DPLL and Watched Literals

Week 7: 2-6 November 2015

Please attempt the entire worksheet in advance of the tutorial, and bring with you all work, including (if a computer is involved) printouts of code and test results. Tutorials cannot function properly unless you do the work in advance.

You may work with others, but you must understand the work; you can't phone a friend during the exam.

Assessment is formative, meaning that marks from coursework do not contribute to the final mark. But coursework is not optional. If you do not do the coursework you are unlikely to pass the exams.

Attendance at tutorials is **obligatory**; please let your tutor know if you cannot attend.

A brute-force search for valuations that satisfy a given conjunctive normal form (cnf) is simple to code:

```
function SAT(phi,V)
   phi|V = {} // {} is the empty set of clauses, equivalent to ⊤
   ||
      {} ∉ phi|V // {} is the empty clause, equivalent to ⊥
      &&
      let A = chooseLiteral (phi|V)
      in
          SAT (phi,V ^ A)
          ||
          SAT (phi,V ^ not A)
```

where `phi|V` is the result of simplifying each clause of `phi` using `V` (remove any literal set false by `V`; remove any clause set true by `V`), and `chooseLiteral(phi|V)` chooses some literal occurring in the simplification .

We can picture the execution of this code by constructing a tree, with a node for each call of `SAT`. The node for a call of `SAT(phi,V)` is labelled `V`, and has a child node for each recursive call it makes, if any.

1. Draw the tree resulting from a call `SAT(phi,[])`, where `phi` consists of the following set of clauses

$$\left\{\begin{array}{lrrrl} \{ & A, & B, & C & \}, \\ \{ & \neg A, & \neg B, & \neg C & \}, \\ \{ & B, & C, & D & \}, \\ \{ & \neg B, & \neg C, & \neg D & \}, \\ \{ & A, & C, & D & \}, \\ \{ & \neg A, & \neg C, & \neg D & \}, \\ \{ & A, & B, & D & \}, \\ \{ & \neg A, & \neg B, & \neg D & \} \end{array}\right\}$$

and `chooseLiteral` selects the first available literal, in the ordering where atoms are ordered alphabetically, and for each atom, X, $\neg X$ comes immediately after X.

2. In this question you will solve the same example, using a single watched literal.

   The first few steps are done for you.

A		$\top$	$\top$	$\top$	$\top$							
B			$\top$	$\top$	$\top$							
C				$\top$	$\bot$							
D												
(A∨B∨C)	1											
(¬A∨¬B∨¬C)	1	2	3	×								
(B∨C∨D)	1											
(¬B∨¬C∨¬D)	1		2									
(A∨C∨D)	1											
(¬A∨¬C∨¬D)	1	2		*								
(A∨B∨D)	1											
(¬A∨¬B∨¬D)	1	2	3									

At each step of the search we use one column of the table provided to record in the first 4 squares the truth values assigned to the atoms, A, B, C, D, and in the remaining 12, the position of the watched literal for each of the eight clauses.

We start with the empty valuation, watching the first literal in each clause.

We only look at watched literals that would be refuted by the latest variable assignment. So, at each step we consider only those clauses watching the literal we want to falsify.Because falsifying this literal cannot affect the invariant for any other clauses.

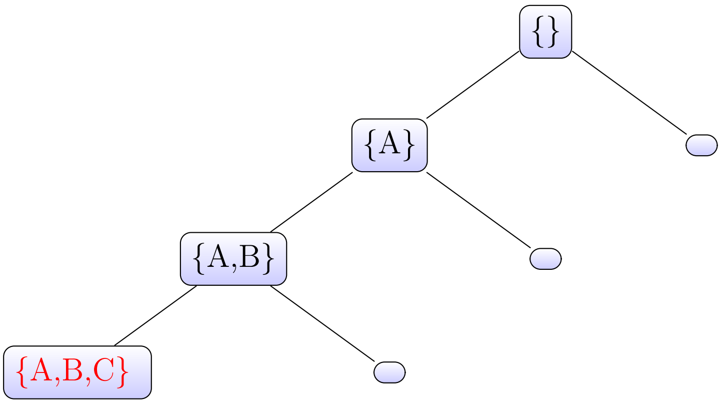
In the next step (in the second column), we make A true. To maintain the invariant, that each watched literal is either unassigned or true, we have to change the literal we are watching for those clauses where we are watching $\neg A$.

