CPSC506 Complexity of Computation Term 2, 2020

Homework # 2

Due Wed Feb 12.

1. Recall that the recursive algorithm Reach(x, y, i) tests, for given graph $G = (V, E)$, whether there is a path in G from node x to node y of length at most 2^i.

   The base case of the algorithm, when $i = 0$, simply tests whether $x = y$ or whether (x, y) is an edge of G.

   Assume that x and y are binary strings, represented with k $(= \lceil \log |V| \rceil)$ bits.

   Describe how to construct a fan-in 2 Boolean circuit with AND, NOT and OR gates that has $2k$ input bits (k bits corresponding to x and k corresponding to y), and outputs true if and only if this base case test is true. Your circuit should have depth $O(\log |V| + \log |E|)$. (The circuit will depend on G; a description of G is not an input to the circuit.)

2. Show one of the following:

   (a) 2SAT is complete for NL, with respect to log space ($\leq_{log}$) reductions.

   The 2SAT problem is: given a Boolean formula that is the conjunction (AND) of clauses, where each clause is the disjunction (OR) of at most two literals (where a literal is either x_i or $\bar{x}_i$, for some variable x_i), is the formula satisfiable?

   (b) The Emptiness Problem for Intersection of Finite Automata (EIFA) is PSPACE-complete, with respect to polynomial time ($\leq_p$) reductions.

   The EIFA problem is: given a list of deterministic finite state automata $A_1, A_2, \ldots A_k$, is the language $L(A_1) \cap L(A_2) \cap \ldots \cap L(A_k)$ empty?

3. The following construction will be useful later in the semester: Describe a polynomial time deterministic algorithm that, given a quantified boolean formula ϕ with no free variables, outputs a quantified boolean formula ϕ' in prenex normal form with negations only over variables, such that ϕ is valid if and only if ϕ' is.

   Formally, we inductively define a quantified Boolean formula (qbf) ϕ over variable set X, and its associated sets F_ϕ and B_ϕ of free and bound variables respectively, as follows. For every $x \in X$, x is a quantified Boolean formula with $F_x = \{x\}$ and $B_x = \emptyset$. Also, if ϕ and ϕ' are quantified Boolean formulas such that $F_\phi \cap B_{\phi'} = \emptyset$ and $F_{\phi'} \cap B_\phi = \emptyset$ then

   - $\bar{\phi}$ is a quantified Boolean formula with free variable set F_ϕ and bound variable set B_ϕ;
   - $(\phi \vee \phi')$ and $(\phi \wedge \phi')$ are quantified Boolean formulas with free variable set $F_\phi \cup F_{\phi'}$ and bound variable set $B_\phi \cup B_{\phi'}$, and
   - if $x \notin B_\phi$ then $(\exists x \phi)$ and $(\forall x \phi)$ are quantified Boolean formulas with free variable set $F_\phi - \{x\}$ and bound literal set $B_\phi \cup \{x\}$.

(Sometimes when there is no ambiguity, parentheses are omitted.)

For example,

$$\phi = \forall x(\exists y((x \vee y) \wedge \forall z((x \wedge z) \vee (y \wedge \bar{z}))) \vee \exists w(z \vee (y \wedge \bar{w}))), \text{ and}$$
$$\phi' = \forall x((x \vee y) \wedge \forall z(((x \wedge z) \vee (y \wedge \bar{z})) \vee \forall w(z \vee (x \wedge \bar{w})))).$$

are quantified Boolean formulas; in ϕ all variables are bound, whereas in ϕ', y is free.

If all variables of a quantified Boolean formula ϕ are bound, then ϕ's truth value can be defined inductively in a natural way. We say that ϕ is *valid* if its value is true. Also we can define what is the quantifier to which a particular variable is bound.

We say that ϕ is in *prenex normal form* if ϕ is of the form $(Q_1 x_1(Q_2 x_2 \ldots (Q_n x_n \phi') \ldots))$, where each Q_i is either $\exists$ or $\forall$, and ϕ' does not have quantifiers. (If parentheses are omitted, ϕ is of the form $Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \phi'$.) We say that ϕ has *negations only over variables* if any expression of the form $\bar{\rho}$ of ϕ is such that ρ is a variable.

4. (More on quantified boolean formulas.) If a variable x occurs in qbf $(Q_y \rho)$, where $Q \in \{\exists, \forall\}$, then we say that x *is in the scope* of Q. We say that a qbf ϕ is *simple* if any variable x of ϕ is in the scope of at most one $\forall$ quantifier that is within the scope of the quantifier of ϕ to which x is bound. For example, the quantified formula ϕ given in problem 3 is simple, but the formula ϕ' is not.

   Describe a log space deterministic algorithm that, given as input a quantified Boolean formula ϕ with no free variables and negations only over variables, outputs a simple quantified Boolean formula ϕ' which also has negations only over variables and such that ϕ is valid if and only if ϕ' is.