Notes on Satisfiability-Based Problem Solving

Linear Time: Unit Propagation and Horn-SAT

David Mitchell
mitchell@cs.sfu.ca
February 5, 2020

In this section, we describe a linear time algorithm for an efficient but incomplete reasoning process, called unit propagation, that lies at the core of high-performance SAT solvers. We also apply it to give a linear-time algorithm for formulas in a restricted syntactic form.

Terms and Conventions In this section Γ denotes a set of m clauses, over a set S of n atoms, with length (number of occurrences of literals) l. τ and α are (possibly partial) truth assignments for S. We will often write CNF formulas in a simple form where $(P \vee \neg Q) \wedge (R \vee \neg S)$ becomes $(P \vee \neg Q), (R \vee \neg S)$ or even $(P, \overline{Q}), (R, \overline{S})$. We say a literal is positive if it is an atom, and negative if it is a negated atom.

1 Unit Propagation

Unit propagation is an important simplification operation for CNF formulas based on the observation that, if Γ contains a unit clause (P), then every satisfying assignment for Γ must make P true. If Γ also contains a clause $(\neg P \vee Q)$, we may observe that it is "effectively unit", because the only way to satisfy it is to set Q true. Unit Propagation (UP) is the name for the process of following this line of implications.

Definition 1. The "Unit Clause Rule" (UCR) is the following simplification rule for CNF formulas.

> **UCR:** If Γ contains a unit clause (L), delete every occurrence of $\overline{L}$ from clauses of Γ, and delete every clause containing L from Γ.

We denote by UP(Γ) the result of repeated application of UCR until there are no unit clauses in the resulting formula. That is, it is a fixpoint of UCR.

The fixpoint of UCR is unique. That is, if at certain points there is more than one unit clause, the final outcome is independent of which we apply UCR to first.

Example 1. *Let* Γ *be*

$$(P), (P \vee Q \vee \neg R), (\neg P \vee R \vee U), (\neg P \vee \neg Q), (\neg Q \vee \neg P \vee R), (R \vee \neg U \vee V), (\neg P \vee Q \vee V)$$

Applying UCR with (P)*, we obtain*

$$(R \vee U), (\neg Q), (\neg Q \vee R), (R \vee \neg U \vee V), (Q \vee V).$$

We have a new unit clause $(\neg Q)$*, so apply UCR with* $(\neg Q)$*, and obtain*

$$(R \vee U), (R \vee \neg U \vee V), (V)$$

Finally, we apply UCR with the new unit clause (V)*, obtaining*

$$(R \vee U).$$

Exercise 1. *Construct UP(*Γ*), where* Γ *is*

$$(P)(P, U, V)(\overline{P}, \overline{Q})(\overline{P}, R)(\overline{R}, Q, S)(S, Q, \overline{T}, \overline{U}, V)(\overline{S}, Q, \overline{T}, \overline{W}, X)(\overline{P}, \overline{Q}, U, \overline{V})(P, Q, U, \overline{W}).$$

Proposition 1. *For any CNF formula* Γ*:*

1. Γ *is satisfiable iff UP(*Γ*) is;*
2. *if* $\square \in$ *UP(*Γ*) then* Γ *is unsatisfiable.*

In later sections we will see that efficient unit propagation is a central component in industrial-strength SAT Solvers.

2 Horn-SAT

Definition 2. A clause is Horn if it contains at most one positive literal. A formula is Horn if it is a conjunction (or set) of Horn clauses. Horn-SAT is the problem of deciding satisfiability of Horn Formulas.

We will see that unit propagation provides an algorithm for deciding Horn-SAT.

Proposition 2. *A set* Γ *of Horn clauses if unsatisfiable if an only if* $UP(\Gamma)$ *contains the empty clause.*

We will show first that a restricted form of unit propagation, which we call "positive unit propagation" is sufficient, after which it is easy to see that Proposition 2 holds.

Let Γ^- denote the set of negative clauses of Γ (those with no positive literal), and Γ^+ the non-negative clauses (those containing at least one positive literal).

