The Post Correspondence Problem

Emil L. Post (1897-1954)

Statement of the PCP

given two lists of words $a_1, a_2, .., a_N$ and $b_1, b_2, ..., b_N$ of same length N, a *solution* to the problem consists in a sequence of indices $(i_k)_{1 \leq k \leq K}$ for some $K \geq 1$ and $1 \leq i_k \leq N \forall k$, such that:

$$a_{i_1} ... a_{i_K} = b_{i_1} ... b_{i_K} \quad .$$

Example:

$$\left[\frac{bc}{ca}\right] \quad \left[\frac{a}{ab}\right] \quad \left[\frac{ca}{a}\right]\left[\frac{abc}{c}\right] \longrightarrow \left[\frac{a}{ab}\right] \quad \left[\frac{bc}{ca}\right] \quad \left[\frac{a}{ab}\right]\left[\frac{abc}{c}\right]$$

"Domino Tile"

PCP Theorem:

The Post Correspondence Problem is undecidable, meaning there is no algorithm (Turing machine) that, given an arbitrary PCP instance, always halts and correctly decides whether a solution exists.

NOTES:

- It does not mean that you cannot solve certain instances, but that a general algorithm does not exist.
- Some instances are indeed “easy”, some others can be checked by brute force in finite time.
- PCP is a finite combinatorial system where searching for a match can always be extended, and there is no computable bound on how long you may need to search before knowing that no solution exists.
- **Remember: an instance has a finite number of “tiles”, but you could re-use many times any of them!**

“Difficulty of PCP”

The instance’s $\left[\begin{matrix} a \\ b \end{matrix}\right] \quad \left[\begin{matrix} ab \\ a \end{matrix}\right] \quad \left[\begin{matrix} b \\ bab \end{matrix}\right]$ shortest solution has 44 tiles (!)

A “brute-force” algorithm for an N-tiles instance would be equivalent to trying all the possible sequences up to a length L.
There are:

- N sequences with length 1
- N^2 sequences with length 2 … and so on up to N^L

Therefore we have to test the matching on $\sum_{i=1}^{L} N^i = \frac{N^{L+1} - N}{N - 1} \rightarrow O(N^L)$ strings.

The problem is not the exponential (time) complexity, but the fact that L is never known in advance. **There is no computable upper bound for L.**

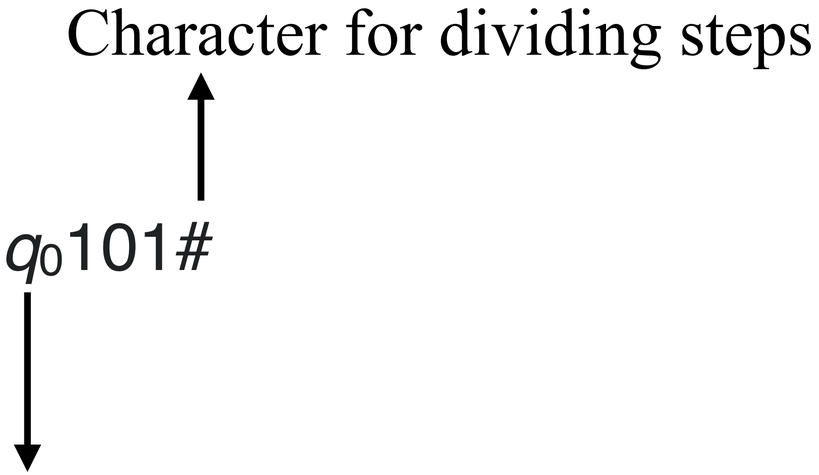
Proof

Proof ideas:

1. We will try to link PCP to a Turing machine
2. The core of the proof is constructing a Turing machine which represent an instance of PCP.
3. The idea is to encode the strings of a “domino tile” with the configuration (state, tape content, head position) of the Turing machine.
4. The full instance (sequence of tiles) will be encoded as a “computational history” of the machine, i.e. the sequence of configurations traversed during the computation.
5. If PCP were decidable, we can decide if M halts (M is a PCP decider), but this is not possible…

Proof

Encoding of as state:



Example Computational history of a TM: $q_0101\#1q_401\#11q_21\#1q_810$.

