Computation Histories

Robert M. Keller
Harvey Mudd College
April 2010

What is this?
• A computation history (CH) is a recording of the history of a Turing-machine computation.

Of what use is it?
• Using CH’s we can demonstrate that various problems are undecidable.
• These problems would be difficult to show using, say, mapping reduction.

State of a Turing Machine
• The total state (often called a "configuration" to distinguish it from the control state) represents everything that needs to be known about the machine to continue the computation.

Capturing States as Strings
• We have seen this before:
• A state can be represented as a string $xqy \in \Gamma^*$
  where
  • $\Gamma$ is the tape alphabet
  • $x$ is the tape to the left of the head
  • $q$ is the control state
  • $y$ is the tape under and to the right of the head

Computation History
• As the TM computes, the states make transitions:
  \[ x_1q_1y_1 \Rightarrow x_2q_2y_2 \Rightarrow x_3q_3y_3 \Rightarrow \ldots \]
• A deterministic machine will eventually either:
  • **Halt**: Reach a state $xq_0y$ where no further transition is defined, or
  • **Diverge**: Never reach such a state, which means it goes on forever.
Encoding An Entire History

- **In the halting case**, the entire history is encodable as a single string:
  \[ x_1q_1y_1 \Rightarrow x_2q_2y_2 \Rightarrow x_3q_3y_3 \Rightarrow \ldots \Rightarrow x_hq_hy_h \]
  where we consider \( \Rightarrow \) to be a symbol in a larger alphabet.

- Usually \# is used in place of \( \Rightarrow \), maybe because it's easier to type.

Checking a History with a TM

- There is a Turing machine C (checker) that, with input \(<M, h>\), where \(<M>\) is an encoding of an arbitrary TM, can check whether \(h\) is a history of \(M\).

- All \(C\) needs to do is see if the symbols around the head of each state change correspond to the transition table of \(M\).

- Further more, \(C\) can check whether \(h\) is a halting (or an accepting) history by looking at the last control state.

Checking a History with a TM

- \(x_1a_1q_1b_1y_1 \Rightarrow x_2a_2q_2b_2y_2 \Rightarrow x_3a_3q_3b_3y_3 \Rightarrow \ldots \)
  where each \(a_i, b_i \in \Gamma\)

  Do the local transitions \(a_iq_ib_i \rightarrow a_{i+1}q_{i+1}b_{i+1}\) correspond to the rules in the description?

  Do the other parts of the string match?

  A TM can check these things.

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 never exceed the amount of tape on which the input is written.

- An LBA can have a tape alphabet that is bigger than the input alphabet, so it can effectively "mark" tape cells, etc. It just can't grow its tape.

LBA vs. DFA

- At first glance, it might appear that an LBA is no more powerful than a DFA, since it cannot add new states.

- The difference is, however, that an LBA’s total state set is a (linear) **function of the input string size**, which is not true for a DFA.

An Aside: 2-way DFAs

- If the LBA never writes on its tape, then it becomes a 2-way DFA.

- In this case, the power is reduced to that of a DFA: 2-way DFA’s only accept regular languages.

- The proof of this is non-trivial, and surprising. It involves the Myhill-Nerode theorem.
LBA for CH checking

- It is easy to see that an LBA can check a CH for correctness.
- It is also easy to see that no DFA could do this. It would require "matching" arbitrarily-long sub-strings. The Myhill-Nerode theorem or the pumping lemma could be used to prove this.

A\textsubscript{LBA} is the Acceptance Language for LBAs

- Define \( A\textsubscript{LBA} = \{<M, w> | M \text{ is an LBA} \land M \text{ accepts } w\} \)

Theorem: \( A\textsubscript{LBA} \) is decidable.

- Proof: Given an LBA encoding with tape \( <M, w> \), we can simulate \( M \) on \( w \).
  - The maximum number of distinct tape states is \( n^{|w|} \), where \( n \) is tape alphabet size and \( |w| \) means the length of \( w \).
  - The number of different head positions is \( 1 + |w| \).
  - The number of control states is \( m \), say.
  - So the total number of different states of \( M \) for an input \( w \) is \( mn^{|w|}(1 + |w|) \).
(continued)

Proof that \( A\textsubscript{LBA} \) is decidable (cont’d)

- For each step in the simulation of \( M \) on \( w \), we increment the count, having started at 0.
- If \( M \) on \( w \) is still computing after \( mn^{|w|}(1 + |w|) \) steps, we know that it is in a loop. So the simulating machine can reject \( <M, w> \) if this happens.
- The simulating machine will otherwise accept or reject \( <M, w> \), depending on what happens with \( M \) on \( w \).
- Hence there is a TM that can decide \( A\textsubscript{LBA} \).

Theorem: \( E\textsubscript{LBA} \) is undecidable.

- Define the Emptiness Language \( E\textsubscript{LBA} = \{<M> | M \text{ is an LBA} \land \text{L}(M) = \emptyset\} \)
- Proof: We show \( A\textsubscript{TM} \leq_m E\textsubscript{LBA} \).
  Define \( f(<M, w>) = M' \), where \( M' \) is an LBA that accepts the accepting computation histories for \( M(w) \). Then \( \text{L}(M') \neq \emptyset \) iff \( <M, w> \in A\textsubscript{TM} \).

ALL\textsubscript{CFG} is undecidable

- Define \( \text{ALL}_{\text{CFG}} = \{<G> | G \text{ is a CFG} \land \text{L}(G) = \Sigma^*\} \)

We know that for every CFG, there is a corresponding PDA that accepts the same language.

We show that PDAs can also, in a sense, check computation histories.

The halting computation of a TM will be identified with the absence of a string accepted by the PDA.
Proof of \(\text{ALL}_{\text{CFG}}\) is undecidable

- For any TM and input \(<M, w>\) we construct a PDA that accepts all strings that are not valid computation histories.

- Thus, if the PDA accepts all strings, then \(M\) does not halt on \(w\).

Why reverse?

- Reversing every other state enables two successive states to be checked by a PDA.

- The PDA, given an input, checks one of these (non-deterministically):
  - Does the input not begin with the initial state of \(<M, w>\)? If not, accept.
  - Does the input not end with an accepting state of \(<M, w>\)? If not, accept.
  - Starting at any one of the sections \(C_i\#C_{i+1}\) or \(\overline{C_i}\#C_{i+1}\), do the two sections not represent consecutive states. If not, accept.

- So a string is accepted iff it does not represent a valid computation history.

- Thus the PDA accepts \(\Sigma^*\) iff \(M(w)\) does not have an accepting computation history.

Corollary: \(\text{EQ}_{\text{CFG}}\) is undecidable

- Define \(\text{EQ}_{\text{CFG}} = \{<G, H> | G \text{ and } H \text{ are CFG's } \land L(G) = L(H)\}\)

Proof: Show \(\text{ALL}_{\text{CFG}} \leq_m \text{EQ}_{\text{CFG}}\).