Pedagogical Puzzles

Only one puzzle this time, which is more an open problem than a puzzle. Please send your contributions to this column to the editor.

Puzzle Let $\mathcal{F}$ be a family of subsets of a finite set X which is closed under intersection—i.e., if $A, B \in \mathcal{F}$, then $A \cap B \in \mathcal{F}$ as well. Prove or disprove the following statement: there is an element $x \in X$ which belongs to at most $|\mathcal{F}|/2$ sets in $\mathcal{F}$.

Solutions to last time's puzzles

There were three puzzles last time, all from the book *Elements of the Theory of Computation* by Harry Lewis and Christos Papadimitriou. We discuss the solutions here.

Puzzle 1 The first problem asks whether a variant of Post's Correspondence Problem (PCP) is decidable. Recall that an instance of PCP is a list of pairs $\{(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)\}$, where each u_i and v_i is a word over Σ. A match for the correspondence system is a sequence $i_1 i_2 \ldots i_n$ of indices, where each $i_j \in \{1, 2, \ldots, k\}$ such that $u_{i_1} u_{i_2} \cdots u_{i_n} = v_{i_1} v_{i_2} \cdots v_{i_n}$.

A *weak match* of a correspondence system $\{(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)\}$ is a string w such that, for some $n > 0$, $i_1, i_2, \ldots, i_n$, $j_1, j_2, \ldots, j_n$ where $1 \leq i_1, \ldots, i_n, j_1, \ldots, j_n \leq k$

- $u_{i_1} u_{i_2} \cdots u_{i_n} = v_{j_1} v_{j_2} \cdots v_{j_n} = w$.
- $i_1 i_2 \ldots i_n$ is a permutation of $j_1 j_2 \ldots j_n$

That is, w can be obtained by concatenating n first components and also by concatenating n components. The exact sequence of indices used in the two words may differ, but they are required to be permutations of each other.

Is it decidable whether a correspondence system has a weak match?

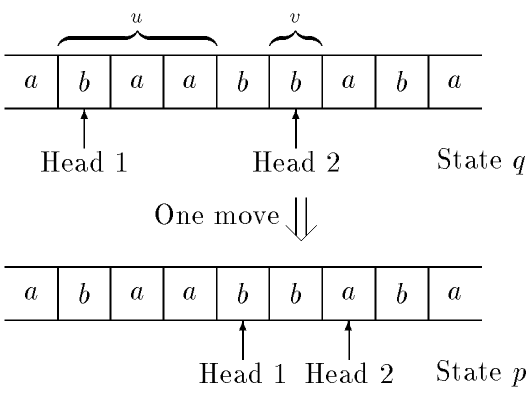
Solution sketch As yet we do not know how to solve this problem. We present a solution to a simpler variant of this problem: Given a set of pairs of strings $(u_1, v_1), \ldots, (u_n, v_n)$ do there exist integers $i_1, i_2, \ldots i_k$ and $j_1, j_2, \ldots j_k$ such that $u_{i_1} u_{i_2} \ldots u_{i_k} = v_{j_1} v_{j_2} \ldots v_{j_k}$. In other words, we just want to pick the same number of elements from each list, without any special connection between the two sets of elements picked.

The solution given below reduces the problem to that of deciding the emptiness of the language accepted by a nondeterministic pushdown automaton. The NPDA operates as follows: As it reads its input it parses it using elements of the u_i's as well as elements of the v_i's and uses the stack to keep track of the difference between the number of elements of u_i's used in the parsing and the number of v_i's used in the parsing (If u_i's have been used up in excess then as many 1's as there have been extra u_i's are stored in the stack and, on the other hand, if v_i's have been used up in excess then as many 0's as there have been extra v_i's are stored in the stack.)

The automaton accepts a string, if at the end of reading the entire string it has been fully parsed in terms of u_i's and v_i's (a condition on states) and the number of u_i's used in parsing it is the same as the number of v_i's used in parsing it (the stack is empty). Thus a string is accepted iff it is a solution to the problem described above. It is an easy exercise to turn such an automaton into an equivalent traditional NPDA that accepts by final states (or equivalently empty stack).