Start state — Accept state

$$q_0101\# \longrightarrow 1q_401\#$$

$$\boxed{\delta(q_0, 1) = (q_4, 1, R)}$$

- Head in q0
- Reads 1
- Writes 1 (no change)
- Moves Right
- Enter state q_4

Intermezzo: Modified PCP

PART 0: First, for simplicity we modify the PCP problem into a modified one: MPCP. In MPCP the match always starts with the first domino tile. This constraint simplifies the proof but we will recover PCP at the end.

The problem is how to construct an instance P' of MPCP such that it is equivalent to a computational history of M while accepting w.
We define a Turing machine in the usual way $M = \{Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej}\}$
For simplifying the construction we require also: 1. the head does not move left of the starting position, 2. if $w = \epsilon$ we use the string $\sqcup$ instead of w.

Proof

Translate the computational history of the the TM into Post "tiles"

PART 1: As first domino tile we put

$$\left[\frac{\#}{\#q_0 w_1 w_2 .. w_n \#}\right] \longrightarrow \text{start-state + input string}$$

PART 2 (move R): $\forall a, b \in \Gamma$ and $\forall q, r \in Q$ $(q \neq q_{rej})$:

$$\text{if } \delta(q,a) = (r,b,R), \text{ put } \left[\frac{qa}{br}\right] \text{ in P'}$$

PART 3 (move L): $\forall a, b, c \in \Gamma$ and $\forall q, r \in Q$ $(q \neq q_{rej})$:

$$\text{if } \delta(q,a) = (r,b,L), \text{ put } \left[\frac{cqa}{rcb}\right] \text{ in P'}$$

Proof

PART 4 (copy): $\forall a \in \Gamma$:

$$\text{put} \left[\frac{a}{a}\right] \text{ in P'}$$

PART 5: This step is to terminate the PCP solution properly. It introduces special ending tiles that can only be used when an accepting configuration has appeared. Once these tiles are used, no further tiles can be appended without breaking the equality.

$$\text{put} \left[\frac{\#}{\#}\right] \text{ and } \left[\frac{\#}{\sqcup\#}\right] \text{ in P'}$$

PART 6: $\forall a \in \Gamma$:

$$\text{put} \left[\frac{aq_{acc}}{q_{acc}}\right] \text{ and } \left[\frac{q_{acc}a}{q_{acc}}\right] \text{ in P'}$$

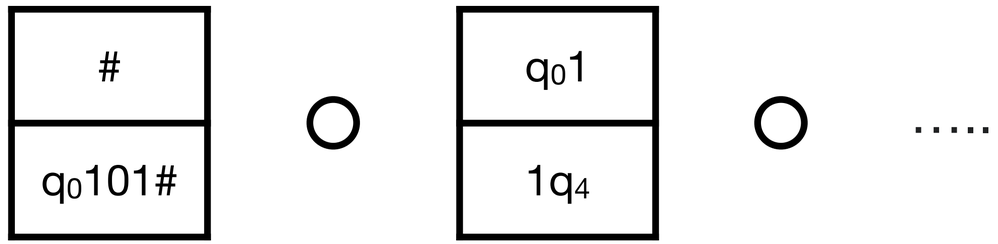
This part helps the top string "catch-up" with the bottom one which otherwise will be always longer.

PART 7: To "close" the sequence we add the final domino tile:

$$\left[\frac{q_{acc}\#\#}{\#}\right]$$

Example

Computational history of a TM: $q_0101\#1q_401\#11q_21\#1q_810$.

