Experiences in the Research and Development of a Non-clausal SAT Solver

John Franco, Sean Weaver

ECECS, University of Cincinnati
United States Department of Defense
MODULE module-name-changed
INPUT
   ID_EX_RegWrite, ID_EX_MemToReg, _Taken_Branch_1_1, EX_MEM_Jump, ...
OUTPUT _temp_1252;
STRUCTURE
   _squash_1_1 = or(_Taken_Branch_1_1, EX_MEM_Jump);
   _squash_bar_1_1 = not(_squash_1_1);
   _EX_Jump_1_1 = and(_squash_bar_1_1, ID_EX_Jump);
   _Taken_Branch_9_1 = and(_squash_bar_1_1, ID_EX_Branch, TakeBranchALU_0);
   _Reg2Used_1_1 = or(IF_ID_UseData2, IF_ID_Branch, IF_ID_MemWrite, IF_ID_MemToReg);
   _temp_967 = and(_Reg2Used_1_1, e_2_1);
   ...
   _temp_976 = ite(_temp_969, IF_ID_Jump, Jump_0);
   ...
   _temp_1249 = and(_temp_1038, _temp_1066, _temp_1072, _temp_1189, _temp_1246);
   ...
true_value = new_int_leaf(1);
are_equal(_temp_1252, true_value); % 1
ENDMODULE

Translation to CNF

Expression: \( v_3 = \text{ite}(v_0, v_1, v_2); \)

Karnaugh Map:

<table>
<thead>
<tr>
<th></th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

CNF:

\[
(v_0 \lor v_2 \lor \bar{v}_3) \\
(v_0 \lor \bar{v}_2 \lor v_3) \\
(\bar{v}_0 \lor \bar{v}_1 \lor v_3) \\
(\bar{v}_0 \lor v_1 \lor \bar{v}_3)
\]
Translation to CNF

Expression: $v_3 = \text{ite}(v_0, v_1, v_2)$;  
What Heuristics Like:

<table>
<thead>
<tr>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Karnaugh Map:

CNF:

$$(v_0 \lor v_2 \lor \bar{v}_3)$$

$$(v_0 \lor \bar{v}_2 \lor v_3)$$

$$(\bar{v}_0 \lor \bar{v}_1 \lor v_3)$$

$$(\bar{v}_0 \lor v_1 \lor \bar{v}_3)$$
States of a SMURF representing $v_3 = \text{ite}(v_0, v_1, v_2)$
SMURFS Don’t Have to be Big

\[ \text{ite}(v_1, v_2 \land (v_3 \oplus v_4), v_4 \land (v_2 \oplus v_3)) \]

\[ \text{ite}(v_1, v_3 \oplus v_4, v_4 \land \bar{v}_3) \]

\[ \text{ite}(v_1, v_2 \land \bar{v}_3, v_2 \oplus v_3) \]

\[ \text{ite}(v_1, v_2 \land \bar{v}_2, \bar{v}_2 \land x_4) \]
Locally Skewed, Globally Balanced Heuristics

Weight of terminal state is 0

If state $s$ has $p$ successors $\{s_1, s_2, \ldots, s_p\}$, weight of $s$ is

$$\frac{\sum_{i=1}^{p} \text{weightOf}(s_i) + \text{numberInferencesMadeEnrouteTo}(s_i)}{K \cdot p}$$

Every state transition gets a weight:

- # inferences to destination state plus weight of that state

Every literal gets a score:

- Sum of transition weights for that literal across SMURFs

Every variable $v$ gets a score:

$$(\text{score}(v) + \epsilon) \times (\text{score}(\overline{v}) + \epsilon)$$

Branch on highest scoring variable
Locally Skewed, Globally Balanced Heuristics

\[ \text{ite}(v_1, v_2 \land (v_3 \oplus v_4), v_4 \land (v_2 \oplus v_3)) \]

\[ \langle 0 \rangle \]

\[ \langle (10 + (8/3)(1/K) + 1/(3K^2))/(8K) \rangle \]
Require Pre-processing

Intuitively, we desire:

