### The Overall Picture

<table>
<thead>
<tr>
<th>Application</th>
<th>System + Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>Algorithm + Implementation</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Calculus + Strategy</td>
</tr>
<tr>
<td>Calculus</td>
<td>Logic + States + Rules</td>
</tr>
<tr>
<td>Logic</td>
<td>Syntax + Semantics</td>
</tr>
</tbody>
</table>
1 Algorithm: CDCL($S$)

Input : An initial state ($\epsilon ; N ; \emptyset ; 0 ; \top$).
Output: A final state $S = (M ; N ; U ; k ; \top)$ or $S = (M ; N ; U ; k ; \bot)$

2 while (any rule applicable) do

3 if rule (Conflict($S$)) then

4     while (Skip($S$) || Resolve($S$)) do

5         update VSIDS on resolved literals;

6     update VSIDS on learned clause, Backtrack($S$);

7     if (forget heuristic) then

8         Forget($S$), Restart($S$);

9     else

10    else

11    if (restart heuristic) then

12       Restart($S$);

13 else

14 if rule (! Propagate($S$)) then

15       Decide($S$) literal with max. VSIDS score;
Implementation: Data Structures

**Propagate** \((M; N; U; k; \top) \Rightarrow_{\text{CDCL}} (ML^{C\lor L}; N; U; k; \top)\) provided \(C \lor L \in (N \cup U)\), \(M \models \lnot C\), and \(L\) is undefined in \(M\)

**Conflict** \((M; N; U; k; \top) \Rightarrow_{\text{CDCL}} (M; N; U; k; D)\) provided \(D \in (N \cup U)\) and \(M \models \lnot D\)
Implementation

- data structures: clauses, trail, and the rules
- heuristics: decision literal, forget, restart
- space efficiency: forget
- quality: restarts
- special cases
Data Structures

Idea: Select two literals from each clause for indexing.

2.10.1 Invariant (2-Watched Literal Indexing)

If one of the watched literals is false and the other watched literal is not true, then all other literals of the clause are false.
\[ N = \{ P \lor \neg R, P \lor \neg Q, R \lor Q \lor P, \neg P \lor R \lor Q \} \]
VSIDS: Variable State Independent Decaying Sum

- each propositional variable has a positive score, initially 0
- decide the variable with maximal score, remember sign (phase saving)
- increment the score of variables involved in resolution by $b$
- increment the score of variables in learned clauses by $b$
- initially $b > 0$
- at Backtrack set $b := c \times b$ where $2 >> c > 1$, i.e., $b_n = c^n \times b$
- take care of overflows, i.e., rescale from time to time
- sometimes pick a variable randomly
Forget

- fix a limit \( d \) on the number of learned clauses
- if more than \( |U| > d \) start forgetting
- remove redundant clauses
- sort the learned clauses according to a score
- typical elements of the score are clause length, the VSIDS score, dependency on decisions
- remove the \( k\% \) clauses with minimal score from \( U \)
- \( d := d + e \) for some \( e, e >> 1 \)
- do a Restart
Restart

- after forgetting do a restart
- if a unit is learned do a restart
- restart often at the beginning of a run
- classics: Luby sequence 1, 1, 2, 1, 1, 2, 4, …

\[(u_1, v_1) := (1, 1), \]
\[(u_{n+1}, v_{n+1}) := ((u_n & - u_n) = v_n? (u_n + 1, 1) : (u_n, 2 * v_n))\]
Memory Matters: SPASS-SATT

<table>
<thead>
<tr>
<th></th>
<th>Value 1</th>
<th>Value 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forget-Start</td>
<td>800</td>
<td>108800</td>
</tr>
<tr>
<td>Restarts</td>
<td>412</td>
<td>369</td>
</tr>
<tr>
<td>Conflicts</td>
<td>153640</td>
<td>133403</td>
</tr>
<tr>
<td>Decisions</td>
<td>184034</td>
<td>159005</td>
</tr>
<tr>
<td>Propagations</td>
<td>17770298</td>
<td>15544812</td>
</tr>
<tr>
<td>Time</td>
<td>11</td>
<td>23</td>
</tr>
<tr>
<td>Memory</td>
<td>16</td>
<td>41</td>
</tr>
</tbody>
</table>