Practical SAT Solving

Lecture 3

Carsten Sinz, Tomáš Balyo | May 17, 2018
The Resolution Rule

\[(l \lor x_1 \lor x_2 \lor \cdots \lor x_n) \land (\bar{l} \lor y_1 \lor y_2 \lor \cdots \lor y_m) \]

\[\frac{(x_1 \lor x_2 \lor \cdots \lor x_n \lor y_1 \lor y_2 \lor \cdots \lor y_m)}{x_1 \lor x_2 \lor \cdots \lor x_n \lor y_1 \lor y_2 \lor \cdots \lor y_m}\]

The upper two clauses are called *Input Clauses* the bottom clause is called the *Resolvent*
The Resolution Rule

\[(l \lor x_1 \lor x_2 \lor \cdots \lor x_n) \land (\tilde{l} \lor y_1 \lor y_2 \lor \cdots \lor y_m)\]

\[\frac{}{(x_1 \lor x_2 \lor \cdots \lor x_n \lor y_1 \lor y_2 \lor \cdots \lor y_m)}\]

The upper two clauses are called *Input Clauses* the bottom clause is called the *Resolvent*

Examples

\[(x_1 \lor x_3 \lor x_7) \land (\overline{x_1} \lor x_2) \vdash (x_3 \lor \overline{x_7} \lor x_2)\]
The Resolution Rule

\[(l \lor x_1 \lor x_2 \lor \cdots \lor x_n) \land (\neg l \lor y_1 \lor y_2 \lor \cdots \lor y_m)\]

\[\vdash (x_1 \lor x_2 \lor \cdots \lor x_n \lor y_1 \lor y_2 \lor \cdots \lor y_m)\]

The upper two clauses are called *Input Clauses* the bottom clause is called the *Resolvent*

**Examples**

- \[(x_1 \lor x_3 \lor \overline{x}_7) \land (\overline{x}_1 \lor x_2) \vdash (x_3 \lor \overline{x}_7 \lor x_2)\]
- \[(x_4 \lor x_5) \land (\overline{x}_5) \vdash (x_4)\]
### The Resolution Rule

\[
(l \lor x_1 \lor x_2 \lor \cdots \lor x_n) \land (\bar{l} \lor y_1 \lor y_2 \lor \cdots \lor y_m) \\
(x_1 \lor x_2 \lor \cdots \lor x_n \lor y_1 \lor y_2 \lor \cdots \lor y_m)
\]

The upper two clauses are called *Input Clauses* the bottom clause is called the *Resolvent*

### Examples

- \((x_1 \lor x_3 \lor \overline{x_7}) \land (\overline{x_1} \lor x_2) \vdash (x_3 \lor \overline{x_7} \lor x_2)\)
- \((x_4 \lor x_5) \land (\overline{x_5}) \vdash (x_4)\)
- \((x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2}) \vdash \)
SAT Tools – Resolution

The Resolution Rule

\[(l \lor x_1 \lor x_2 \lor \cdots \lor x_n) \land (\bar{l} \lor y_1 \lor y_2 \lor \cdots \lor y_m)\]

\[(x_1 \lor x_2 \lor \cdots \lor x_n \lor y_1 \lor y_2 \lor \cdots \lor y_m)\]

The upper two clauses are called *Input Clauses* the bottom clause is called the *Resolvent*

Examples

- \((x_1 \lor x_3 \lor \overline{x_7}) \land (\overline{x_1} \lor x_2) \vdash (x_3 \lor \overline{x_7} \lor x_2)\)
- \((x_4 \lor x_5) \land (\overline{x_5}) \vdash (x_4)\)
- \((x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2}) \vdash\)
- \((x_1) \land (\overline{x_1}) \vdash\)
SAT Tools – Resolution

The Resolution Rule

\[
(\bar{l} \lor x_1 \lor x_2 \lor \cdots \lor x_n) \land (\bar{l} \lor y_1 \lor y_2 \lor \cdots \lor y_m)
\]

\[
(x_1 \lor x_2 \lor \cdots \lor x_n \lor y_1 \lor y_2 \lor \cdots \lor y_m)
\]

Special Cases

- Tautological Resolvent \((x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2}) \models (x_1 \lor \overline{x_1})\)
  - Usually forbidden, does no harm, will be useful later

- Empty Clause \((x_1) \land (\overline{x_1}) \models \bot\)
  - The empty clause a.k.a conflict clause a.k.a ”⊥” is unsatisfiable

Notation

- \(R((x_1 \lor x_2), (\overline{x_1} \lor x_3)) = (x_2 \lor x_3)\)
Theorem: Resolution maintains satisfiability

Let $F$ be a CNF formula and $C_1$ and $C_2$ two of its clauses with a pair complementary literals. Then $F$ is satisfiable if and only if $F \land \text{R}(C_1, C_2)$ is satisfiable.
Theorem: Resolution maintains satisfiability

Let $F$ be a CNF formula and $C_1$ and $C_2$ two of its clauses with a pair complementary literals. Then $F$ is satisfiable if and only if $F \land R(C_1, C_2)$ is satisfiable.