K Narayan Kumar, SMI, Chennai
kumar@smi.ernet.in

Puzzle 2 The second puzzle is a decision problem for finite automata with 2-heads. A 2-head finite automaton (THFA) consists of a finite control, an input tape, and two heads that can read but not write on the tape and move only left to right. The machine is started in its initial state with both heads on the leftmost square of the tape. Each transition is of the form (q, u, v, p) where q and p are states and u and v are strings. If (q, u, v, p) is a transition of M, then M can, when in state q, read u with its first head and v with its second head and enter state p, as shown in the figure. M accepts an input string by moving both heads together off the right end of the tape while entering a designated final state.



A THFA is said to be *deterministic* if there is only one choice of move from any configuration. Is the emptiness problem for deterministic THFA decidable (i.e., can we check whether a deterministic THFA M accepts at least one input)? (Note: The problem is undecidable for non-deterministic THFA.)

Solution sketch This problem is undecidable. Let us first see why the emptiness for the nondeterminisitic version of this automaton is undecidable. It is easy to reduce Post's Correspondence Problem (PCP) to the emptiness problem for these automata.

The automaton corresponding to an instance of PCP has just one state q, which is deemed accepting. The automaton has a transition of the form (q, u_i, v_i, q) for each pair of strings (u_i, v_i) in the PCP instance. For convenience let us call such a transition t_i. If the automaton has gone through the transition sequence $t_{i_1}, t_{i_2}, \ldots, t_{i_k}$ then the first head has read the string $u_{i_1} u_{i_2} \ldots u_{i_k}$ and the second head has read the string $v_{i_1} v_{i_2} \ldots v_{i_k}$. It is easy to see that a sequence of transitions $t_{i_1}, t_{i_2}, \ldots, t_{i_k}$ leads to acceptance iff the sequence $i_1 i_2 \cdots i_k$ is a solution to the PCP instance.

But note that the above automaton is nondeterministic. Even though it has only one state, more than one transition may be enabled in some configurations. There might be more than one (u_i, v_i) pair that can be read from the current head position. We will now show that this nondeterminism can be got rid off. This will be done by "marking" the string to indicate how to parse it in terms of the u_i's and v_i's.

Formally, if the strings of the PCP come from the alphabet Σ then we construct an automaton that works over the alphabet $\Sigma \times \{s, m, e\} \times \{s, m, e\}$. Here s stands for start, m for middle and e for end. If $(a_1a_2\dots a_k, b_1b_2\dots b_l)$ is a pair specified by the PCP then there are transitions labelled $(q, (a_1,s,x_1)\ \cdots\ (a_{k-1},m,x_{k-1})\ (a_k,e,x_k)\ , (b_1,y_1,s)\ \cdots\ (b_{l-1},y_{k-1},m)\ (b_l,y_l,e)\ , q)$ for each choice of x_i, y_i from $\{s,m,e\}$.

The input can be thought of as consisting of three rows. The first row consists of a string over the alphabet Σ while the second row specifies (correctly or incorrectly) how to break it up into u_i's and the third row specifies how to break it up into v_i's. For any given configuration there is at most one transition enabled corresponding to a pair (u_i, v_i) if such a pair is correctly marked in the input from the current head positions. Thus the automaton is indeed deterministic. Its language consists of precisely the solutions to the PCP problem with the second and third row indicating how to break it up into u_i's and v_i's respectively. An interested reader can fill in the details as well as "correct" the above solution to work even when the pairs specified by the PCP contain strings of length 1 or less.

P Madhusudan, IMSc, Chennai
madhu@imsc.ernet.in

Puzzle 3 A *weak tiling system* is a quadruple $\mathcal{D} = (D, D_0, H, V)$ where D is a finite set, $D_0 \subseteq D$ and $H, V \subseteq D \times D$. A *tiling* by $\mathcal{D}$ is a function $f : \mathbb{N} \times \mathbb{N} \to D$ such that the following hold:

$$\begin{array}{ll} f(m,0) \in D_0 & \text{for all } m \in \mathbb{N}. \\ (f(n,m), f(n{+}1,m)) \in H & \text{for all } m,n \in \mathbb{N}. \\ (f(n,m), f(n,m{+}1)) \in V & \text{for all } m,n \in \mathbb{N}. \end{array}$$

Show that it is undecidable whether there exists a tiling by a weak tiling system.