Proposition 3. *A Horn formula Γ is satisfiable if either of the following hold:*

1. *Γ^- is empty;*
2. *Γ^+ has no unit clause. (This case includes the case of Γ^+ being empty.)*

Exercise 2. *Prove Proposition 3. (First, think why an arbitrary CNF formula with no negative clause is satisfiable.)*

Given a Horn formula Γ, generate a set of positive unit clauses Γ^{++} as follows. Initialize Γ^{++} to the unit clauses in Γ^+, and then apply following "Positive Unit" rule as many times as possible:

> If Γ^+ contains a clause of the form $(\neg P_1 \vee \neg P_2 \vee \ldots \vee \neg P_k \vee P)$, and all the positive unit clauses $(P_1), (P_2), \ldots (P_k)$, are in Γ^{++} but (P) is not in Γ^{++}, add (P) to Γ^{++}.

Now, define truth assignment τ by:

$$\tau(P) = \begin{cases} true & \text{if } (P) \in \Gamma^{++}, \\ false & \text{otherwise.} \end{cases} \tag{1}$$

Proposition 4. *Let τ be defined as in Equation 1. Then:*

1. *$\tau \models \Gamma$ if and only if $\tau \models \Gamma^-$;*
2. *If Γ is satisfiable, τ is the unique minimal satisfying assignment for Γ (that is, the satisfying assignment with the fewest atoms set true);*

Proof. (Sketch) By construction, τ satisfies every clause in Γ^+. Now, consider any application of the Positive Unit rule. The clause $(\neg P_1 \vee \neg P_2 \vee \ldots \vee \neg P_k \vee P)$ and clauses $(P_1), (P_2), \ldots (P_k)$ together imply (P). (Indeed, (P) can be derived from them by k applications of the resolution rule.) So, induction on the number of applications of the rule will show that every unit clause in Γ^{++} is implied by Γ and so every atom made true by τ is true in every satisfying assignment. It follows that, if τ satisfies Γ it is minimal. Further, if τ makes a clause of Γ^- false, then Γ is unsatisfiable, and otherwise $\tau \models \Gamma$. $\square$

Exercise 3. *Use the Horn-SAT algorithm described above to determine satisfiability of each clause set. For satisfiable sets, give the minimal model. For unsatisfiable sets, give the resolution refutation implied by the algorithm execution.*

1. $(P \vee \overline{Q} \vee \overline{R}), (Q \vee \overline{S}), (R), (\overline{W} \vee S), (W), (\overline{W} \vee \overline{X} \vee \overline{P})$
2. $(P \vee \overline{Q} \vee \overline{R}), (Q \vee \overline{S}), (R), (\overline{W} \vee S), (W), (\overline{W} \vee \overline{X} \vee \overline{P}), (\overline{S} \vee \overline{P} \vee X)$

3 Linear Time Positive Unit Propagation

We now show how to execute the positive unit propagation algorithm for Horn-SAT in linear time. (The method can be extended to work for general unit propagation.) Executed naively, the algorithm requires repeatedly scanning Γ^+ to find clauses to which the Positive-Unit rule applies. This leads to quadratic time - one scan of Γ^+ for each unit clause put into Γ^{++}, and in the worst case the number of scans is the number of clauses in $\Gamma+$.

To obtain linear-time execution, we need to find clauses to which the Positive-Unit rule applies more efficiently. The key observation is that, as long as at least one negative literal in a clause has not been made false, the Positive-Unit rule does not apply to it. We maintain two data structures. We use a "watched literal" data structure to efficiently detect when the Positive-Unit rule applies to a clause. We also keep a queue **Q** of positive literals, which will contain (P) if it is in Γ^{++}, but the algorithm has not "propagated" the effects of this to determine "new" applications of the Positive-Unit rule that are implied by the addition of (P) to Γ^{++}.

The watched-literal data structure is constructed over the set of non-unit clauses in Γ^+. Write each of these clauses with the literals ordered so that the positive literal comes last. We will "watch" one literal in each of these clauses, which initially will be the first literal. For each atom P, we construct a list of (pointers to) all watched occurrences of the literal $\overline{P}$ in clauses of Γ^+. Figure 1 illustrates an initial configuration of a watched-literal data structure.

