Undecidability of Post Correspondence Problem

Chatterjee Arnab

2019 November 28

Outline

- Definition of Post correspondence problem (PCP).
- Definition of modified Post correspondence problem (MPCP).
- Reduction from MPCP to PCP.
- Reduction from halting problem to MPCP.
- Demonstration using Java implementation of a halting DTM.
- PCP timeline, decidable variants and an open question.
- The PCP language and a takeaway question.
- References, citations and bibliography.

Prerequisites:

- Familiarization with Turing machines (Q, Σ, Γ, δ, $\vdash$, $\_$, s, t, r).
- Undecidability of the halting problem for Turing machines.

Post Correspondence Problem: Formal Definition

- Consider a binary alphabet having symbols a and b. Consider two non-empty finite lists U and L of non-empty strings, such that the lengths of both U and L are equal (let it be n).

- Indexing both lists from left to right starting at 0, PCP asks for a sequence of index positions $i_0, i_1, \ldots, i_{k-1}$ (indexes may repeat), such that $U[i_0].U[i_1]. \ldots .U[i_{k-1}] = L[i_0].L[i_1]. \ldots .L[i_{k-1}]$.

- Example: $U = [a, abaaa, ab]$, $L = [aaa, ab, b]$.
  Solution: [1, 0, 0, 2]. Explanation: $abaaa.a.a.ab = ab.aaa.aaa.b$.

- Question: Is is possible to obtain a solvable instance of PCP which permits only a finite number of solutions?

Post Correspondence Problem: Another Definition

- Consider a non-empty finite list of dominoes, whose upper and lower halves have non-empty strings over some given alphabet. Additionally, there are unlimited copies for each domino available.

- If the sequence of all unique dominoes is $D_0, D_1, \ldots, D_{n-1}$ then PCP can be stated as: find a sequence of (possibly repeating) dominoes $D_{i_0}, D_{i_1}, \ldots, D_{i_{k-1}}$ such that the string read off from the upper halves is same as the string read off from the lower halves.

- The solution set of any PCP instance is closed under concatenation (if two sequences of dominoes are solutions for an instance of PCP then simply joining them gives another solution for that instance).

Decidability of PCP over any Unary Alphabet

- Given a PCP instance over some unary alphabet, is it solvable?
- Necessary condition for solution: It must not be the case that for every index, the string in one of the lists is strictly shorter (or longer) than the string at the corresponding index in the other list.
- Sufficient condition for solution: Same as necessary condition. Proof: Let i and j be two index positions such that $|U[i]| \leq |L[i]|$ and $|U[j]| \geq L[j]$. Also, let $m = |L[i]| - |U[i]|$ and $n = |U[j]| - L[j]$. Then for $k = \text{LCM}(m, n)$, $U[i]^{k/m}.U[j]^{k/n} = L[i]^{k/m}.L[j]^{k/n}$.
- Due to the existence of a condition that is both necessary and sufficient, PCP is decidable over any unary alphabet.

Undecidability of PCP over Non-Unary Alphabets

- The condition for unary alphabets is also required for non-unary alphabets; however, here it is merely necessary but not sufficient.
- Another trivial necessary condition is that there must be an index for which the string in one list is a prefix of the string at the same index in the other list. Also, there must be an index for which the string in one list is a suffix of the corresponding string in other list.
- Non-trivial conditions for solution may be found for specific PCM instances. Example: $U = \{ab, baa, aba\}$ and $L = \{aba, aa, baa\}$.
- However, it is impossible to obtain any sufficient condition for all PCP instances; else the halting problem would become decidable.

The Need for Introducing Modified PCP

- How to prove undecidability? Reduce halting problem to PCP.
- Intuition behind reduction? Given a DTM and input, find a PCP instance which is solvable if and only if DTM halts on the input.
- General idea? If DTM halts on input, then it will have a unique computation history that ends in either accept or reject state. An encoding of this computation will be the minimal PCP solution.
- Problem? PCP allows any domino to be chosen as the first one. This may generate spurious solutions even if DTM does not halt.

Definition of Modified Post Correspondence Problem

- Solution? We impose an additional condition that exactly one of the dominoes will be designated as the first one in any solution.

- What to call that? Modified PCP! Specifically, MPCP mandates that every solution must start with the first domino in the list. Example of a dominoes list solvable as PCP but not as MPCP?

- How to obtain PCP from MPCP? We must ensure that the prefix condition necessary for first domino in any solution is violated for all dominoes, except for that domino which will correspond to the initial state/tape configuration of DTM on input. In effect, we eliminate all other solutions not starting with this domino.

Reducing Modified PCP to PCP

- Introduce two new symbol * and $ that are not in input alphabet.
- For upper string of each domino, put * to left of every symbol.
- For lower string of each domino, put * to right of every symbol.
- Add a new domino based on the first domino of MPCP using the above rules, except that the lower string has an extra * at its left.
- Add a new domino whose upper string is *$ and lower string is $.

