SAT-Solving in a Nutshell

Prof. Dr. Erika Ábrahám

Theory of Hybrid Systems
Informatik 2

WS 09/10
Overview

1. Propositional logic, theories, normal forms
2. Propositional SAT-solving
3. SMT-solving
Overview

1. Propositional logic, theories, normal forms
2. Propositional SAT-solving
3. SMT-solving
Abstract grammar:

\[ \varphi := \text{prop} \mid (\neg \varphi) \mid (\varphi \land \varphi) \]

with prop \(\in\) Prop.

Syntactic sugar:

\[ \bot := (a \land \neg a) \]
\[ \top := (a \lor \neg a) \]
\[ (\varphi_1 \lor \varphi_2) := \neg((\neg \varphi_1) \land (\neg \varphi_2)) \]
\[ (\varphi_1 \rightarrow \varphi_2) := ((\neg \varphi_1) \lor \varphi_2) \]
\[ (\varphi_1 \leftrightarrow \varphi_2) := ((\varphi_1 \rightarrow \varphi_2) \land (\varphi_2 \rightarrow \varphi_1)) \]
\[ (\varphi_1 \oplus \varphi_2) := (\varphi_1 \leftrightarrow (\neg \varphi_2)) \]
Propositional logic: Semantics

$\models \subseteq (2^{\text{Prop}} \times \text{Formula})$ is defined recursively:

\[
\begin{align*}
\alpha \models p & \quad \text{iff } \alpha(p) = \text{true} \\
\alpha \models \neg \varphi & \quad \text{iff } \alpha \not\models \varphi \\
\alpha \models \varphi_1 \land \varphi_2 & \quad \text{iff } \alpha \models \varphi_1 \text{ and } \alpha \models \varphi_2 \\
\alpha \models \varphi_1 \lor \varphi_2 & \quad \text{iff } \alpha \models \varphi_1 \text{ or } \alpha \models \varphi_2 \\
\alpha \models \varphi_1 \rightarrow \varphi_2 & \quad \text{iff } \alpha \models \varphi_1 \text{ implies } \alpha \models \varphi_2 \\
\alpha \models \varphi_1 \leftrightarrow \varphi_2 & \quad \text{iff } \alpha \models \varphi_2 \text{ iff } \alpha \models \varphi_2
\end{align*}
\]
<table>
<thead>
<tr>
<th>Theory</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propositional logic</td>
<td>$x \land (y \lor z)$</td>
</tr>
<tr>
<td>Equality</td>
<td>$(x = y \land \neg(y = z)) \rightarrow \neg(x = z)$</td>
</tr>
<tr>
<td>Linear arithmetic</td>
<td>$(2x + 3y \leq 5) \lor (x + 5y - 10z \geq 6)$</td>
</tr>
<tr>
<td>Bitvectors</td>
<td>$((a &gt;&gt; b) &amp; c) &lt; c$</td>
</tr>
<tr>
<td>Arrays</td>
<td>$(i = j \land a[j] = 1) \rightarrow a[i] = 1$</td>
</tr>
<tr>
<td>Pointer</td>
<td>$p = q \land *p = 5 \rightarrow *q = 5$</td>
</tr>
<tr>
<td>Combined theories</td>
<td>$(i \leq j \land a[j] = 1) \rightarrow a[i] &lt; 2$</td>
</tr>
</tbody>
</table>
Normal forms

Input for solvers:

- Negation Normal Form (NNF)
- Conjunctive Normal Form (CNF)
Consider the formula

\[ \phi = (a \rightarrow (b \land c)) \]

The Parse Tree:

- Associate a new auxiliary variable with each gate.
- Add constraints that define these new variables.
- Finally, enforce the root node.
Converting to CNF: Tseitin’s encoding

Need to satisfy:

\[(h_1 \leftrightarrow (a \to h_2)) \land (h_2 \leftrightarrow (b \land c)) \land (h_1)\]

Each gate encoding has a CNF representation with 3 or 4 clauses.
Overview

1. Propositional logic, theories, normal forms
2. Propositional SAT-solving
3. SMT-solving
A Basic SAT algorithm

While (true)
{
    if (!Decide()) return (SAT);
    while (!BCP())
        if (!Resolve_Conflict()) return (UNSAT);
}

Choose the next variable and value. Return false if all variables are assigned.

