Assignment 4: Imperative Procedures & Decision Procedures
15-414/15-424 Bug Catching: Automated Program Verification

Due: **11:59pm**, April 5
Total Points: 50

1. **Partially incorrect (15 points).** Consider the following rule for dealing with recursive procedure contracts, which allows the assumption that recursive calls within a procedure body satisfy the contract to prove partial correctness of the procedure.

$$(\text{[rec]}) \quad \frac{\Gamma, \forall x_1, \ldots, x_n. A \to [\mathtt{m}()]B \vdash A \to [\alpha]B, \Delta}{\Gamma \vdash A \to [\mathtt{m}()]B, \Delta} \quad (\alpha \text{ is body of } \mathtt{m}, x_1, \ldots, x_n \text{ used in } \alpha)$$

   Is this rule sound? If so, prove it either by derivation or by giving a semantic argument. It it is not sound, then provide a counterexample proof that is uses this rule, but is unsound (i.e., comes to a false conclusion).

2. **Exploding formulas (10 points).** A formula is in *Disjunctive Normal Form* (DNF) if it is a disjunction of conjunctive clauses. That is, for a set of literals ℓ_{ij}, the formula is of the form:

$$\bigvee_i \left(\bigwedge_j \ell_{ij} \right)$$

   An arbitrary propositional formula can be converted to DNF by eliminating double negations, applying De Morgan's law, and distributive rules. Describe a propositional formula containing n literals that is of size linear in n, but whose equivalent DNF has at least 2^n clauses. For full credit, be sure to explain why the formula explodes in size by describing how it is transformed to DNF, and how many clauses are generated.

3. **Pigeonhole SAT (15 points)** The pigeonhole problem asks us to find a one-to-one mapping between n pigeons and m holes. Obviously, this isn't possible when $n > m$. Consider an encoding of this problem as SAT for n pigeons and $n - 1$ holes, where we have the following CNF clauses and propositional variables p_{ij} which assert that pigeon i is placed in hole j.

   - *Pigeon clauses*: For each pigeon $1 \leqslant i \leqslant n$, assert that it is placed in some hole.

$$p_{i,1} \vee \ldots \vee p_{i,n-1}$$

   - *Hole clauses*: For each hole $1 \leqslant j < n$ and each pair of pigeons $1 \leqslant i < k \leqslant n$, these two pigeons aren't placed in the same hole:

$$\neg p_{i,j} \vee \neg p_{k,j}$$

   First, write down a CNF for the pigeonhole problem for $n = 3$. Then, apply the DPLL algorithm with clause learning to the formula. You should write down the steps of your evaluation in the following form:

   (1) Decide p
   (2) Unit propagate q from clause C_2
   (3) Decide $\neg r$
   (4) Unit propagate s from clause C_1
   (5) Conflicted clause C_1
   (6) Learn conflict clause $\neg p \vee r$

(7) ...

You are free to generate conflict clauses using any of the methods described in Lecture 13 [1], but you should explain how you find them.

4. **Positive reasoning (10 points)** The DPLL(T) algorithm given on page 8 of Lecture 14 constructs a conjunctive T-formula to send to the theory solver using an interpretation returned by the SAT solver as follows:

$$\psi \equiv B^{-1}\left(\bigwedge_{i=1}^{n} P_i \leftrightarrow I(P_i)\right)$$

A potential optimization would be to replace this step with one that only sends the conjunction of theory literals that the SAT interpretation assigns *true*:

$$\psi \equiv B^{-1}\left(\bigwedge_{\{i:I(P_i)=true\}} P_i\right)$$

Unfortunately, this is not sound. Give an example formula for which this optimization would reduce the number of times DPLL(T) must iterate, but would cause it to yield an incorrect answer.

[1] Available at `https://www.cs.cmu.edu/~15414/lectures/13-dpll.pdf`