Sample MPCP Instance Reduced to PCP

- How does that help? Only the penultimate domino added is eligible as a suitable candidate to become the first element of any solution. Also the last domino is added to complete any solution.

- Example: After modifying our previous example, let the two lists $U = \{abaaa, a, ab\}$ and $L = \{ab, aaa, b\}$ represent the upper and lower halves of the list of original MPCP dominoes, respectively.

- Using the construction described, we obtain two new PCP lists:

  $U' = \{*a*b*a*a*a, *a, *a*b, *a*b*a*a*a, *\$\}$

  $L' = \{a*b*, a*a*a*, b*, *a*b*, \$\}$

MPCP is Solvable iff the Reduced PCP is Solvable

- Every MPCP solution yields a corresponding reduced PCP solution.

  Let $U[i_0].U[i_1]. \ldots .U[i_{k-1}] = L[i_0].L[i_1]. \ldots .L[i_{k-1}]$ be a solution for the MPCP. Replacing every $U[i_j]$ by $U'[i_j]$ and every $L[i_j]$ by $L'[i_j]$, our construction ensures that the LHS will have a * to the left of every symbol, and the RHS will have * to the right of every symbol.

  Both sides will still have the same length, since equal number of *s will be added on either side. However the LHS will now begin with a * but not the RHS. To solve this, we replace $U[i_0]$ by $U'[n]$ and $L'[i_0]$ by $L'[n]$, corresponding to the penultimate PCP domino created. This will not only add a * to the left of RHS but also "push" the RHS to the right by one position, making its symbols align with the LHS, except for an extra * at the right of RHS. We complete the solution by the final PCP domino given by $U'[n+1] = *\$$ and $L'[n+1] = \$$.

Proof of Correctness for MPCP to PCP Reduction

- Every minimal solution of reduced PCP yields an MPCP solution.

A solution index sequence for a PCP (or MPCP) instance is called a minimal solution if and only if it does not contain any non-trivial contiguous subsequence which is also a solution for that instance.

Let $U'[i_0].U'[i_1]. \ldots .U'[i_{k-1}] = L'[i_0].L'[i_1]. \ldots .L'[i_{k-1}]$ be a solution for the reduced PCP. From the construction of U' and L' we know that $i_0 = n$ due to prefix constraint, and also $i_{k-1} = n + 1$ due to suffix constraint. Any minimal solution cannot have a $ in between.

Remove $U'[i_{k-1}]$ and $L'[i_{k-1}]$. Both sides differ only by an extra * at right of RHS. Then replace $U'[i_0]$ by $U'[0]$ and $L'[i_0]$ by $L'[0]$. LHS now has an extra * at its left, making both sides equal in length. Replacing every $U'[i_j]$ by $U[i_j]$ and $L'[i_j]$ by $L[i_j]$, an equal number of *s are removed from both sides, giving a solution for the MPCP.

Obtaining Halting Computation History using MPCP

- If (non-blank) tape contents are $\vdash a_0 a_1 \ldots a_n$ and read head is at i, TM being at state q is denoted by snapshot $\vdash a_0 a_1 \ldots a_{i-1} q a_i \ldots a_n$.
- Introduce a new symbol # to delimit snapshots of the DTM in its computation history (ensure # does not occur in tape alphabet).
- Represent each domino of the MPCP as [upper] [lower].
- The first domino representing initial configuration of the DTM is given by [##] [##s $\vdash w$#], where s is start state and w is input string.
- After using the first domino, the upper half will be lagging behind the lower half by one computation snapshot. We ensure to keep it that way until we use a domino with t or r state in the lower half.

Copying Symbols Between Upper and Lower Halves

- We do not simulate the transitions of accept and reject states; rather they are used to make the upper half "catch up" with the lower half.
- To simulate a transition, we write its LHS on the upper half and its RHS on the lower half of a domino. This enables lower half of the partial solution to stay ahead of upper half by one computation.
- Before simulating the next transition, we need to copy the lower half symbols onto the upper half, and to the right of the lower half as well, so that the "staying ahead" condition is satisfied. We do so by using "copy" dominoes that add a new symbol to upper and lower parts of the partial solution, until the next transition domino is used.
- For each symbol a in the tape alphabet, construct a domino [a] [a]. Also add a domino [#] [#] to copy the additional delimiter symbol.

Simulating Non-Halting *L*/*R* Moves Using Dominoes

- Consider a left transition given by $\delta(q, a) \rightarrow (p, b, L)$. Since we need to know which symbol is present at the left of the read head, for each symbol c in the tape alphabet, construct a domino $[cqa]$ $[pcb]$.
- Consider a right transition given by $\delta(q, a) \rightarrow (p, b, R)$. The symbol at left of read head being irrelevant, construct a domino $[qa]$ $[bp]$.
- Based on our representation, if the read head reaches an index beyond the input for the first time, the snapshot is $\vdash a_0 a_1 \ldots a_n q$. Also, since lower half is ahead of upper half by one computation, the delimiter # will appear after q in the lower half before the next snapshot begins.
- Since # is not a part of the tape alphabet, whenever the pattern $q\#$ occurs, we need to use special dominoes to simulate the effect of extending tape on the right, thereby simulating a right-infinite tape.