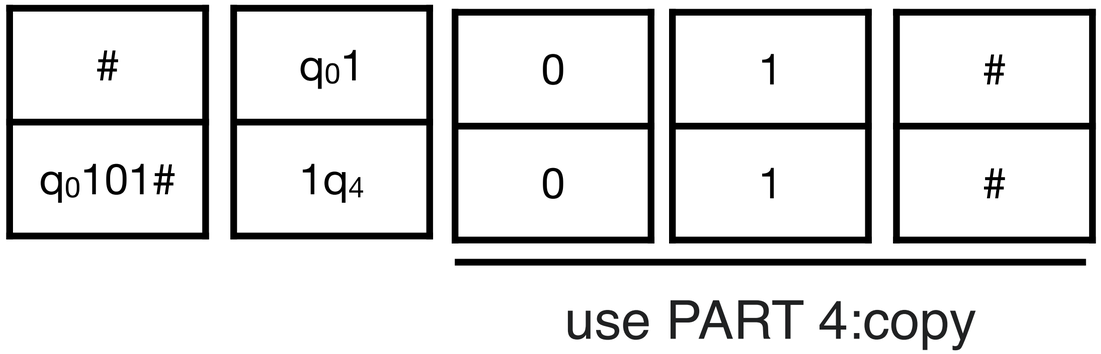


PART 1
Start

use PART 2
Move right

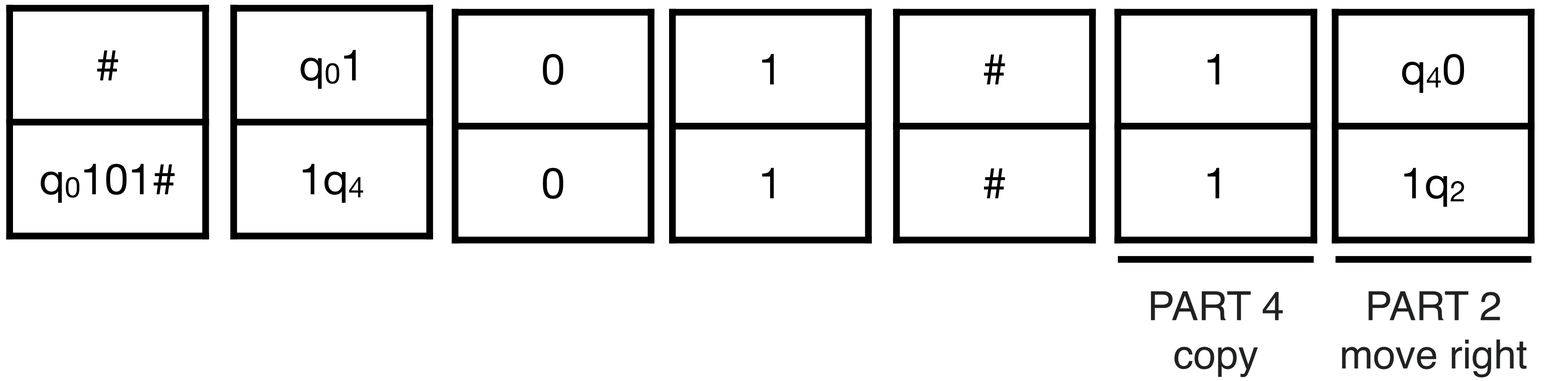
Example

Computational history of a TM: $q_0101\#1q_401\#11q_21\#1q_810$.



