Consider the unary instance of PCP $P = \{[\frac{1^{a_1}}{1^{b_1}}], \dots, [\frac{1^{a_n}}{1^{b_n}}]\}$.

• The machine M performs as follows:

$M =$ "On input $\langle a_1, b_1, \dots, a_n, b_n \rangle$:

1. Check if $a_i = b_i$ for some i ; if so, *accept*
2. Check if there exist i, j such that $a_i > b_i$ and $a_j < b_j$. If so, *accept*; otherwise *reject*."

Rationale

- In the first stage M check for a single domino which forms a solution.
- In the second stage, M looks for two dominos which form a solution; if it finds such a pair M can construct a solution by picking $(b_j - a_j)$ of the i -th domino and putting them together with $(a_i - b_i)$ pieces of j -th domino.
- The construction in the first case has
$$a_i(b_j - a_j) + a_j(a_i - b_i) = a_i b_j - a_j b_i$$
 1-s on the top, and
$$b_i(b_j - a_j) + b_j(a_i - b_i) = a_i b_j - a_j b_i$$
 1-s on the bottom.
- If neither stages of M accept, the problem instance contains dominos with all the upper parts having more (or less) 1-s than lower parts. In such a case no solution can exist. Therefore M rejects

PCP over $\Sigma = \{0, 1\}$ is undecidable

Proof idea: by reduction from PCP.

Reduction:

- Construct a computable function, *Encode*, that takes a PCP instance of any finite alphabet and produces a binary alphabet PCP, called BPCP.
- Since PCP is undecidable so is BPCP

Constructing BPCP from PCP

- For any PCP instance A over a finite alphabet $\{a_1, \dots, a_n\}$ the function $Encode$ encodes each a_i into the binary as 10^i
- Thus, $Encode$ is a mapping reduction from PCP problem to BPCP problem.
- For any instance P of PCP, the converted instance P' is in BPCP because a match for P becomes a match for P' using the same set of dominos after conversion.
- Conversely, if P' is in BPCP, P is in PCP, because we can uniquely decode a match in P' to a match in P .

Consequently $Encode$ is a mapping reduction from PCP to BPCP