As the algorithm proceeds, we move the watches in clauses to the right, maintaining the invariant that every for every literal $\overline{P}$ to the left of the literal being watched, the unit clause (P) is in Γ^+. It follows that the literal must be false in any satisfying assignment. This implies that, to satisfy a clause, we must either satisfy either the watched literal or a literal to its right. When we find that a watched literal has been made false, we scan the clause, moving to the right, looking for a new literal to watch. This must be the first literal we find that we do not (yet) know must be made false. If we reach the last literal

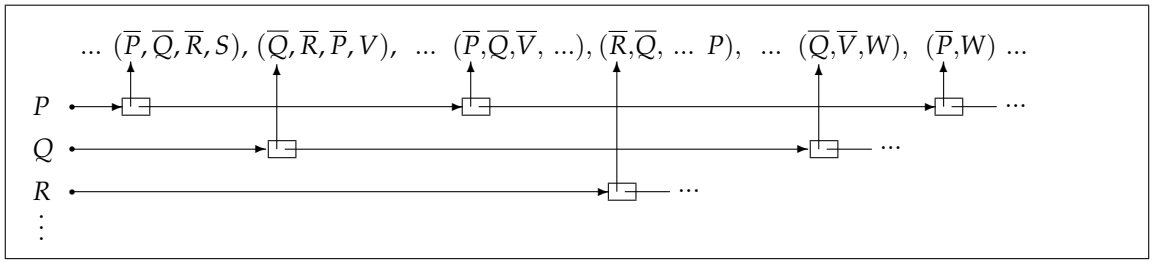

Figure 1: An initial configuration of a watched literal data structure. The clause list appears across the top; the list of watch lists down the left side.

of the clause in this scan, then we have a positive literal that must be satisfied, and add the corresponding unit clause to Γ^{++}.

To execute the algorithm, initialize **Q** with all the positive unit clauses in Γ^{+}, and then as long as **Q** is not empty, remove a unit clause (P) from **Q** and traverse the list of watched occurrences of $\overline{P}$. For each watched occurrence of $\overline{P}$, scan the clause C in which it appears for a new literal $\overline{R}$ to watch. If you find one, add (a pointer to) the occurrence of $\overline{R}$ in C to the watch list for R. If you do not find one, let R be the positive literal in C, and add (R) to Γ^{++}, and to **Q** (and stop watching C).

Example 2. *Suppose we are running the algorithm on a set* Γ *of Horn clauses with the initial watched literal data structure as shown in Figure 1. Suppose also that the only unit clauses in* Γ *is* (P) *and* (Q)*, and that the first iteration of the algorithm removes* (P) *from* **Q***. It will traverse the list of watched occurrences of* $\overline{P}$*. (There are 5 occurrences of the atom* P*, but only the three are in watched occurrences of* $\overline{P}$*. We ignore the other two.) At the first two clauses where* $\overline{P}$ *is watched, we move the watch to* $\overline{Q}$*. At the third, we add* (W) *to both* **Q** *and* Γ^{++}*. In the next iteration,* (Q) *is removed from* **Q***. There are three watched occurrences of* $\overline{Q}$*. The watches for the corresponding clauses then move to* $\overline{R}$*,* $\overline{R}$*,* $\overline{V}$ *and* $\overline{V}$ *(respectively, scanning the clauses left to right). The state of the watched-literal structure, at this point, is illustrated in Figure 2.*

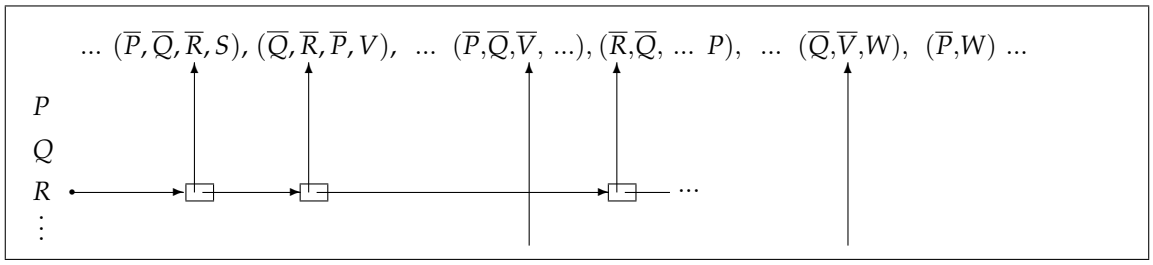

Figure 2: Configuration of the watched literal data structure corresponding to Example 2.

Proposition 5. *Horn-SAT can be solved in time O(l), where l is the number of literal occurrences.*

Proof. (Sketch) The data structures can be constructed in time $O(l)$. During execution, the algorithm visits each literal occurrence in Γ at most once, and the cost for each visit is $O(1)$, so total execution time is $O(l)$. □