CSE547T Class 16

Jeremy Buhler

March 22, 2017

1 Correspondence Systems

Following problem was formulated by Emil Post.

- We are given a finite set of *tiles* $T = t_1 \dots t_n$.
- Each tile t_i contains an ordered pair of nonempty strings $[\alpha_i \mid \beta_i]$.
- If you like, you can think of symbols as labeling "top" and "bottom" halves of tile.

- **Problem (PCP)**: is there a finite sequence of tiles from T (perhaps with repetitions) such that the concatenations of the strings on their top and bottom halves are the same?
- In other words, are there $t_{i_1}, t_{i_2}, \dots t_{i_k}$, $k > 0$, such that

$$\alpha_{i_1}\alpha_{i_2} \dots \alpha_{i_k} = \beta_{i_1}\beta_{i_2} \dots \beta_{i_k}?$$

- **Solvable Example**:

- **Unsolvable Example**:

How hard is it to tell if an instance of PCP is solvable?

- Could simply enumerate all finite sequences of tiles in some canonical order (increasing in size).
- If there is a solution, we'll find it eventually.
- A TM can surely do this, so the language of solvable PCP instances T (suitably encoded) is surely RE.
- Of course, if there's no solution, above construction runs forever...

Is there some fancier way to solve the problem?

- **Theorem** (Post, 1946): PCP is undecidable.
- We will sketch a proof by showing how PCP can be used to simulate the operation of a Turing machine!
- To simplify matters, we will prove undecidability only for a restricted version of the problem, called Modified PCP (MPCP).
- In MPCP, one tile is designated as the "start" and must occur first in the solution.
- There is an easy reduction from MPCP to full PCP (details in book).

2 Configurations and Computations of a TM

- We begin by formally defining the configurations of a TM.
- A (single-tape) TM is in configuration $\alpha q \beta$ if
  - The contents of its tape, up to the last non-blank cell, are exactly the string $\alpha\beta$
  - The head is over the first symbol of β
  - The finite control of the TM is in state q
- An *initial configuration* of a TM is one that reflects the starting condition of a TM, i.e. $q\gamma$ for some input string γ.
- Let c and c' be two configurations of a TM M.
- If M can transition from configuration c to configuration c' in a single move, we say that $c \vdash c'$ (c "entails" c').
- Note that if $c \vdash c'$, then the position of the state in c' can move by at most one character compared to its position in c.
- Moreover, only one character of $\alpha\beta$ can change as the result of a move.
- A *computation* of a TM is a sequence of configurations $C = c_1 \ldots c_n$ such that
  - c_1 is an initial configuration
  - $c_i \vdash c_{i+1}$ for all i
- If c_n is of the form $\alpha h_a \beta$, where h_a is the accepting state of the TM, then C is an *accepting computation.*

3 Encoding TM Computations with MPCP

- We will use MPCP to encode computations of a TM M.
- In particular, given M and w, we will build a set of MPCP tiles that has a solution iff there exists an accepting computation for M on w.
- Hence, ACC reduces to MPCP.

Let's build some tiles.

- Our "starting" tile is $[\varepsilon|q_0w\#]$. It is empty on top and has the starting configuration of M on w on the bottom.
- (We use # to seperate configurations.)
- The goal of the remaining tiles is to let the top configuration "catch up" to the bottom.
- For every symbol $a \in \Gamma$, add a tile $[a|a]$.
- Also add a tile $[\#|\#]$
- Now, for each legal move of the TM of the form

$$\delta(q, a) = (p, b, R)$$

  add a tile $[qa|bp]$.
- For each legal move of the form

$$\delta(q, a) = (p, b, S)$$

  add a tile $[qa|pb]$.
- Finally, for each legal move of the form

$$\delta(q, a) = (p, b, L)$$

  and each $d \in \Gamma$, add a tile $[dqa|pdb]$.

What can we do so far?

- Suppose we have a legal computation of the form $c_1 ... c_n$.
- I claim we can build up a sequence of tiles whose top forms the string $c_1\#c_2\# \ldots \#c_{n-1}\#$, and whose bottom forms the string $c_1\#c_2\# \ldots \#c_n\#$.
- Starting tile gives us $c_1\#$ on the bottom and nothing on top.
- To match this initial string, we need to string together tiles that form $c_1\#$ on top.
- For all of c_1 except the area around the head, we can do this one character at a time.

- For the vicinity of the head, the only available tile that has the state and head context of c_1 on top has the corresponding state and head context for c_2 on the bottom, where $c_1 \vdash c_2$.
- Hence, by matching $c_1\#$ on top, we create the successor configuration $c_2\#$ on the bottom.
- This observation extends to computations of any length.
- Conversely, any sequence of tiles that does not contain a top-bottom mismatch must describe a succession of legal configurations on top, and the same set of configurations, plus one more, on the bottom.
- **Example:**

Great, but how do we finish?

- A legal computation corresponds to a pair of matching strings with an "overhang" of one configuration on the bottom.
- We want to add tiles to let us fill in the missing config on top while not extending the bottom.
- Add tiles of the form $[h_a a \mid h_a]$ and $[a h_a \mid h_a]$ for each $a \in \Gamma$, as well as $[h_a \mid \varepsilon]$.
- If a computation ends with an accepting configuration $\alpha h_a \beta$ on the bottom, we can use our single-character tiles to match α, then match h_a and the first char of β, top to just h_a on the bottom, then use single-char tiles to match the rest of β.
- This results in a slightly smaller bottom overhang, which is the final configuration without the first char of β.
- Repeat the above until we have consumed all of β, and the bottom string ends with some αh_a.
- Now do the same thing with tiles of the form $[a h_a \mid a]$ to consume all of α, one char at a time.
- Finally, use $[h_a \mid \varepsilon]$ to match h_a on the bottom. the two strings have now caught up to each other.

- **Example:**

- Note that we cannot do this unless the bottom computation is accepting, i.e. ends with state h_a.
- Conversely, can show that there is no way to finish *unless* the bottom reaches an accepting configuration, since no tiles other than those involving h_a let the top catch up to the bottom.

Conclude that there is a way to form a set of tiles with matching tops and bottoms iff there exists an accepting computation of M on string w.

4 PCP Fun Facts

PCP is a convenient source for proving lots of other things undecidable. Examples of undecidable problems from random Google search:

- Validating an XML document against a DTD or schema with foreign keys
- Intersection of a line with attractor of an IFS (iterated function system, from theory of fractal geometry)
- Determining whether two pointers in a C program can ever alias each other, i.e. can refer to same memory location

There are many restricted versions of PCP that are of interest.

- PCP is decidable if input contains only two tiles.
- However, a result by Matiyasevich and Senizergues from 1996 shows that PCP is undecidable for inputs with at least seven tiles.
- (I don't know of any result for 3-6 tiles.)
- PCP is undecidable if the alphabet size is at least 2.
- PCP is decidable if all tile strings are constructed from an alphabet of size 1 (i.e. a single character) – homework.