Computer Science 2401F
St. George Campus

October 29, 2002
University of Toronto

Homework Assignment #2

Due: November 12, 2002, by 1:10PM

We need some definitions for this assignment.

For Questions 1 and 2, we define a *monotone* circuit to be a circuit whose only gates are $\wedge$ and $\vee$; in particular, we do not permit $\neg$ gates.

For Question 4 we define the language 3DM, or *3-dimensional matching*. Recall that the 2-dimensional matching language consists of those bipartite graphs that contain a perfect matching; this language is in **P**. 3DM consists of those *tri*partite graphs that contain a perfect matching, where in this case an "edge" consists of a triple of vertices; this language turns out to be *NP*-complete.

More rigorously, we define 3DM as follows. An instance is a pair (V, E) where:
V is the union of 3 (pairwise) disjoint sets, $V = A \cup B \cup C$ where $A = \{a_1, a_2, \ldots, a_n\}$, $B = \{b_1, b_2, \ldots, b_n\}$, $C = \{c_1, c_2, \ldots, c_n\}$; $E \subseteq A \times B \times C$.
3DM is the set of such instances (V, E) such that there is a set $M \subseteq E$, $|M| = n$, such that every member of V occurs in exactly one edge from E.

Question 1. (10 marks) Define MCVP, the *monotone circuit value problem*, to be the language consisting of (encodings of) pairs $(C, \overline{a})$ such that C is a monotone circuit that accepts input $\overline{a}$.

Prove that MCVP is complete for **P**.

Question 2. (10 marks) Define L to be the set of (encodings of) pairs (C_1, C_2) such that C_1 and C_2 are *monotone* circuits on the same number of inputs, such that there exists some input that is accepted by C_1 but not by C_2.

Prove that L is complete for **NP**.

Question 3. (15 marks) Define a *horn clause* (of the propositional calculus) to be a clause (that is a disjunction of literals) containing at most *one* positive literal. Define a *horn formula* to be a CNF formula such that each clause is a horn clause. Define HornSAT to be the set of satisfiable horn formulas.

Prove that HornSAT is complete for **P**.

Hint: To show HornSAT is in **P**, consider simplifying a formula if it contains a clause with a single literal.

Hint: To show that HornSAT is hard for **P**, do a reduction from CVP. Also, observe that the horn clause $(\neg x_1 \vee \neg x_2 \vee \ldots \vee \neg x_n \vee y)$ is equivalent to $(x_1 \wedge \ldots \wedge x_n) \rightarrow y$.

Question 4. (10 marks) The *makespan* problem is informally defined as follows. We are given n jobs that we wish to schedule using m processors; each job has small runnning time, and must be scheduled on one from a given subset of the processors; we wish to schedule the jobs such that the *span*, or worst-case finishing time over all processors, is as small as possible. The makespan language, MS, is the decision version of this problem.

More rigorously, we define MS as follows. An instance is a sequence $n, m, k, J_1, \ell_1, J_2, \ell_2, \ldots, J_n, \ell_n$ where $n \geq 1$ is the number of jobs, $m \geq 1$ is the number of processors, each $J_j \subseteq \{1, 2, \ldots, m\}$ is the set of processors that we allow job j to be scheduled on, and each ℓ_j is the running time of job j, $\ell_j \in \{1, 2, 3\}$. (Note that if we allowed arbitrary binary integers for the running times, then it would be easy to show this problem **NP**-hard by reducing PARTITION to it.) A *schedule* is a function $S : \{1, 2, \ldots, n\} \rightarrow \{1, 2, \ldots, m\}$ such that for every job j, $S(j) \in J_j$; the *span* in S of processor i is $\sum_{S(j)=i} \ell_j$; the span of S is the maximum value of the span of i over all processors i. MS is the set of such instances for which there is a schedule with span $\leq k$. It is easy to see that MS is in **NP**.

Prove that MS is **NP**-hard by showing how to reduce 3DM (defined above) to MS.

Hint: We will have one processor for each edge of the 3DM instance.