(Normally, a tiling system is specified as a quadruple $\mathcal{D} = (D, d_0, H, V)$, where D, H and V are as above and $d_0 \in D$. A tiling f by such a system $\mathcal{D}$ must satisfy $f(0,0) = d_0$. In the puzzle, the constraint at the origin is replaced by a weak constraint for the bottom row of the tiling. Notice that the weak constraint does *not* insist that *all* tiles from D_0 be used on the bottom row.)

A Solution Sketch Recall the standard way in which the standard tiling problem is shown to be undecidable. The halting problem for Turing machines starting with blank tape is reduced to tiling. The tiles are so designed that row i in the tiled plane contains the configuration of the Turing machine at time i. Each tile holds one cell of the Turing machine tape, and additionally maybe a state and/or left-right endmarkers and extra signals to indicate whether the head is moving left/right. If the machine halts or hangs at time i, then the tiling cannot be extended beyong row i. We assume without loss of generality that at any step, the Turing machine either rewrites the symbol on the scanned square or moves its head one square left/right, but not both. We also assume that the Turing machine never hangs (falls off the left end)—it either halts, or moves forever.

In this puzzle, we can only specify what tiles are to be used in the first row. We will use a similar reduction—given a Turing machine M, to decide whether it halts when started on blank tape, we design a tiling system with the new constraints. A major difference is that each row now contains infinite copies of the configuration of M at the corresponding time.

Let the configuration be described by a string as follows: `L a b q c B B B R`. `L` and `R` are left and right endmarkers (represented in the tiling, not present on the tape), `B` is the blank symbol, `q` is the state of the machine. In a departure from convention, we allow the configuration to contain some trailing blanks beyond the rightmost used symbol as well, as long as this number of blanks is finite.

The tiles contain one square each of the tape. In addition, the tile representing the extreme left/right square of the configuration also has the endmarker `L` or `R`, and the tile containing the scanned square has the state as well. A tile containing just `B` is called a passive tile. The configuration at time 0 is represented by the single tile containing $\mathtt{L}, q_0, \mathtt{B}, \mathtt{R}$. This is the *only* tile in D_0; thus the first row of the tiling contains the initial configuration of M repeated infinitely often.

Let the plane be tiled upto row i, and let row i contain copies of the configuration at time i, each copy using s tiles. The next row is tiled as follows. If M changes a scanned symbol, this change can be propagated to the tiles of the next row as in the standard case. So row $i+1$ also contains the same number of copies of the new configuration, each of size s. If M makes a left/right move without crossing the extreme squares in the represented configuration, again the change can be propagated to the next row.

If M makes a right move, and the tape head is already in the rightmost sqaure of the configuration, then we need to extend the size of each configuration. We allow two tiles to be placed on top of each tile containing a state and an `R` endmarker and requiring a right move (q, a, R), where $\delta(q, a) = (p, R)$. One tile has the letter `a`, no right endmarker, and a special signal signifying "propagate"; the other has a `B`, a right endmarker `R`, and a special signal "kill". For all other tiles, we allow at any stage three types of tiles to be placed above: one is the tile T that would be allowed in the standard reduction, another is the tile T with an additional signal "propagate", and the third is a passive tile with the "kill" signal. The horizontal rules are as follows: adjacent tiles must both have "propagate" or both have "kill" or both have neither signal, except at `L`/`R` boundaries. At these boundaries, a kill/propagate signal, if present, must toggle.

The effect of these horizontal rules is that if we place a propagate tile above the right cell of a copy of the configuration, then all tiles in this copy of the configuration must be of type propagate, all tiles in copies on either side must be of type kill, all tiles in the copies beyond those must be of type propagate and so on. That is, alternate copies of the configuration will be of type propagate and type kill. A kill configuration kills itself and lets its tiles become passive to be used as extra space for the previous configuration. A propagate configuration stays alive and uses extra space from its right neighbour. So at the next time step, there will be half the number of copies of the new configuration, and each copy will be twice the size.

The propagate signal must carry some more information too at the `R` boundary. Since the tape head is moving right, the state of M must now be written in the right neighbour (in a newly acquired passive neighbour). The horizontal rules can take care of this too.

The remaining design is exactly as in the standard case. So the set of tiles and the horizontal/vertical rules can be designed in an analogous fashion, with extra rules to account for the kill/propagate constraints. The details are left to the interested reader.

Meena Mahajan, IMSc, Chennai
meena@imsc.ernet.in