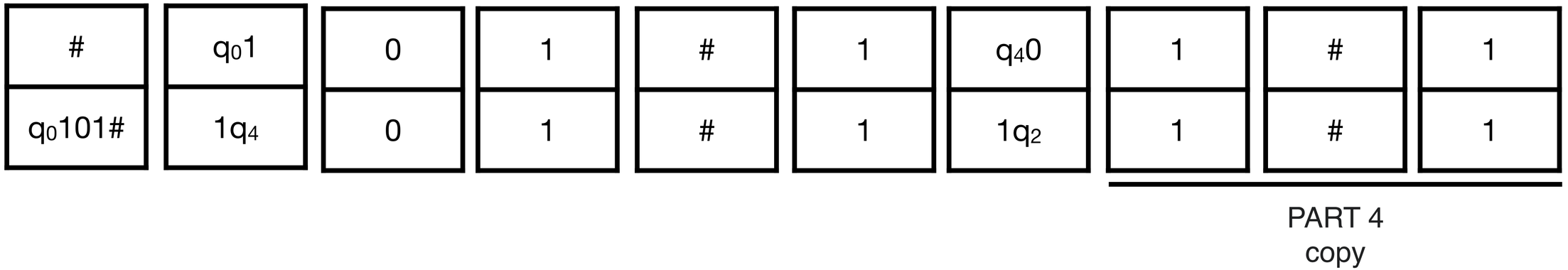
Example

Computational history of a TM: $q_0101\#1q_401\#11q_21\#1q_810$.



Example

Computational history of a TM: $q_0101\#1q_401\#11q_21\#1q_810$.



Example

Computational history of a TM: $q_0101\#1q_401\#11q_21\#1q_810$.

#	$q_0$1	0	1	#	1	$q_4$0	1	#	1	1$q_2$1
#$q_0$101#	1q_4	0	1	#	1	1q_2	1	#	1	$q_8$10

PART 3
Move Left

Example

Computational history of a TM: $q_0101\#1q_401\#11q_21\#1q_810$.

#	$q_0$1	0	1	#	1	$q_4$0	1	#	1	1$q_2$1
#$q_0$101#	1q_4	0	1	#	1	1q_2	1	#	1	$q_8$10

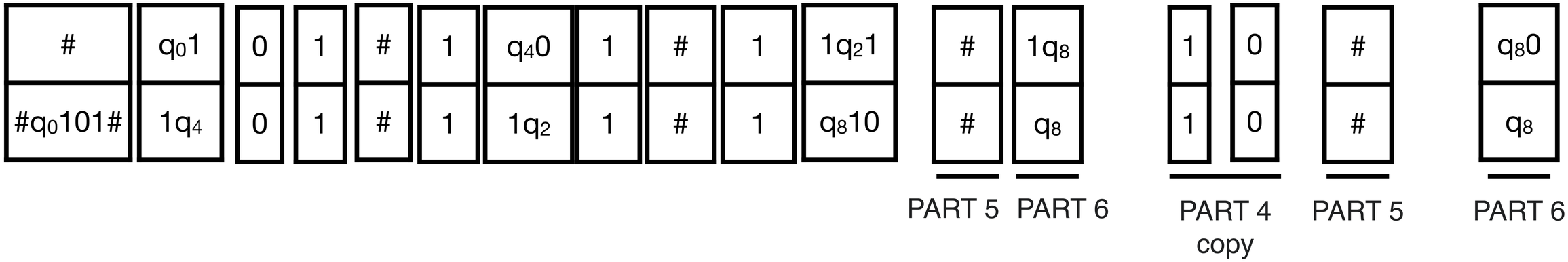
Matching strings so far:

#$q_0$101#1$q_4$01#11$q_2$1

$q_0$101#1$q_4$01#11$q_2$1#1$q_8$10 → This string is longer:
use the other rules to “catch up”…

Example

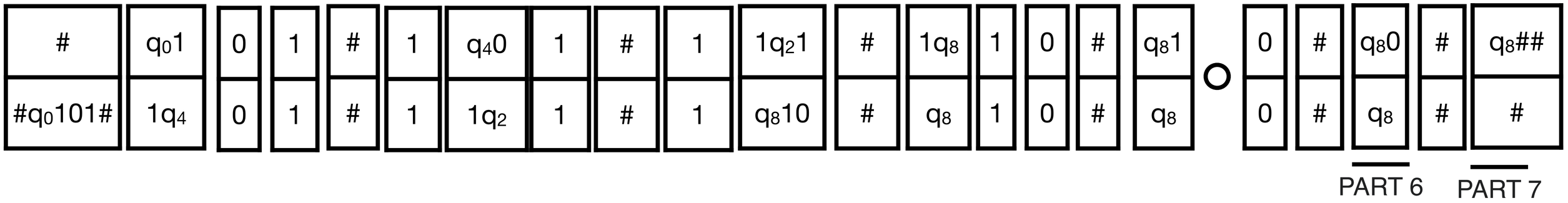
Computational history of a TM: $q_0101\#1q_401\#11q_21\#1q_810$.