- Fewer State Machines
- Smaller State Machines
- Redundancies Removed Across State Machines
- Inferences Revealed and Assigned as Early as Possible
- Safe Assignments Revealed and Assigned Early
Search Space Profile for Hard Problems

![Graph showing the relationship between search depth and breadth.](image-url)
Hard Problems That Fit This Profile

For example, Bounded Model Checking problems
Tools for Pre-processing

- Existential Quantification (eliminate variables)
- Restriction (eliminate redundancies, find inferences)
- Strengthening (find inferences missed by restriction)
- Generalized co-factor (eliminate functions)
- Cluster some functions (conjoin them)
- Assign uninferred but safe values (reductions)
- Add uninferred and unsafe constraints (tunnel)
Existential Quantification

Replace $f$ with $(f|_{v=0} \lor f|_{v=1})$

$\implies$

Replace $f$ with $(f|_{v=0} \lor f|_{v=1})$
Existential Quantification

Replace $f$ with $(f|_{v=0} \lor f|_{v=1})$
Choose \( v_7 \) to separate \( v_1, v_2, v_3 \) from \( v_4, v_5, v_6 \)
Existential Quantification
Restrict
Spreading an inference from one function to another.

If $v_2 = 0$ in $f$ then $v_3 = v_4 = 0$ is inferred.

Replacing $f$ with $f'$, gives inference $v_4 = 0$ from $c$ (if $v_2 = 0$) and then inference $v_3 = 0$ from $f'$. 
Strengthening

Existentially quantify away $v_1$ from $f$, then ...
Strengthening

Existentially quantify away $v_1$ from $f$, then ...

Conjoin $f$ and $c$ to reveal inference $v_3 = 1$. 
Splitting An Expression Is Desirable

\[ f_1 \land f_2 \land \ldots \land f_i \land f_{i+1} \land \ldots \land f_{m-1} \land f_m \]

If no variables in \( f_1 \) to \( f_i \) are in \( f_{i+1} \) to \( f_m \) and no variables in \( f_{i+1} \) to \( f_m \) are in \( f_1 \) to \( f_i \), then \( \phi_1 = f_1 \land \ldots \land f_i \) and \( \phi_2 = f_{i+1} \land \ldots \land f_m \) can be solved independently.
Autark Assignments Come Close

\[ f_1 \land f_2 \land \ldots \land f_i \land f_{i+1} \land \ldots \land f_{m-1} \land f_m \]

If there is a subset \( V' \) of variables in \( f_1 \) to \( f_i \) and an assignment \( t_{V'} \) of values to \( V' \) that satisfies \( f_1 \) to \( f_i \) and none of the variables of \( V' \) is in \( f_{i+1} \) to \( f_m \) then satisfy \( \phi_1 = f_1 \land \ldots \land f_i \) with partial assignment \( t_{V'} \) and solve \( \phi_2 = f_{i+1} \land \ldots \land f_m \) independently.
Safe Assignments

If \((f_1 \land \ldots \land f_m)|_{v=0} \equiv (f_1 \land \ldots \land f_m)|_{v=0} \lor (f_1 \land \ldots \land f_m)|_{v=1}\)

then \(v = 0\) is safe

If \((f_1 \land \ldots \land f_m)|_{v=1} \equiv (f_1 \land \ldots \land f_m)|_{v=0} \lor (f_1 \land \ldots \land f_m)|_{v=1}\)

then \(v = 1\) is safe

Example: \(v = 1\) is a safe assignment but is not autark:

\[f = (v \lor a) \land (v \lor \bar{a} \lor b) \land (\bar{v} \lor \bar{a} \lor b \lor c) \land (\bar{b} \lor \bar{c}) \ldots\]

Because

\[f|_{v=0} = (a) \land (\bar{a} \lor b) \land (\bar{b} \lor \bar{c}) \rightarrow\]

\[f|_{v=1} = (\bar{a} \lor b \lor c) \land (\bar{b} \lor \bar{c})\]
Safe Assignments: Checking isn’t too bad

But conjoining functions to find safe assignments can be expensive

   Luckily, the computational effort can be distributed:
   
