Computation Histories and PCP
(not Probabilistically Checkable Proofs)
(also not Angel Dust)

CS 81: Computability and Logic
November 23, 2010
Computation History

✓ Step-by-step recording of a TM computation.
✓ Used to show more problems not decidable.
We can describe the configuration of any TM using a string $C = \lambda q \in \Gamma^*$. 

$x \in \Gamma^*$ = symbols to the left of head  
$q \in Q$ = current control state  
$y \in \Gamma^*$ = symbols under and to the right of head

Over the course of a computation, we have  

$\cdots \Rightarrow x_1 q_1 y_1 \Rightarrow x_2 q_2 y_2 \Rightarrow x_3 q_3 y_3 \Rightarrow \cdots$

If the TM halts, we can represent the whole history of the computation as a single (finite) string!

Traditionally written $\#C_1 \#C_2 \#C_3 \# \cdots \#C_n \#$
Checking a History

✓ Checker: a Turing Machine \( C \) that, given \(<M, h>\), checks whether \( h \) is a history of TM \( M \).

✓ Consecutive states should be equal, except around the head (where the change corresponds to the transition table of \( M \)).

✓ Can check whether \( h \) is a halting (or an accepting) history by looking at the last control state.
Digression: LBAs

In fact, the CH can be checked for validity by a less-than-general TM called an LBA.

LBA = “Linear Bounded Automaton,” a TM that can only use the part of the tape containing input.

An LBA can have a large tape alphabet and can “mark” tape cells. It just can’t grow its tape.

More powerful than a DFA

- Number of potential states grows with input size
- DFA wouldn’t be able to check a computation history.
Accepting for LBAs

\[ A_{LBA} = \{ \langle M, w \rangle \mid M \text{ a LBA accepting } w \} \text{ is decidable.} \]

Proof: When running \( M \) on \( w \), there are at most

\[ n := |Q| \times |\Gamma|^{|w|} \times |w| \]

distinct “states” during the computation.

So,

Run \( M \) on \( w \).

If computation takes longer than \( n \) steps, it's in an infinite loop; \( M \) doesn't accept \( w \).
Emptiness for LBAs

$E_{LBA} = \{ <M> \mid M \text{ a LBA, } L(M) = \emptyset \}$ isn’t decidable.

Proof: $A_{TM} \leq E_{LBA}$. 
All_{CFG} = \{ <G> \mid G \text{ a CFG, } L(G) = \Sigma^* \} \text{ is undecidable.}

**Proof:** $A_{TM} \leq A_{CFG}$.

- Key: given $<M,w>$ create a PDA/CFG for strings that aren’t accepting computation histories!
  - PDA accepts strings that
    - Don’t start with $q_0w$
    - Or, don’t end with $xq_{accept}y$
    - Or, two successive configurations don’t match properly
  - Hack: need to reverse every other configuration.
  - The grammar for this PDA is $\Sigma^*$ iff $M,w$ has no finite, accepting history.
EQ_{CFG}

Eq_{CFG} = \{ <G_1, G_2> \mid L(G_1) = L(G_2) \} is undecidable.

Proof: All_{CFG} \leq EQ_{CFG}. 
Post Correspondence Problem
Emil Post

Named after logician Emil Post (1897-1954)

- studied fundamental models of computation
- “scooped” by Gödel, Turing, and Church
Why PCP?

- Trivial problem to state
  - Looks nothing like Turing Machines
  - A child can understand it
  - Superficially, doesn’t look that hard

- Can reduce PCP to other problems, showing them undecidable
PCP
(not a Probabilistically Checkable Proofs)
(not Angel Dust)

Given a set of “dominos” (pairs of strings), find a sequence of dominos so that the top strings and the bottom strings match.

Theorem: PCP is undecidable.
Proof: Halting $\leq$ MPCP $\leq$ PCP
Modified PCP (MPCP)

Like PCP, but solution starts with first domino.
The following MPCP instance has no solution.
MPCP $\leq$ PCP

Given an instance of MPCP, solve by translating dominos and doing PCP.
Halts $\leq$ MPCP

- Idea: if a TM halts, it has a computation history
- Given a $<M,w>$, construct dominos whose solution would yield such a history (on top and bottom); use MPCP-solver to check for a solution.
The Dominos

✔ First domino: set up the initial state

![Diagram showing initial state with symbols: #, q0w#]

✔ Helper dominos: copy unchanged parts of the tape from C_i to C_{i+1}. (optional: expand the tape)

![Diagram showing helper dominos with symbols: a, b, c, _, #, _#]
More Dominos

For each transition  \( q,x \rightarrow q', y, R \)

\[
\begin{array}{c}
\text{previous config} \\
qx \\
yq' \\
\end{array}
\]

For each transition  \( q,x \rightarrow q', y, L \)

\[
\begin{array}{c}
aqx \\
q'ay \\
\end{array}
\quad \quad \quad
\begin{array}{c}
bqx \\
q'by \\
\end{array}
\]

\[\ldots\]
Completion Dominos

✓ Technical “Trick”

✓ When we reach a halting state $h$, we delete the configuration (one symbol at a time) until it disappears.

✓ For each halting state $h$, and each symbol $x$, we need

\[
\begin{array}{c}
hx \\
h \\
\h \# \\
\h \#
\end{array}
\begin{array}{c}
xh \\
h \\
\h \# \\
\h \#
\end{array}
\]

✓ Finally,
Example: $w=11$