Integrated Systems Design: Introduction to Formal Verification
CPSC 513, Term 1, Winter 2008–2009
Assigned Thursday September 11. Due Thursday October 2.

Homework #1

Questions 1–4 should be done with paper and pencil, and handed in. The results for question 5 should be submitted by email to `mitchell@cs.ubc.ca`. Keep your answers brief, and **get started early!**

1. **Canonical Forms.** In each question a representation and a set are listed. For each, specify whether the representation is canonical for the set. If it is not canonical, explain why.

   (a) Hexidecimal notation (base-16) for the set of nonnegative integers.

   (b) For a vector in $\mathbb{R}^2$, distance from the origin and counter-clockwise angle from the positive x-axis.

2. **Equivalence.** Are the following two Boolean expressions equivalent? In other words, given the same inputs a_0, b_0 and c_0, do they produce the same outputs s_0 and c_1? Provide evidence of your claim.

   - The functions: $c_1 = \overline{a_0 b_0} \cdot \overline{a_0 c_0} \cdot \overline{b_0 c_0}$ and $s_0 = c_o \; ? \; a_0 \cdot b_0 \cdot c_0 : a_0 + b_0 + c_0$
   - The circuit shown in figure 1.

3. **ROBDDs.** Draw a good Reduced Ordered Binary Decision Diagram for the Boolean expression (given in C notation):

   $$g \; ? \; (f \; ? \; e : d) : (c \; ? \; b : a)$$

   where a through g are Boolean variables. **Bonus:** Demonstrate that your BDD is optimal, in the sense that it uses the fewest possible nodes to represent the expression.

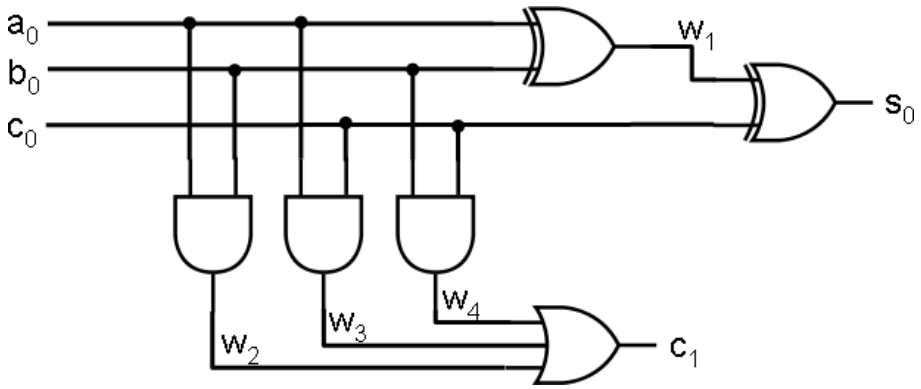

Figure 1: Circuit for questions 2 and 4.

4. **SAT.** The CNF expression for circuit outputs in terms of inputs is often very large, so the normal approach for SAT-based verification is to define new variables for each internal wire and then construct clauses for each gate to force the outputs of that gate (which may be either circuit outputs or one of the new internal variables) to be consistent with the inputs of that gate (which may be circuit inputs and/or internal variables).

   (a) Using the inputs a_0, b_0 and c_0 and whichever internal wire variables w_i for $i = 1, \ldots, 4$ shown in figure 1 are relevant, construct such an expression in CNF form for s_0.

   (b) Assume that you have created a Boolean expression S for the output s_0 as above. Now somebody gives you a Boolean expression $\hat{S}$ that describes an "optimized" version of the circuit taking inputs $\hat{a}_0$, $\hat{b}_0$ and $\hat{c}_0$ and producing output $\hat{s}_0$, and that person claims that the two circuits are equivalent. Write a Boolean expression (in terms of S, $\hat{S}$, s_0, $\hat{s}_0$, a_0, b_0, c_0, $\hat{a}_0$, $\hat{b}_0$, and/or $\hat{c}_0$) which is satisfiable if and only if the two circuits are **not** equivalent (use any Boolean operators that are convenient—there is no need to reduce to CNF).

5. **Write a combinational circuit equivalence checker.** (Thanks to Alan Hu for providing this question.) The program takes the names of two input files containing circuits and determines whether the two circuits are equivalent; if they are not equivalent, it specifies how they can be distinguished.

   To give you interesting data to work with, you will use the ISCAS 85 benchmarks. This is an industry-standard set of combinational benchmarks. The benchmarks have names like `c1908`. You are to use your program to compare the version of the circuit in the `DATA` subdirectory (e.g. `c1908.isc`) with a logic-optimized version of the same circuit in the `NONREDUN` subdirectory (e.g. `c1908nr.isc`). These circuits are supposed to be the same. Note, however, that in the past there were bugs in the benchmark set—circuits that were supposed to be the same were actually different. You can see if your program can catch these mistakes by comparing against the old versions in the `NONREDUN` subdirectory (e.g. `c1908nr_old.isc`). Given that there were bugs in the benchmark suite before, there may still be bugs, and maybe your program can find them!

   You have a choice of solving this problem using either BDDs or a SAT solver (**bonus:** try both). At the web site you will find links to descriptions of both versions of the problem, including skeleton code that you will modify and links to software packages for BDDs and SAT.

   When you are finished, email to `mitchell@cs.ubc.ca` your code (either `cmbcmp-bdd.c` or `cmbcmp-sat.c`) and a brief description of your results on the benchmarks.