   If \((f_i|_v \land f_i|_{\bar{v}}) \equiv 0\) for every \(i\) such that \(v\) is in \(f_i\) then \(v = 1\) is safe
   
   If \((f_i|_v \land \overline{f_i|_{\bar{v}}}) \equiv 0\) for every \(i\) such that \(v\) is in \(f_i\) then \(v = 0\) is safe

   But the check may fail on some safe assignments

   The idea extends to groups of variables
Consider two functions $f_1$ and $f_2$.
Finding safe assignment for $v_4$ in $f_1$ alone:
Finding safe assignment for $v_4$ in $f_1$ alone: $v_4 = 0$ only
Finding safe assignment for $v_4$ in $f_2$ alone:
Finding safe assignment for $v_4$ in $f_2$ alone: neither one
Conjoin the two functions $f_1 \land f_2$: 
Safe Assignments: Some Are Missed

Conjoin the two functions then find $v_4 = 0$ is safe
Safe Assignments: Some Are Missed

Conjoin the two functions then find $v_4 = 0$ is safe, also $v_2 = 0$ is safe.
Safe assignments $v_1 = v_2 = 1$ and $v_1 = 1, v_3 = 0$ are found as pairs only
Unsafe Assignments

- Guess some (uninferred) constraints based on solution structure in the same family
- Add those constraints initially to reduce the “hump”
- Run the search breadth-first
- When search breadth begins to decline, remove the constraints
- Solve to completion
- Possibly no solution found for a satisfiable input
Example: Van der Waerden Numbers

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

10100011101001101101000111010

2 categories, progression length 4 ($W_{max}(2, 4)$ formula)
Example: Van der Waerden Numbers

2 categories, progression length 4 \((W_{max}(2, 4) \text{ formula})\)

2 categories, progression length 5 \((W_{max}(2, 5) \text{ formula})\)
Conjecture:

For every $W_{max}(2, l)$ formula there exists a solution that contains at least one reflected pattern of length $W(2, l)/(2 \ast (l - 1))$ with the middle positioned somewhere between $W(2, l)/(l - 1)$ and $W(2, l) \ast (l - 2)/(l - 1)$. 
Conjecture:

For every $W_{max}(2, l)$ formula there exists a solution that contains at least one reflected pattern of length $W(2, l)/(2 \ast (l - 1))$ with the middle positioned somewhere between $W(2, l)/(l - 1)$ and $W(2, l) \ast (l - 2)/(l - 1)$.

From search profile:

The maximum breadth occurs near depth $W(2, l)/(2(l - 1))$.

$W(2, l) \approx l \ast W(2, l - 1)$, for small $l$ anyway.
Add the unsafe constraints:

<table>
<thead>
<tr>
<th>Function Seg</th>
<th>Range</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>((v_{-i} \lor v_{i+1}) \land (\bar{v}<em>{-i} \lor \bar{v}</em>{i+1}))</td>
<td>(0 \leq i &lt; s/2)</td>
<td>force (v_{-i} \equiv \bar{v}_{i+1}).</td>
</tr>
</tbody>
</table>
Add the unsafe constraints:

<table>
<thead>
<tr>
<th>Function Seg</th>
<th>Range</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>((v_i \lor v_{i+1}) \land (\bar{v}<em>i \lor \bar{v}</em>{i+1}))</td>
<td>(0 \leq i &lt; s/2)</td>
<td>force (v_i \equiv \bar{v}_{i+1}).</td>
</tr>
</tbody>
</table>

Retract the constraints at depth

\((l \times W(2, l - 1))/(2(l - 1))\)
Add the unsafe constraints:

<table>
<thead>
<tr>
<th>Function Seg</th>
<th>Range</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>((v_i \lor v_{i+1}) \land (\bar{v}<em>i \lor \bar{v}</em>{i+1}))</td>
<td>(0 \leq i &lt; s/2)</td>
<td>force (v_i \equiv \bar{v}_{i+1}).</td>
</tr>
</tbody>
</table>

Retract the constraints at depth

\[
(l \ast W(2, l - 1))/(2(l - 1))
\]

Continue without unsafe constraints until the end