In the next step we make B true, and have find a new literal to watch for those clauses where we are watching $\neg B$.

In the following step, we try making C true. The first clause is OK, but in the second clause, we are watching $\neg C$ and every literal in this clause is refuted by the current assignment. We have to backtrack.

We place an $\times$ in the watched variable square if we find a clause for which the current partial valuation leaves no unrefuted literal we can watch to maintain our invariant. In this case the search must backtrack.

So far we have explored only part of the search tree:



We are searching for a valuation that makes all clauses true. Once we have arrived at a valuation that makes one clause false, our way is blocked. We know that we must find another path to satisfaction. Backtracking means that we retrace our steps, returning to previous partial valuations, to find some unexplored turning. The pattern of exploration specified by the recursion in Q 1 is to take the first unexplored path.

We backtrack, abandoning our attempt to falsify the watched literal, which we leave unchanged. This means that the invariant still holds. When we backtrack, we don't need to undo any changes we have already made, to the watched literals for other clauses, since making more variables unassigned won't harm our invariant.

In this case we found the contradiction before we got to looking at clause 6, but if we had taken the clauses in a different order the outcome would have been the same.

In this case we only have to backtrack one step. We try to make C false, and find that our invariant is satisfied with the existing watched literals.

You should continue the search, until you find a state where every watched literal is assigned $\top$, which means that we have a satisfying valuation.

Here is another copy of the table..

A											
B											
C											
D											

3. Now we will solve the same problem, using two watched literals.

   The invariant is that if one of the watched literals is $\bot$ then the other is $\top$. So if we make one of the watched literals false, and there are no more unrefuted literals we can watch, we have to make the other true, or fail.

A		$\top$	$\top$	$\top$	$\top$						
B			$\top$	$\top$	$\top$						
C				$\bot$	$\bot$						
D					$\bot$						
(A∨B∨C)	1,2										
(¬A∨¬B∨¬C)	1,2	2,3	$\neg C$								
(B∨C∨D)	1,2			+							
(¬B∨¬C∨¬D)	1,2		2,3								
(A∨C∨D)	1,2			+							
(¬A∨¬C∨¬D)	1,2	2,3									
(A∨B∨D)	1,2										
(¬A∨¬B∨¬D)	1,2	2,3	$\neg D$								

At each step we consider only those clauses watching the literal we want to falsify. Because falsifying this literal cannot affect the invariant for any other clauses.

For the first step, setting A true, we consider those clauses watching $\neg A$. We simply move our sights for one of the watched literals, as before.

When we set B true, we must look at those clauses watching $\neg B$ — clauses 2, 4, and 8. For clause 4 we can restore the invariant by watching a different literal, but for clauses 2 and 8 we have no room to move. If we make B true then we must make both C false, and D false.

We try both of these, in sequence — if either attempt fails then our attempt to make B true will have failed.

In the next column we have to look at clauses watching C (because we are trying to make C false. In each case (marked by '+') we find that the other watched literal is true, so we can make C false without violating our invariant.

In the final column, we try making D false. We have to check any clauses watching D. — There are none! This is sufficient to show that we have a total valuation and our invariant is satisfied.

Because we have managed to find a set of watched variables satifying our invariant, while making both C and D false, we have assigned values to all the variables.

For each watched pair of literals, if one is false under our valuation, then the other is true. Since none are unassigned, at least one of each pair is true, so each clause is true under the valuation we have found. We have achieved our goal.

You should now use the each watched literal method to find a satisfying valuation for the following clauses:

$$\left\{\begin{array}{lllllllllllllll}
\{ & \neg A, & \neg B, & C & \}, & \{ & B, & \neg C, & D & \}, & \{ & A, & \neg C, & D & \}, \\
\{ & \neg A, & B, & \neg C & \}, & \{ & B, & C, & \neg D & \}, & \{ & A, & C, & \neg D & \}, \\
\{ & A, & \neg B, & C & \}, & c\{ & \neg B, & \neg C, & \neg D & \}, & \{ & \neg A, & C, & D & \}
\end{array}\right\}$$

1. One watched literal:

A											
B											
C											
D											

You should check your valuation against each clause to see every clause is satisfied.

2. Two watched literals

A											
B											
C											
D											

If you're not yet satisfied — and you have have both the time and the inclination — you could add the negation of your satisfying valuation to the original clauses, and use either method again to see if you can find a different satisfying valuation.