EECS 219C: Computer-Aided Verification **Sanjit A. Seshia**

Homework 1: Boolean Functions, SAT, and BDDs

Assigned: January 31, 2007 *Due in class: February 14, 2007*

Note: For this and subsequent homeworks, if a problem requires you to come up with an algorithm, you should prove your algorithm's correctness as well as state and prove its asymptotic running time. See the webpage for rules on collaboration.

1. **CNF and DNF** (25 points)
   Consider the following DNF formula on $2n$ variables:

   $$(x_1 \wedge x_2) \vee (x_3 \wedge x_4) \vee \ldots \vee (x_{2n-1} \wedge x_{2n})$$

   (a) (15 points) Suppose you rewrite the above formula into CNF without introducing any new variables, by repeated application of the following distributive law: $y_i \vee (y_j \wedge y_k) = (y_i \vee y_j) \wedge (y_i \vee y_k)$.
   Prove that the resulting CNF formula is exponential-size in n.

   (b) (10 points) Can you rewrite the above formula in CNF using a total of n additional (new) variables so that the CNF comprises a total of $2n + 1$ clauses? Justify your answer.

2. **Renamable Horn-SAT** (20 points)
   A renamable Horn formula is a CNF formula that can be made into a Horn formula by complementing some of its variables (i.e., a variable is replaced by its complement, and if the formula turns out to be satisfiable, the relevant bits of the satisfying assignment are complemented to get a solution to the original problem). For example, consider the CNF formula

   $$(x_1 \vee \neg x_2 \vee \neg x_3)(x_2 \vee x_3)(\neg x_1)$$

   This is neither Horn nor negated Horn, but if we complement x_1 and x_2, it turns into a Horn formula.

   Give a polynomial-time algorithm to check whether a formula on n variables comprising m CNF clauses is renamable Horn. (Hint: try to express this problem itself as a SAT problem.)

3. **DLL Algorithm and Exponential Search** (20 points)
   The *pigeon-hole SAT problem* expresses the problem of finding a way to place n pigeons in $n - 1$ pigeon-holes such that no hole contains more than one pigeon. Obviously,

this problem is unsatisfiable. If x_{ij} is true when pigeon i is placed in hole j, then this problem can be written as

$$\bigvee_{j=1}^{n-1} x_{ij}, \quad \text{for all } i \in \{1, 2, \ldots, n\}$$

$$\neg x_{ik} \vee \neg x_{jk}, \quad \text{for all } k \in \{1, 2, \ldots, n-1\},\ i, j \in \{1, \ldots, n\}, i \neq j$$

The first set of CNF clauses require a pigeon to be placed in some hole, while the second enforce the constraint that no hole contains more than one pigeon.

Prove that the basic DLL search algorithm (without learning) will take exponential time (in n) to decide the unsatisfiability of the pigeon-hole SAT problem (for $n \geq 4$), no matter what order variables are branched on. (Hint: use an inductive argument)

4. **Comparing head-tail pointers with 2-literal watching** (20 points)
The SAT solver SATO introduced a technique for BCP based on so-called "head" and "tail" pointers. The idea is to watch 2 literals per clause, just like the Chaff SAT solver's scheme we discussed in class, except that these literals must be the first non-zero literals reachable from either end of the clause.

For example, in the clause $(x_1\ \overline{x_2}\ x_3\ x_5\ \overline{x_{10}})$ where $x_{10} = 1$ is the only assignment so far, the head literal will be x_1 and the tail literal x_5.

If a head literal is assigned 0, then we look to its right for the next non-zero literal to watch. Similarly, for a tail literal, we look to its left. The important point is that we always maintain the invariant that all the literals to the left of the head literal and to the right of the tail literal evaluate to 0.

(a) (4 points) For the Head-Tail scheme, state how one detects if the clause becomes a unit clause and a conflict clause.

(b) (16 points) Consider the following clause:

$$(x_1\ \overline{x_2}\ x_4\ \overline{x_6}\ x_7\ x_{10}\ x_{15})$$

For the following steps of assignments and backtracks, indicate how both SATO's Head-Tail scheme and Chaff's 2-literal watching will operate by showing the position of the watched literals after each step for each scheme. Indicate watched literals by arrows pointing down into the clause on them. Indicate implications when they happen, and where the two schemes behave differently.

Assume that all variables are initially un-assigned. Then the following steps occur in sequence:
1. $x_1 = 0$
2. $x_4 = 0$, $x_{10} = 0$, $x_7 = 0$
3. $x_{15} = 0$
4. $x_2 = 1$

5. Backtrack, undoing assignments made in steps 2-4
6. $x_2 = 0$.

5. **BDDs** (15 points)
A Boolean function on n variables $f(x_1, x_2, \ldots, x_n)$ is said to be symmetric if for every permutation π of its variables $f(x_1, x_2, \ldots, x_n) = f(\pi(x_1), \pi(x_2), \ldots, \pi(x_n))$.

Prove that the number of nodes of a BDD representing a symmetric Boolean function is $O(n^2)$.

[Hint: consider how the number of nodes varies in going from level i to level $i + 1$.]