Lecture 2: SAT Solvers & the DPLL Algorithm

Adam Hawley

April 7, 2019

Contents

1 Intro

The slides are very similar to the content of the book. See chapter 7.

2 Knowledge Bases

Knowledge base Set of **sentences** in a **formal** language.

The declarative approach to building an agent (or other system) is to tell it what it needs to know. Then it can ask itself what to do and the answers should follow from the KB. Agents can be viewed:

- At the knowledge level: what they know, regardless of how they are implemented
- At the implementation level: data structures in KB and algorithms that manipulate them

3 Logic in General

- **Logics** are formal languages for representing information such that conclusions can be drawn.
- **Syntax** defines the sentences in the language.
- **Semantics** define the *meaning* of the sentences; i.e define truth of a sentence in a world.

4 Entailment

- **Entailment** means that one thing follows from another:

$$KB \models \alpha \tag{1}$$

This means knowledge base KB entails sentence α if and only if α is true in all worlds where KB is true. E.g. $x + y = 4$ entails $4 = x + y$ Entailment is a relationship between sentences (i.e. syntax) that is based on semantics.

5 Models

Logicians typically think of **models**, which are formally structured worlds with respect to which truth can be evaluated. We say m **is a model of** a sentence α if α is true in m. M(α) is the set of all models of α

6 Inference

In logic an inference is a procedure by which you can deduce that something does follow from something else.

- $KB \vdash_i \alpha$ = sentence α can be derived form KB by procedure i.
- **Soundness**: i is sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$.
- **Completeness**: i is complete if whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$.

7 Propositional Logic: Syntax

Propositional logic is the simples logic. The proposition sumbold P_1, P_2 etc are senetences. If S is a sentence, ¬S is also a sentence (through negation). This also applies for the rules of:

- Conjuction
- Disjunction
- Implication
- Biconditional

8 Logical Equivalence

Two sentences are **logically equivalent** iff true in the same models:

- $\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

9 Validity & Satisfiability

A sentence is **valid** if it is true in all models (e.g $True$, $A \vee \neg A$, $A \implies A$). Validity is connected to inference via the **Deduction Theorem**:

- $KB \models \alpha$ if and only f $(KB \implies \alpha)$ is valid.

A sentence is **satisfiable** if it is true in *some* model (e.g. $A \vee B$). A sentence is **unsatisfiable** if it is true in *no* models (e.g. $A \wedge \neg A$) Satisfiability is connected to inference via the following:

- $KB \models \alpha$ if and only if $(KB \wedge \neg\alpha)$ is unsatisfiable, i.e. prove α by *reductio ad absurdum*.

10 Proof Methods

Contents

11 Why SAT Matters for AI?

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

12 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.

13 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.

14 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.

15 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.

16 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.

17 Pure Symbols

Consider:

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

$$\neg B \vee \neg C \tag{3}$$

$$A \vee C \tag{4}$$

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.

18 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.

19 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**.

20 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.

21 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.

22 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!