Proof:

- If $F$ is not satisfiable then $F \land C$ for any $C$ is also not satisfiable.
- If $F$ is satisfiable and $\phi$ is a satisfying assignment of $F$ then we show that $\phi$ also satisfies $R(C_1, C_2)$.
  - If $C_1 = (l \lor P_1)$ and $C_2 = (\bar{l} \lor P_2)$ then $R(C_1, C_2) = (P_1 \lor P_2)$
  - Since $\phi$ satisfies both $C_1$ and $C_2$ it must satisfy at least one of the literals in $P_1$ or $P_2$.
    - if $\phi$ satisfies $l$ then it satisfies some literal in $P_2$
    - if $\phi$ satisfies $\bar{l}$ then it satisfies some literal in $P_1$
Theorem: Resolution maintains satisfiability

Let $F$ be a CNF formula and $C_1$ and $C_2$ two of its clauses with a pair complementary literals. Then $F$ is satisfiable if and only if $F \land R(C_1, C_2)$ is satisfiable.

Consequences

- If we manage to resolve the empty clause ($\perp$) the original formula is unsatisfiable.
Theorem: Resolution maintains satisfiability

Let $F$ be a CNF formula and $C_1$ and $C_2$ two of its clauses with a pair complementary literals. Then $F$ is satisfiable if and only if $F \land R(C_1, C_2)$ is satisfiable.

Consequences

- If we manage to resolve the empty clause ($\bot$) the original formula is unsatisfiable

Usage

- Proof of unsatisfiability – Resolution Proof
  - A resolution proof is a sequence of clauses such that each clause is either a clause of the original formula or a resolvent of two previous clauses ending with $\bot$. 
Theorem: Resolution maintains satisfiability

Let $F$ be a CNF formula and $C_1$ and $C_2$ two of its clauses with a pair complementary literals. Then $F$ is satisfiable if and only if $F \land R(C_1, C_2)$ is satisfiable.

Consequences
- If we manage to resolve the empty clause ($\bot$) the original formula is unsatisfiable

Usage
- Proof of unsatisfiability – Resolution Proof
  - A resolution proof is a sequence of clauses such that each clause is either a clause of the original formula or a resolvent of two previous clauses ending with $\bot$.

Example: $(x_1 \lor x_2), (\overline{x_1} \lor x_2), (x_1 \lor \overline{x_2}), (\overline{x_1} \lor \overline{x_2})$
Theorem: Resolution maintains satisfiability

Let $F$ be a CNF formula and $C_1$ and $C_2$ two of its clauses with a pair complementary literals. Then $F$ is satisfiable if and only if $F \land R(C_1, C_2)$ is satisfiable.

Consequences

- If we manage to resolve the empty clause ($\bot$) the original formula is unsatisfiable.

Usage

- Proof of unsatisfiability – Resolution Proof
  - A resolution proof is a sequence of clauses such that each clause is either a clause of the original formula or a resolvent of two previous clauses ending with $\bot$.

Example: $(x_1 \lor x_2), (\overline{x_1} \lor x_2), (x_1 \lor \overline{x_2}), (\overline{x_1} \lor \overline{x_2}), (x_2), (\overline{x_2}), \bot$
### Saturation Algorithm

- **INPUT:** CNF formula \( F \)
- **OUTPUT:** \{SAT, UNSAT\}

```plaintext
while (true) do
    \( R = \text{resolveAll}(F) \)
    if \( R \cap F \neq R \) then \( F = F \cup R \)
    else break

if \( \bot \in F \) then return UNSAT else return SAT
```
Saturation Algorithm

- **INPUT:** CNF formula \( F \)
- **OUTPUT:** \{ SAT, UNSAT \}

while (true) do
   \[ R = \text{resolveAll}(F) \]
   if \( R \cap F \neq R \) then \( F = F \cup R \)
   else break

if \( \bot \in F \) then return UNSAT else return SAT

Properties of the saturation algorithm:
- it is sound and complete – always terminates and answers correctly
- has exponential time and space complexity (always for Pigeons)
Unit Resolution

= at least one of the resolved clauses is unit (has one literal).

