CSE681, Spr’12  Take-Home Final  Due Tue. 5/15

Otherwise known as Problem Set 6

Three pages, four problems, 228 pts., some open-ended content but not intended to be front-and-center. Policy on Net-use is no *unless expressly allowed*, as on 4(d); if you happen to have seen something like 1(c) previously, fine *but let me know*—and otherwise please work it out as best you can.

(1) Let us allow quantifiers of the form $(\exists u_1, \ldots, u_k)$ where $k$ is a parameter different from the input length $n$. If $k$ were really fixed then this would need no special privilege, and we could rewrite expressions like $\wedge_{j=1}^k u_j$ longhand as $u_1 \wedge u_2 \wedge \ldots \wedge u_k$. However, we will treat $k$ as “quasi-fixed,” so the distinction is meaningful. Call the resulting system $\text{FOFP}$ for first-order fixed-parameter. When only one such quantifier is allowed, and it is existential out front, call the system $\text{FOFP}_\exists$.

For example, the following is an $\text{FOFP}_\exists$ formula defining the language $VC_k$ of graphs with a vertex cover of size (at most) $k$—and the same formula works for any particular $k$.

$$(\exists u_1, \ldots, u_k)(\forall v, w)[E(v, w) \rightarrow \forall_{j=1}^k (v = u_j \vee w = u_j)].$$

This says that there is a set $S$ of $k$ (or fewer) nodes such that every edge is incident to a node in $S$. Note that we are not regarding this as a full-fledged second-order quantifier over the subset $S$, because we are limiting by the size parameter $k$.

(a) Write $\text{FOFP}_\exists$ formulas for the following languages. For graphs you may use the edge relation, while for Boolean formulas in conjunctive normal form (CNF) you may quantify separately over variables $v$ (with signs $b \in \{0, 1\}$) and clauses $C$, and reference the predicate $\text{In}(v, b, C) \equiv \text{“variable } v \text{ appears with sign } b \text{ in clause } C.”$

(i) The language $I_k$ of undirected graphs with an independent set of size (at least) $k$.

(ii) The language $D_k$ of undirected graphs with a dominating set of size (at most) $k$, meaning a set $D \subset V$, $|D| = k$, such that every vertex outside of $D$ has a neighbor in $D$.

(iii) The language $S_k$ of Boolean formulas in CNF that can be satisfied by fixing the value (true or false) of just $k$ of the $n$ variables.

(iv) Same as (iii) except the formula must be in 3CNF.

(b) Does the depth of the $\text{AC}^0$ circuits you get for these languages depend on $k$? How about the exponent of the polynomial bounding their size?
(c) Give an algorithm for deciding $VC_k$ that runs in time $O(2^k n^2)$—on the kind of random-access machine typically used in an intro algorithms course. Hint: Take fixed orderings of both the vertices and the edges of the graph, and cycle through strings $b$ of length $k$, using the next bit of $b$ to make choices as-needed.

(d) Give an algorithm for deciding $I_k$ that runs in time $O(f(k)n^a)$ where we don’t care what the dependence $f(k)$ on $k$ is, but importantly $a < k$. Hint: Recall the case $k = 3$ from homework—can you build on it?

Whether the exponent $a$ of $n$ can be fixed to be independent of $k$ like in part (c) is a major open problem, so don’t think that is what the question demands—but it motivates the remaining two parts, along with the question of how many pure-$FO$ quantifiers one needs after the initial existential $k$-dependent one.

(e) Does the standard reduction from (3)SAT to Independent Set (as shown in lecture and notes) reduce each $S_k$ to $I_k$, or even to $I_{k'}$ for some fixed $k' > k$ independent of $n$?

(f) Do a reduction from (3)SAT to Dominating Set that also reduces each language $S_k$ to some $D_{k'}$ where $k' \geq k$. Can you get $k' = k$?

For a final footnote, part (f) becomes much harder if $S_k$ is replaced by $S_k' = \text{the language of Boolean formulas with a satisfying assignment in which (at most) } k \text{ variables are set to true.}$ Overall this is another case where the boundary between $FO$ and $AC^0$ on one hand, and NP-complete on the other, seems tantalizingly fine.

Points are $4 \times 6 = 24$ on (a), $6 + 6 = 12$ on (b), $18$ on (c), $18$ on (d), $6$ on (e), and $18$ on (f), for 96 total.

(2) Sketch the design of $AC^0$ circuits for adding $(\log n)^{O(1)}$-many $n$-bit binary numbers. You may assume that you already have $AC^0$-circuits that add $k$-many $k$-bit numbers, where $k = \lceil \log_2 n \rceil$, as was sketched in lecture, you need not sketch circuits you’ll use “at the end” for adding two $n$-bit numbers, and you may handwave the $DLOGTIME$ uniformity.

Deduce from this that for any fixed $k$, the language $W_k$ of binary strings with at most $(\log n)^k 1$’s in them belongs to $DLOGTIME$-uniform $AC^0$, i.e., to $FO$. Thus $AC^0$ can simulate threshold gates with polylog thresholds, giving it a modicum of ability to count. $(30 + 6 = 36$ pts.)

(3) Prove, however, that the $(\log n)^k$ in problem (2) cannot be improved to $n^\epsilon$, no matter how small the fixed $\epsilon > 0$ is. (It is AOK to use previouslyproved results from lecture or homework or notes. 24 pts.)
(4) Let us revisit the function \( f \) on strings over alphabet \( T = \{0, 1, 2\} \) that moves all 2’s flush-right while leaving the order of the 0’s and 1’s the same, for instance: \( f(20121202) = 01102222 \), \( f(0010) = 0010 \), \( f(\lambda) = \lambda \), \( f(00100102) = 00100102 \). Note that this is the stable topological sort of the partial order where \( 2 > 0, 1 \) with 0, 1 unrelated to each other.

We desire circuits \( C_n \) with \( n \) input gates and \( n \) output gates that compute this function, where the gates may use ternary logic—i.e., wires may hold 0, 1, 2 as values. The problem is essentially unchanged if you re-code 0 by 00, 1 by 11, and 2 by 01, using \( 2n \)-many inputs and outputs and ordinary Boolean logic.

(a) Show by direct reduction that if \( f \) were in \( AC^0 \) then \( \text{Parity} \) would be in \( AC^0 \).

(b) Show that \( f \) is in \( TC^0 \). \textit{Hint:} Convert the stable sort of 0, 1 < 2 into an ordinary sort of a bigger set, and then convert it into a bunch of separate counting problems—note that we’re not caring yet whether the circuits have size that is quadratic, cubic, whatever.

(c) Now show that \( f \) is \( TC^0 \)-complete, by reducing \( \text{Majority} \) to it.

(d) Show that \( f \) has quasi-linear-size circuits of depth \( O(\log^2 n) \). Quasi-linear means \( n \) times a polynomial in \( \log n \). (For this problem it is OK if you look up “Batcher’s bitonic sort,” though there are more-elementary ways to do it. Open-ended extra credit if you achieve depth \( O(\log n) \)—the only way I know to do this uses the \textit{Ajtai-Komlos-Szemeredi sorting network} and even then isn’t easy.)

(e) Now we address the question of whether \( f \) has linear-size circuits, of any depth. Show that it suffices to create linear-size circuits that work only for \( n \) that are a power of 2, and work only when it happens that exactly \( n/2 \) of the entries are 2’s. (This is open—indeed \( f \) is to my mind the simplest function that I don’t know to have linear-size circuits.)

Points are \( 9 + 18 + 9 + 18 + 18 = 72 \) points, for 228 points total.