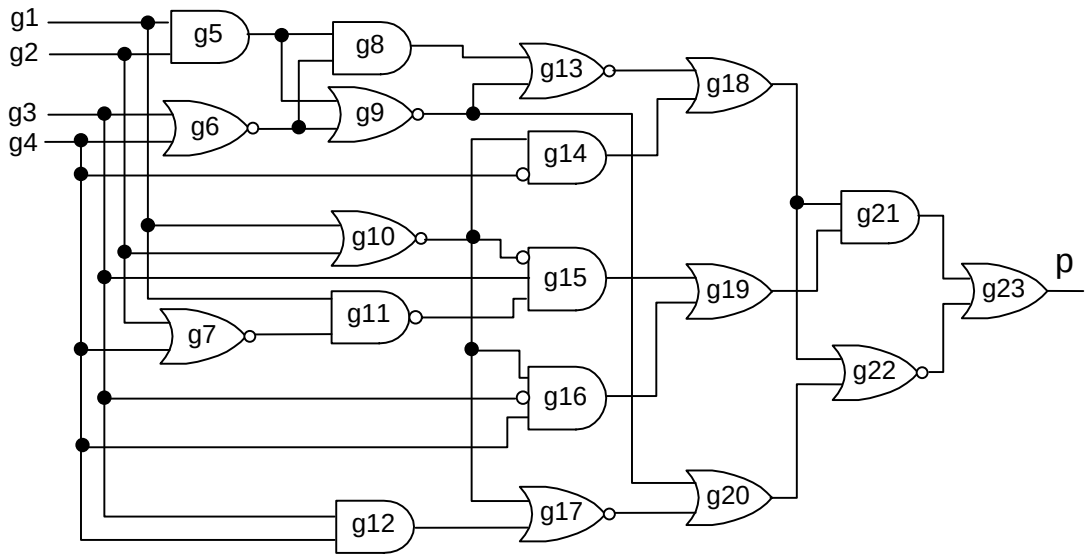
SAT Algorithms --- FLOLAC 2009

Prof. Chung-Yang (Ric) Huang, DV Lab, GIEE, NTU

Homework Assignment (Due: 9:00am, Wednesday July 1, 2009)

Id:_____ Name:_____

1. In the circuit below, the bubble "○" means "inversion", and the dot "●" is for the wire connection. The ID for the gate is the "number" on its name. For example, the ID of gate "g8" is 8.



(a) What are the CNF formulae for the following primitive gates:

(i) f = inv(a);

(ii) f = and(a, b'); // b' is the inversion of b

(iii) f = nand (a, b);

(iv) f = or(a, b);

(v) f = nor(a, b);

(b) [Literal counts of a gate] The literal counts are the number of times a literal appears in the clauses. Given a variable, its positive and negative literal counts are represented as a pair "(negLitCount, posLitCount)". For example, given the following clauses,

(a + f') (b + f') (a' + b' + f)

the literal counts for a, b, and f are (1, 1), (1, 1), and (2, 1), respectively.

To get the literal counts of a gate, we need to examine the CNF formulae of this gate and its fanouts. We then count the appearances of its positive and negative literals in these clauses.

Please list the literal counts (as pairs) for all the gates in the circuit (including PIs).

(c) [Decision order] We will determine the decision order of the gates in the circuit based on the "literal counts" of their corresponding CNF formulae. The detailed rules are as follows:

(i) The "decision score" of a gate is the bigger number of its literal counts.

(ii) The decision value is the opposite polarity of the literal with the bigger count. If the counts for the positive and negative literals are the same, choose 0 as the decision value. For example, if the (0, 1)-literal counts of the gate 'f' and 'g' are (5, 2) and (3, 3), then their decision scores are 5 and 3, and their decision values are 1 and 0.

(iii) The decision order of the gates is determined by the decision scores, with the bigger scores in the front. If the scores of two gates are tied, check the counts of the other literals. If tied again, compare their IDs (bigger ID wins). For example, if the literal counts of gates 'f', 'g' and 'h' are (5, 2) and (3, 4), (4, 1), the decision order will be (f = 1) → (g = 0) → (h = 1).

(iv) The orders remain unchanged throughout the decision process.

Please derive the top 7 decision gates in the circuit (including PIs).

(d) [Conflict-driven learning] We will try to witness p(g23) = 1. Please follow the decision orders and values in (c) to make the decisions, perform logic implications, and construct the implication graph. Do not make decision on a gate if it has been implied in an earlier decision level.

You should encounter a conflict after a few decisions. Please perform the conflict analysis to derive the conflict sources on the first UIP cut and construct a learned gate (i.e. AND gate with constrained value '0' on its output) for it.

You can refer to the C++ code below for the procedure in identifying the first UIP cut.

```
// ------------------------------------------------
// "imp0Src" is the list of implications that
//     imply the conflict gate '0'
// "imp1Src" is the list of implications that
```

```
//     imply the conflict gate '1'
list<ImpNode>
conflictAnalysis(const list<impNode>& imp0Src,
                 const list<impNode>& imp1Src) {
   int numMarked = 0; // num of marks in last dLevel
   ImpNode imp;
   list<ImpNode> conflictSrc;  // to be returned
   for_each_imp(imp, imp0Src)
      checkImp(imp, numMarked, conflictSrc);
   for_each_imp(imp, imp1Src)
      checkImp(imp, numMarked, conflictSrc);
   // Traverse backwards in the implication list of
   // the last decision level
   for_each_imp_rev(imp, lastDLevelImps) {
      if (!imp.isMarked()) continue;
      if (--numMarked == 0) {   // UIP found!!
         conflictSrc.push_back(imp);
         break;  // ready to return
      }
      imp.unsetMark();
      for_each_imp_src(imp_src, imp) {
         // implication sources of imp
         // (i.e. incoming edges on the imp graph)
         checkImp(imp_src, numMarked, conflictSrc);
      }
   }
   for_each_imp(imp, conflictSrc)
      imp.unsetMark();
   return conflictSrc;
}

void checkImp(ImpNode& imp, int& numMarked,
              list<impNode>& conflictSrc) {
   if (imp.isMarked()) return;
   imp.setMark();
   if (!imp.isLastDecisionLevel())
      conflictSrc.push_back(imp);
   else ++numMarked;
}
```

(e) [Witness generation] Backtrack from the learned constraint in (d) to an earlier decision level. What is the derived learned implication? At which decision level? Perform BCP on this learned implication. Will there be another conflict? If yes, perform conflict-driven learning again. If not, pick the next unassigned gate in the decision ordered list to make the next decision. Continue this process until a witness is found, or conclude that this target assignment is unsatisfiable.

2. Solving Sudoku (http://en.wikipedia.org/wiki/Sudoku) is known to be NP-Complete and can be modeled as a SAT problem. Please define a Sudoku problem as "formal" as possible, and design a SAT-based algorithm (in pseudo code) to solve it. (You don't need to go into details as source code. Just the algorithm.)