Note: PART6 puts more characters on top

Example

Computational history of a TM: $q_0101\#1q_401\#11q_21\#1q_810$.

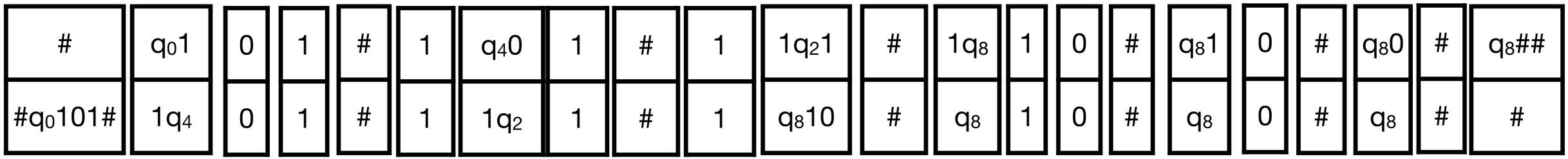


Final matching ?

$\#q_0101\#1q_401\#11q_21\#1q_810\#q_810\#q_80\#q_8\#\#$
$\#q_0101\#1q_401\#11q_21\#1q_810\#q_810\#q_80\#q_8\#\#$

See notes on how to go from the modified PCP to PCP.

Final Matching Tiles



For reconstructing 4 computational steps we used 22 “tiles”.
Again: the number of usable tiles is not fixed by the instance of the PCP.

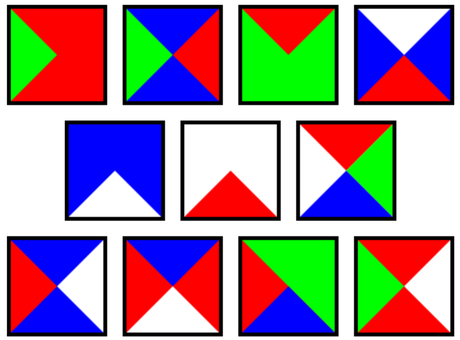
Summary

- A PCP instance is constituted by pairs of strings.
- The solution is a rearrangement (repetitions allowed) of the string pairs such that the two strings (on top and bottom) match.
- Mathematically: a valid reshuffling of the indices labelling the strings.
- We showed the equivalence between a TM computational history and PCP.
- Solution of PCP implies TM halting: undecidable.

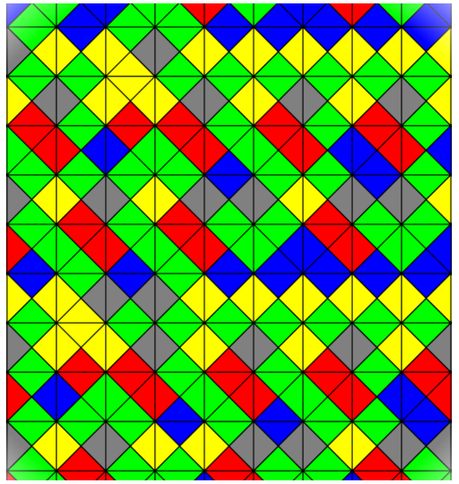
- Note: PCP can encode a TM computation, therefore it is Turing-complete (TC).
- TC: what a TM can compute, also PCP can.
- Interesting: PCP is a kind of computation where there are no machines or particular mathematical constructions: just a combinatorial problem with string matching.
- PCP: captures the “essence” of computation with very simple rules.
- Why PCP is so difficult?
    - Global constraint (matching) with local rules (local matching).

Another Undecidable Problem: Wang Tiles

Given a set of tiles like:



can we tile the plane?



The problem turns out to be undecidable: no generic algorithm exists.
Again, the proof consists in mapping the tiles into a Turing machine.