Example:

\[ R((x_1 \lor x_7 \lor \overline{x_2} \lor x_4), (x_2)) = (x_1 \lor x_7 \lor x_4) \]
Unit Resolution

= at least one of the resolved clauses is unit (has one literal).

Example:

\[ R((x_1 \lor x_7 \lor \overline{x_2} \lor x_4), (x_2)) = (x_1 \lor x_7 \lor x_4) \]

Unit Propagation

= a process of applying unit resolution as long as we get new clauses.

Example:

\[ (x_1) \land (x_7 \lor x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor x_3) \]
\[ (x_1) \land (x_7 \lor x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor x_3) \land (x_3) \]
\[ (x_1) \land (x_7 \lor x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor x_3) \land (x_3) \land (x_7 \lor x_2) \]
Easy Cases

SAT is not always hard, in the following cases it is polynomially solvable

- 2-SAT
- Horn-SAT
- Hidden Horn-SAT
- SLUR
2-SAT

2-SAT Formula

= each clause has exactly 2 literals.

Example:

- \((x_1 \lor x_3) \land (x_7 \lor \overline{x_3}) \land (\overline{x_1} \lor x_3)\)
- \((x_1 \lor x_2) \land (\overline{x_1} \lor x_2) \land (x_1 \lor \overline{x_2}) \land (\overline{x_1} \lor \overline{x_2})\)

Also called Binary SAT or Quadratic SAT

How to solve 2-SAT?
How to solve 2-SAT?

Saturation Algorithm

The resolution saturation algorithm is polynomial for 2-SAT

Proof:
- Only 2-literal resolvents are possible
- There are only $O(n^2)$ 2-literal clauses on $n$ variables

Complexity:
- Both time and space $O(n^2)$
- There exists a linear algorithm! [1]
Implication graph of a formula $F$ is an oriented graph that has:

- a vertex for each literal of $F$
- 2 edges for each clause $(l_1 \lor l_2)$
  - $\bar{l}_1 \rightarrow l_2$
  - $\bar{l}_2 \rightarrow l_1$

Example:
$$(\overline{x}_1 \lor x_2) \land (\overline{x}_2 \lor x_3) \land (\overline{x}_3 \lor x_1) \land (x_2 \lor x_4) \land (x_3 \lor x_4) \land (x_1 \lor x_3)$$
The next step is to analyze the *Strongly Connected Components* of the implication graphs.

- SCC = there is a path in each direction between each pair.
- Tarjan’s algorithm finds SCCs in $\mathcal{O}(|V| + |E|)$.
- If any $x$ and $\overline{x}$ literal pair is in the same SCC then the formula is UNSAT.
  - All the literals in an SCC must be all True or all False.
How to find the solution?

- Construct the *Condensation* of the implication graph
  - contract each SCC into one vertex
- Topologically order the vertices of the condensation
- In reverse topological order, if the variables do not already have truth assignments, set all the terms to true.

Example: $x_1 = x_2 = x_3 = True, x_4 = True$, the rest is already assigned.
How to solve 2-SAT?

**Linear Algorithm**

- Construct the Implication Graph
- Find all the SCCs
- Check if any SCC contains a complementary pair
- Construct a condensation of the implication graph
- Run topological sort on the condensation
- Construct the solution

**Complexity:**

- All the steps can be done in linear time
Horn Formula

A CNF formula is a *Horn formula* is each of its clauses contains at most one positive literal.

Example: \((\overline{x}_1 \lor \overline{x}_7 \lor x_3) \land (\overline{x}_2 \lor \overline{x}_4) \land (x_1)\)
Horn Formula

A CNF formula is a *Horn formula* is each of its clauses contains at most one positive literal.

Example: \((\overline{x}_1 \lor \overline{x}_7 \lor x_3) \land (\overline{x}_2 \lor \overline{x}_4) \land (x_1)\)

Solving Horn Formulas

- Perform unit propagation on the input formula
- If you resolve \(\bot\) then the formula is UNSAT otherwise it is SAT
- Get the solution:
  - Assign the variables in unit clauses to satisfy them
  - Set the rest of the variables to False
Mixed Horn

Mixed Horn Formula

A CNF formula is Mixed Horn if it contains only quadratic and Horn clauses.

