More Reducibility

CS154
Chris Pollett
Apr 24, 2006.

Outline

- Midterm II
- Reductions via Computation Histories
- Post Correspondence Problem

Midterm II

- Since the midterm scores were low, I have decided to give students an opportunity to try to recoup some of the points they lost.
- To get points back, on separate sheets of paper correctly work out each of the problems from the midterm.
- When you have done this, come to my office hours, with your original test, together with the corrected solutions.
- I will on the spot make up some questions about your corrected solutions.
- Depending on how well these are answered, I will give up to half credit back on all the points you lost on the midterm.
- For instance, if your score was 6 and you do the above and correctly answer my question then you could get $(15-6)/2 = 9/2 = 4.5$ points added to your score.
- To facilitate everyone getting the opportunity to get points back I will have extra office hours, this Friday, April 28 and Friday, May 5 from 1-3pm.
- May 5 is the last day to try to recoup these points.

Reductions via Computation Histories

- Consider the following language:

  $R := \{\langle w,M, x\rangle \mid w$ is the code of a sequence of configurations, each configuration yielding the next according to the transition table of TM M on input x. Further, the last configuration is accepting.$\}$

- This language is decidable.
- Notice $A_{TM} := \{\langle M, x\rangle \mid \exists w \, \langle w,M, x\rangle \in R \}$.
- The string w in the above can be viewed as a computation history.
- Such histories are often useful in doing reductions from one problem to another.

Formal Definition of a Computation History.

- Let M be a TM and x an input string.
- An **accepting computation history** for M on x is a sequence of configurations $C_1,.., C_k$, where C_1 is the start configuration of M on x, C_k is an accepting configuration of M, and each C_i legally follows from C_{i-1} according to the rules of M.
- A **rejecting computation history** for M is defined similarly, except that C_k is a rejecting configuration.

Linear Bounded Automata

- We will next work towards using Computation Histories to give undecidability proofs.
- Our first example will involve a new machine model which has strength between a PDA and a TM.
- A **linear bounded automata (LBA)** is a restricted type of TM wherein the tape head isn't permitted to move off the portion of the tape containing the input.
- If an LBA tries to move off this part of the tape to the right, the tape head stays where it is.

Strength of LBAs

- One can verify that each of the TMs we gave for the languages A_{DFA}, A_{CFG}, E_{DFA}, and E_{CFG} are either LBAs or easily modified into LBAs.
- For example, E_{CFG} involved marking each terminal, then marking a variable A if it appear in a A--> $B_1 \ldots B_n$ and the B_i's had already been marked. Finally, one checks if the start variable has been marked.
- This marking can be done without using any more tape squares so the above can be done by an LBA.

A Useful Lemma about LBAs

Lemma. Let M be an LBA with q states and g symbols in the tape alphabet. There are exactly qng^n distinct configurations of M for a tape of length n.

Proof. A configuration consists of the state of the control of the LBA, the position of the tape head, and the contents of the tape. So there are q possibilities for the state, the head can be in one of at most n positions, each of the n tapes squares could have one of g symbols written in it (so g^n possibilities). All together this gives, qng^n.

Decidability and LBAs

Theorem. A_{LBA} is decidable.

Proof. The algorithm that decides A_{LBA} is as follows:

L=“On input <M, w>, where M is an LBA and w is a string:

1. Simulate M on w for qng^n steps or until it halts.
2. If M has halted, *accept* if it accepted; and *reject* if it rejected. If it has not halted *reject*.”

LBAs and Undecidability

- In contrast to the last theorem above, not all problems about LBAs are decidable:

Theorem E_{LBA} is undecidable.

Proof. The reduction is from A_{TM}. We show if E_{LBA} is decidable then A_{TM} also would be decidable. Let L={w | w is a string of the form $C_1\#C_2..\#C_k$ given a legal accepting computation history of M on input x}. One can show that L can be recognized by an LBA; let's call it B. Further, if L is empty, <*M, x*> is not in A_{TM}. So if E_{LBA} were decidable the following would be a decision procedure for A_{TM}:

S= "On input <*M,w*>, where *M* is a TM and w is a string:

1. Construct LBA *B* from *M* on *w* as described in the proof idea.
2. Run *R* on input <*B*>.
3. If *R* rejects, *accept*; if R accepts, *reject*."