Simulating Semi-Infinite Tape and Tape Erasing

- Consider a left transition given by $\delta(q, \_) \rightarrow (p, b, L)$. For each symbol c in the tape alphabet, construct a domino [cq#] [pcb#].
- For all right transitions $\delta(q, \_) \rightarrow (p, b, R)$ add a domino [q#] [bq#].
- Once we select a domino with a halting state in the lower half, we do not simulate the DTM, but focus on bridging the gap between upper and lower halves. For each tape symbol a, we add these dominoes:

  [ta] [t] , [ra] [r] Erasing the right side of the read head.

  [at#] [t#] , [ar#] [r#] Erasing left side after right side is cleared

- Construct dominoes [t##] [#] and [r##] [#] to complete the solution.
- What is a (reasonable) upper bound on the total number of dominoes? (Hint: 1 + number of copy dominoes + number of transition dominoes + number of extender dominoes + number of erasing dominoes + 2).

MPCP Solution Guarantees DTM Halts on Input

- Consider the first symbol for the extended segment of lower half. If it is a non-state symbol c, we must check the symbol next to it. If it is a state symbol q, then look for "the" domino whose upper part is $[cqa]$, where a is the symbol that follows q in extended lower half.
- Due to deterministic nature of $\delta(q, a)$, exactly one such domino will be constructed by our reduction, if and only if it is a left move.
- If such a domino is not found, then use the copy domino for c, followed by the domino whose upper half is $[qa]$, whose existence and uniqueness are both guaranteed by the determinism of TM.
- DTM always reduces to an MPCP instance for which correct move can be determined using only the next 3 symbols in the lower half.
- Since only halting states are capable of shrinking the gap between the upper and lower halves, MPCP solution implies that DTM halts.

Halting Computation Guarantees MPCP Solution

- Let DTM have a halting computation sequence for given input.
- Since the deterministic choices of MPCP corresponds to the transitions of the DTM, it generates the partial computation sequence of DTM until a halting state appears in lower half.
- As DTM halts on the given input, either the accept or reject state will appear in the lower half after using some transition domino.
- Since we do not have copy or transition dominoes for both the halting states, we are forced to deterministically erase symbols, first on the right of the halting state, and then on its left.
- When erasing phase is over, the extended lower half will be *t*# or *r*#; we use [*t*##] [#] or [*r*##] [#] to obtain the minimal solution.

PCP Timeline, Decidable Variants, An Open Question

- 1946: Emil Leon Post first proposed the original problem, and also provided a formal proof of its undecidability. [1]
- 1979: Michael Randolph Garey and David Stifler Johnson in their seminal book showed that bounded PCP is NP-complete. [4]
- 1981: Any PCP instance with only 2 domino tiles is decidable.
- 2001: Vesa Halava, Mika Hirvensalo and Ronald Michiel de Wolf proved that marked PCP is decidable. Further, they also showed that k-marked PCP is undecidable for all $k \geq 2$. [2]
- 2015: Turlough Neary proved that PCP is undecidable if number of dominoes is 5 or more, with no other additional constraint. [3]

Open question: Is PCP decidable if number of dominoes is 3 or 4?

The PCP Language and A Takeaway Question

- Is PCP capable of *simulating* any Turing machine, i. e. given an arbitrary TM, is it always possible to find an instance of PCP whose solution set is precisely the language accepted by TM?

- Consider the notion of PCP class of languages, i. e. the set of all languages that correspond to the solution set of some PCP instance. Where does this class fit into the Chomsky hierarchy?

- Takeaway question: Consider the class of all languages which are closed under complementation. Is this class same as the PCP class?

References and Citations

1. Post, Emil Leon (1946). "A variant of a recursively unsolvable problem". *Bulletin of the American Mathematical Society*. **52**: pp. 264–269.
2. Halava, Vesa; Hirvensalo, Mika; Wolf, Ronald Michiel de (2001). "Marked PCP is decidable". *Theoretical Computer Science (TCS)*. Elsevier Science. **255**: pp. 193–204.
3. Neary, Turlough (2015). "Undecidability in Binary Tag Systems and the Post Correspondence Problem for Five Pairs of Words". In Ernst W. Mayr and Nicolas Ollinger (ed.). *32nd International Symposium on Theoretical Aspects of Computer Science*. STACS 2015. **30**: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. pp. 649–661.

Bibliography

4. Garey, Michael Randolph; Johnson, David Stifler (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. William H. Freeman and Company. p. 228.

5. Hopcroft, John Edward; Motwani, Rajeev; Ullman, Jeffrey David (2006). "Post's Correspondence Problem". *Introduction to Automata Theory, Languages, and Computation* (3rd edition). Pearson Addison-Wesley. pp. 417–428.

6. Sipser, Michael Fredric (2012). "A Simple Undecidable Problem". *Introduction to the Theory of Computation* (3rd edition). Cengage Learning. pp. 227–233.

The End

Thank You.