Example: \((\overline{x}_1 \lor \overline{x}_7 \lor x_3) \land (\overline{x}_2 \lor \overline{x}_4) \land (x_1 \lor x_5) \land (x_3)\)

Questions:

- How to solve a Mixed Horn formula?
- How to hard is it to solve a Mixed Horn formula?
Mixed Horn Formula

We will reduce SAT to Mixed Horn SAT

For each non-Horn clause $C = (l_1 \lor l_2 \lor \ldots)$ do

- for each but one positive $l_i \in C$ introduce a new variable $l'_i$
- replace $l_i$ in $C$ by $\overline{l'_i}$
- add $(l'_i \lor l_i) \land (\overline{l'_i} \lor \overline{l_i})$ to establish $l_i = \overline{l'_i}$

Example:
$$(x_1 \lor \overline{x}_7 \lor x_3) \land (\overline{x}_2 \lor \overline{x}_4) \land (x_1 \lor x_5) \leadsto$$
$$\leadsto (\overline{x'_1} \lor \overline{x}_7 \lor x_3) \land (\overline{x}_2 \lor \overline{x}_4) \land (x'_1 \lor x_5) \land (x'_1 \lor x_1) \land (x'_1 \lor \overline{x}_1)$$
Hidden Horn Formulas

A CNF formula is *Hidden Horn* if it can be made Horn by renaming some of its variables.

Example:
\[(x_1 \lor x_2 \lor x_4) \land (x_2 \lor \overline{x}_4) \land (x_1) \rightsquigarrow (\overline{x}_1 \lor \overline{x}_2 \lor x_4) \land (\overline{x}_2 \lor \overline{x}_4) \land (\overline{x}_1)\]

Questions:
- How to recognize a Hidden Horn formula?
- How to hard is it to recognize and solve a Hidden Horn formula?
Recognizing Hidden Horn Formulas

Translate into 2-SAT

Let \( F \) be original formula, \( R_F \) contains the clause \((l_1 \lor l_2)\) if and only if there is a clause \( C \in F \) such that \( l_1 \in C \) and \( l_2 \in C \).

Example: \( F = (x_1 \lor x_2 \lor x_4) \land (x_2 \lor \overline{x_4}) \land (x_1) \)  
\( R_F = (x_1 \lor x_2) \land (x_1 \lor x_4) \land (x_2 \lor x_4) \land (x_2 \lor \overline{x_4}) \)

Recognize Hidden Horn

If \( R_F \) is satisfiable, then \( F \) is a hidden Horn formula. Furthermore, the satisfying assignment \( \phi \) of \( R_F \) identifies the variables to be renamed.

- if \( x_i = \text{True} \) in \( \phi \) then \( x_i \) needs to be renamed to \( \overline{x_i} \)
Single Look-ahead Unit Resolution - SLUR algorithm

01 if ⊥ ∈ unit-prop(F) then return UNSAT else return SLUR(F)

02 function SLUR(F)

03 if all variables appear in a unit clause then return SAT

04 v = select-variable(F)

05 F_1 = unit-prop(F ∧ (v))

06 F_2 = unit-prop(F ∧ (¬v))

07 if ⊥ ∈ F_1 and ⊥ ∈ F_2 then return GIVE-UP

08 if ⊥ ∈ F_1 and ⊥ /∈ F_2 then SLUR(F_2)

09 if ⊥ /∈ F_1 and ⊥ ∈ F_2 then SLUR(F_1)

10 if ⊥ /∈ F_1 and ⊥ /∈ F_2 then SLUR(F_1) or SLUR(F_2)

A CNF formula F is SLUR if the SLUR algorithm never gives up on for F (regardless of the choices in lines 04 and 10).
SLUR Formulas

Properties of SLUR Formulas [2]:

- SLUR formulas are solvable in polynomial time (using the SLUR algorithm)
- SLUR is an umbrella class for polynomially solvable classes – All Horn and Hidden Horn formulas are SLUR formulas
  - Also true for Extended Horn, CC-balanced, and Propagation Complete formulas
- It is co-NP complete to recognize whether a given CNF is a SLUR formula or not
Stochastic Local Search (SLS)

**SAT as an optimization problem:** minimize the number of unsatisfied clauses

Start with a complete random assignment $\alpha$:

\[
\begin{array}{cccccccccccc}
0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
\end{array}
\]

Repeatedly flip (randomly/heuristically chosen) variables to decrease the number of unsatisfied clauses:

\[
\begin{array}{cccccccccccc}
0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
\end{array}
\]
Local search algorithms are **incomplete**: they cannot show unsatisfiability!

Many variants of local search algorithms

Main question: Which variable should be flipped next?

