Lecture 2: SAT Solvers & the DPLL Algorithm

Adam Hawley

March 26, 2019

Contents

1 Why SAT Matters for AI?

- SAT is NP-Complete; it is a hard problem.
- Many other problems can be converted to SAT.

2 Satisfying a Clause

A **literal** is a propositional symbol or the negation of one. For a clause to be true in a model it is enough for one of the literals to be true in that model.

3 Breaking Clauses

For a clause to be false in a model, all literals must be false in that model. We say the model *breaks* the clause.

4 When a Model Does Not Satisfy a CNF Formula

A CNF (conjunctive normal form) is a conjuction of clauses. In these notes often the phrase 'a CNF' will appear in the place of 'a formula in CNF' in the interests of brevity. Since it is a conjunction, if a model breaks even one clause it fails to satisfy the CNF. If all clauses are satisfied then so is the CNF.

5 Determining Satisfiablility

With n propositional symbols there are 2^n models. We can determine whether a CNF is satisfiable by enumerating all models. If we come across a satisfying model then the answer is YES, otherwise (after checking all 2^n models), the answer is NO.

Note: We typically do not need a fully defined model to decide whether a clause is satisfied (nor if it is broken).

e.g. if $A = true$, then $(A \vee \neg B \vee C)$ is satisfied regardless of the truth-values of B and C. or if $A = false, B = true, C = false$ then $(A \vee \neg B \vee C)$ is broken, regardless of the truth value of D.

6 SAT Solving As Search

One can view the SAT problem as a search for a satisfying model. The states are partially-defined models, i.e. truth assignments for some of the propositional symbols. We can move to a new state by assigning true/false to a variable. And can also backtrack to an earlier state. DPPL is depth-first search with simple, but effective heuristics.

7 Pure Symbols

Consider:

$$A \vee \neg B \tag{1}$$

$$\neg B \vee \neg C \tag{2}$$

$$A \vee C \tag{3}$$

Given these three statements, we can say that A and B are both **pure**, since they have the same 'sign' in all clauses. C, however, is impure.

Note: If a CNF has a model, then it has one with all pure symbols set to make their literals true.

All clauses containing a given pure symbol will be satisfied and other clauses won't depend on it. So **fix** the truth-values of pure symbols.

Given the same statements, if we have $B = false$, then $(\neg B \vee \neg C)$ is already true and therefore C *becomes* pure. In general, when looking for pure symbols we can ignore clauses already known to be true.

8 Unit Clause Heuristic

A **unit clause** contains a single literal. If $B = true$ then $(\neg B \vee \neg C)$ simplifies to $\neg C$ and so C must be set to $false$. In general, if all literals bar one are $false$ in a (particularly-built) modek, then **fix** the last one to satisfy the clause.

9 Unit Propogation

Focing a variable to take a particular value may generate a *cascade* of forced assigments. For example, suppose $C \vee A$ is one of our clauses. If C is set to false then A must be set to true. This is called **unit propogation**.

10 Component Analysis

As we assign variables, satisfied clauses can be removed and literals can be removed from yet-to-be-satisfied clauses. The resulting CNF may end up being representable as $X \wedge Y$ where X and Y are both CNFs with no overlapping variables. X and Y are then components which can be worked on separately.

11 Variable & Value Ordering

Which variable to try next? Which variable to try first? *Degree heuristic* chooses the variable which appears most frequently over the remaining clauses.

12 Other Tricks

- Intelligent Backtracking (as opposed to chronologival backtracking in standard depth-first search).
- Clause learning (very important in modern SAT solvers, it is where you write clauses which weren't given explicitly at the beginning).
- Random restarts
- Good programming!