Boolean Constraint Propagation. Return false if all variables are assigned.

Conflict resolution and backtracking. Return False if impossible.
Assume the CNF formula
\[ \phi : (x \lor y \lor z) \land (\neg x \lor y) \land (\neg y \lor z) \land (\neg x \lor \neg y \lor \neg z) \]
SAT-solving: Components

- Decision
- Boolean Constraint Propagation
- Conflict resolution
- Backtracking
Boolean Constraint Propagation

- A clause can be
  - Satisfied: at least one literal is true
  - Unsatisfied: all literals are false → Conflict
  - Unit: one literal is unassigned, the remaining literals are false → Propagation
  - Unresolved: all other cases

- Example: \( C = (x_1 \lor x_2 \lor x_3) \)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>( x_2 )</td>
<td>( x_3 )</td>
<td></td>
<td>( C )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td>satisfied</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>unsatisfied</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>unit</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>unresolved</td>
</tr>
</tbody>
</table>
Organize the search in the form of a decision tree

- Each node corresponds to a decision
- Definition: Decision Level (DL) is the depth of the node in the decision tree.
- Notation: $x = v @ d$
  - $x \in \{0, 1\}$ is assigned to $v$ at the decision level $d$
Conflict resolution

\[ x_1 = 1@6 \]
\[ x_2 = 1@6 \]
\[ x_3 = 1@6 \]
\[ x_4 = 1@6 \]
\[ x_5 = 1@6 \]
\[ x_6 = 1@6 \]
\[ x_9 = 0@1 \]
\[ x_{10} = 0@3 \]
\[ x_{11} = 0@3 \]
Conflict resolution

The **resolution** inference rule for CNF:

$$
\frac{(l \lor l_1 \lor l_2 \lor \ldots \lor l_n) \quad (\neg l \lor l'_1 \lor \ldots \lor l'_m)}{(l_1 \lor \ldots \lor l_n \lor l'_1 \lor \ldots \lor l'_m)} \quad \text{Resolution}
$$

Example:

$$
\frac{(a \lor b) \quad (\neg a \lor c)}{(b \lor c)}
$$

- Resolution is a **sound and complete** inference system for CNF.
- If the input formula is unsatisfiable, there exists a proof of the empty clause.
Conflict resolution

Apply resolution up in the implication tree until a UIP (Unique Implication Point) has been reached:

1. \((x_{10} \lor \neg x_1 \lor x_9 \lor x_{11})\)
2. \((x_{10} \lor \neg x_4 \lor x_{11})\)
3. \((x_{10} \lor \neg x_2 \lor \neg x_3 \lor x_{11})\)
   ...
   ...

Prof. Dr. Erika Ábrahám - SAT-Solving in a Nutshell
- Backtrack to the largest decision level in the conflict clause.
- This resolves the conflict and triggers an implication by the new conflict clause.
Progress of a SAT solver

work invested in refuting \( x = 1 \)

\( x = 1 \)

Decision Level

Decision

Conflict

Refutation of \( x = 1 \)
VSIDS (Variable State Independent Decaying Sum)

1. Each variable (in each polarity) has an activity initialized to 0.
2. When resolution gets applied to a clause, the activities of its literals are increased.
3. Decision: The unassigned variable with the highest activity is chosen.
4. Periodically, all the activities are divided by a constant.
The SAT competitions

Taken from http://baldur.iti.uka.de/sat-race-2008/analysis.html
Overview

1. Propositional logic, theories, normal forms
2. Propositional SAT-solving
3. SMT-solving
Lazy satisfiability checking

\[ \phi \]

(Boolean abstraction)

SAT-solver

unsatisfiable

UNSAT

satisfiable

SAT

(In)equation set

satisfiable

Theory solver

Explanation

unsatisfiable

satisfiable

Prof. Dr. Erika Ábrahám - SAT-Solving in a Nutshell