- select variable from an unsatisfied clause
- select variable that increases the number of satisfied clauses most

How to avoid local minima?
GSAT Algorithm [3]

Maybe[Assignment] GSAT(ClauseSet S)
{
    for i = 1 to MAX_TRIES do {
        $\alpha$ = random-assignment to variables in S
        for j = 1 to MAX_FLIPS do {
            if ( $\alpha$ satisfies all clauses in S ) return $\alpha$
            $x$ = variable that produces least number of
            unsatisfied clauses when flipped
            flip $x$
        }
    }
    return Nothing // no solution found
}
SLS: Illustration

[Source: Alan Mackworth, UBC, Canada]
Walksat [4]

- Variant of GSAT
- Try to avoid local minima by introducing “random noise”
  - Select unsatisfied clause $C$ at random
  - If by flipping a variable $x \in C$ no new unsatisfied clauses emerge, flip $x$
  - Otherwise:
    - With probability $p$ select a variable $x \in C$ at random
    - With probability $1 - p$ select a variable that changes as few as possible clauses from satisfied to unsatisfied when flipped
Consider a flip taking $\alpha$ to $\alpha'$

- **breakcount**: number of clauses satisfied in $\alpha$, but not in $\alpha'$
- **makecount**: number of clauses unsatisfied in $\alpha$, but satisfied in $\alpha'$
- **diffscore**: number of unsatisfied clauses in $\alpha$ minus number of clauses unsatisfied in $\alpha'$

Typically, **breakcount**, **makecount** and **diffscore** are updated after each flip

Question: How can we do this efficiently?
GSAT and Walksat Flip Heuristics

- GSAT: select variable with highest diffscore
- Walksat:
  - First randomly select unsatisfied clause $C$
  - If there is a variable with breakcount 0 in $C$, select it
  - otherwise with probability $p$ select a random variable from $C$, and with probability $1 - p$ a variable with minimal breakcount from $C$
## Runtime Comparison Walksat vs. GSAT

<table>
<thead>
<tr>
<th>formula</th>
<th>id</th>
<th>vars</th>
<th>clauses</th>
<th>DP time</th>
<th>GSAT+w time</th>
<th>WSAT time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2bitadd.12</td>
<td>708</td>
<td>1702</td>
<td>*</td>
<td>0.081</td>
<td>0.013</td>
<td></td>
</tr>
<tr>
<td>2bitadd.11</td>
<td>649</td>
<td>1562</td>
<td>*</td>
<td>0.058</td>
<td>0.014</td>
<td></td>
</tr>
<tr>
<td>3bitadd.32</td>
<td>8704</td>
<td>32316</td>
<td>*</td>
<td>94.1</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>3bitadd.31</td>
<td>8432</td>
<td>31310</td>
<td>*</td>
<td>456.6</td>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td>2bitcomp.12</td>
<td>300</td>
<td>730</td>
<td>23096</td>
<td>0.009</td>
<td>0.002</td>
<td></td>
</tr>
<tr>
<td>2bitcomp.5</td>
<td>125</td>
<td>310</td>
<td>1.4</td>
<td>0.009</td>
<td>0.001</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Comparing an efficient complete method (DP) with local search strategies on circuit synthesis problems. (Timings in seconds.)

<table>
<thead>
<tr>
<th>formula</th>
<th>id</th>
<th>vars</th>
<th>clauses</th>
<th>DP time</th>
<th>GSAT+w time</th>
<th>WSAT time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssa7552-038</td>
<td>1501</td>
<td>3575</td>
<td>7</td>
<td>129</td>
<td>2.3</td>
<td></td>
</tr>
<tr>
<td>ssa7552-158</td>
<td>1363</td>
<td>3034</td>
<td>*</td>
<td>90</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>ssa7552-159</td>
<td>1363</td>
<td>3032</td>
<td>*</td>
<td>14</td>
<td>0.8</td>
<td></td>
</tr>
<tr>
<td>ssa7552-160</td>
<td>1391</td>
<td>3126</td>
<td>*</td>
<td>18</td>
<td>1.5</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Comparing DP with local search strategies on circuit diagnosis problems by Larrabee (1989). (Timings in seconds.)

[Source: Selman, Kautz, Cohen Local Search Strategies for Satisfiability Testing, 1993]
Are you touched by SAT?

**Quran Karim**

*Those who swallow usury will not stand [on the day of resurrection] except like the standing of a person touched by SAT.*


URL http://dl.acm.org/citation.cfm?id=1867135.1867203