CS120: Intro. to Algorithms and their Limitations Prof. Salil Vadhan

Problem Set 9

Harvard SEAS - Fall 2021 *Due: Wed Nov. 17, 2021 (5pm)*

Your name:
Collaborators:
No. of late days used on previous psets:
No. of late days used after including this pset:

The purpose of this problem set is to solidify your understanding of the Resolution algorithm, practice reductions to and from SAT problems, and develop some intuition for what kinds of problems about programs are computationally solvable.

1. (Resolution) Use the algorithm `ResolutionInOrder` that we saw in Lecture 18 to decide the satisfiability of the following formulas, and use the algorithm `ExtractAssignment` to obtain a satisfying assignment for the ones that are satisfiable. (Please make sure to follow both algorithms *exactly*, including the order in which the clauses are processed. A correct final solution that does not show all of the intermediate steps of both algorithms will not receive full score.)

   (a) $\varphi(x_0, x_1, x_2, x_3) = (x_2 \vee \neg x_1) \wedge (x_3 \vee x_1) \wedge (x_0 \vee x_1) \wedge (\neg x_3) \wedge (\neg x_1 \vee \neg x_2)$.

   (b) $\varphi(x_0, x_1, x_2, x_3) = (x_3 \vee x_0 \vee x_2) \wedge (\neg x_3 \vee \neg x_2) \wedge (x_2 \vee x_0) \wedge (x_1 \vee x_0)$.

   (c) $\varphi(x_0, x_1, x_2, x_3) = (x_0 \vee x_1) \wedge (\neg x_0 \vee x_1) \wedge (x_1 \vee x_3 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2)$.

2. (Faster 2-SAT Algorithm) In class, we saw how Resolution can solve 2-SAT in time $O(n^3)$. Here we will obtain a faster algorithm by reduction to reachability in directed graphs. Note that a 2-clause $(\ell \vee \ell')$, where ℓ and ℓ' are literals, is equivalent to each of the implications $\neg\ell \to \ell'$ and $\neg\ell' \to \ell$. Thus, given a 2-CNF formula φ on variables $x_0, \ldots, x_{n-1}$, it is natural to construct a directed graph G whose vertices are the $2n$ literals and where we include the directed edges $(\neg\ell, \ell')$ and $(\neg\ell', \ell)$ for every clause $(\ell \vee \ell')$ in φ. Throughout this problem, view any 1-clause (ℓ) as the 2-clause $(\ell \vee \ell)$, yielding the directed edge $(\neg\ell, \ell)$ in G.

   (a) Show that if there is a path from literal $\neg\ell$ to literal ℓ' in the graph G, then resolution applied to φ will (eventually) derive the clause $(\ell \vee \ell')$.

   (b) Conversely, show that if resolution applied to φ (eventually) derives a clause $(\ell \vee \ell')$, then there is a path from $\neg\ell$ to ℓ' in the graph G. (Hint: use induction on the number of applications of the resolution rule.)

   (c) Using the previous parts, reduce 2-SAT to reachability in directed graphs and obtain an $O(mn)$-time algorithm for *deciding* 2-SAT instances with n variables and m clauses. (Hint: the only way to derive the empty clause is to resolve two 1-clauses of the form (x) and $(\neg x)$.)

   (d) (challenge*) Give an $O(mn)$-time algorithm for solving the search version of 2-SAT (i.e., if the input formula φ is satisfiable, produce a satisfying assignment).

3. (Solvable problems about general programs) Show that, in contrast to the Halting Problem, there *is* an algorithm for solving each of the following computational problems. In all cases, you should describe and justify your algorithm for RAM programs P, and then briefly mention what modifications, if any, are needed to handle Word-RAM programs. You should describe your algorithms in pseudocode and briefly justify correctness. When P is a Word-RAM programs, you may assume that P's arithmetic never overflows and that it does not access memory out of bounds.

   (a) (Syntactic properties) Given a RAM (or Word-RAM) Program $P = (V, C_0, \ldots, C_{\ell-1})$, find the largest subset $S \subseteq [\ell]$ such that for every $i \neq j \in S$, commands C_i and C_j do not have any variables in common.

   (b) (Bounded halting) Given a RAM (or Word-RAM) Program P and an input x of length n, decide whether or not P halts within $3n^2 + 2n$ steps on x.

   (c) (Enforcing runtime) Given a RAM (or Word-RAM) Program P, construct a new RAM (resp., Word-RAM) program P' such that P' has running time $O(n^2)$ and if P has running time at most $3n^2 + 2n$ on inputs of length n, then $P'(x) = P(x)$ for all x. (Hint: insert a few lines before every line of P to keep track of and check a time counter.)