Consistency Checking of All Different Constraints over Bit-Vectors within a SAT Solver

Armin Biere, Robert Brummayer
Institute for Formal Models and Verification
Johannes Kepler University
Linz, Austria

FMCAD’08
Formal Methods in Computer Aided Design
Portland, Oregon, USA

Thursday, November 20, 2008
Simple Path Constraints

- bounded model checking: \([\text{BiereCimattiClarkeZhu'99}]\)

\[
I(s_1) \land T(s_1, s_2) \land \ldots \land T(s_{k-1}, s_k) \land \bigvee_{0 \leq i \leq k} B(s_i) \quad \text{satisfiable?}
\]

- reoccurrence diameter checking: \([\text{BiereCimattiClarkeZhu'99}]\)

\[
T(s_1, s_2) \land \ldots \land T(s_{k-1}, s_k) \land \bigwedge_{1 \leq i < j \leq k} s_i \neq s_j \quad \text{unsatisfiable?}
\]

- \(k\)-induction base case: \([\text{SheeranSinghStålmarck'00}]\)

\[
I(s_1) \land T(s_1, s_2) \land \ldots \land T(s_{k-1}, s_k) \land B(s_k) \land \bigwedge_{0 \leq i < k} \neg B(s_i) \quad \text{satisfiable?}
\]

- \(k\)-induction induction step: \([\text{SheeranSinghStålmarck'00}]\)

\[
T(s_1, s_2) \land \ldots \land T(s_{k-1}, s_k) \land B(s_k) \land \bigwedge_{0 \leq i < k} \neg B(s_i) \land \bigwedge_{1 \leq i < j \leq k} s_i \neq s_j \quad \text{unsatisfiable?}
\]
All Different Constraints (ADC)

- classical concept in constraint programming:
  - $k$ variables over a domain of size $m$ supposed to have different values
  - for instance $k$-queen problem

- propagation algorithms to establish arc-consistency
  - explicit propagators: [Régis’n’94]
    * $O(k \cdot m)$ space
    * $O(k^2 \cdot m^2)$ time
  - symbolic propagators: [GentNightingale’n’04] also [MarquesSilvaLynce’n’07]
    * one-hot CNF encoding with $\Omega(k \cdot m)$ boolean variables

- in model checking $k << m$ typically $k < 1000 \quad m = 2^n > 2^{100}$ n latches
Symbolic ADCs for Large Domains

- encoding bit-vector inequalities directly:
  - let $u, v$ be two $n$-bit vectors, $d_0, \ldots, d_{n-1}$ fresh boolean variables
    
    $u \neq v$ is equisatisfiable to $(d_0 \lor \cdots \lor d_{n-1}) \land \bigwedge_{j=0}^{n-1} (u_j \lor v_j \lor \overline{d_j}) \land (\overline{u}_j \lor \overline{v}_j \lor \overline{d_j})$

  - can be extended to encode Ackermann Constraints + McCarthy Axioms

  - either eagerly encode all $s_i \neq s_j$ quadratic in $k$

  - or refine adding bit-vector inequalities on demand [EénSörensson’03]

- natively handle ADCs within SAT solver: our main contribution

  - similar to theory consistency checking in lazy SMT vs. “lemmas on demand”

  - can be extended to also perform theory propagation

- sorting networks ineffective in our experience [KröningStrichman’03,JussilaBiere’06]
All Different Objects (ADOs)

- ADO for $v$
- ADO for $w$
- ADO for $u$

Consistency Checking of ADC(BV) – FMCAD’08 Armin Biere – FMV – JKU Linz
All Different Objects (ADOs)

ADO for $u$

assign

ADO for $v$

ADO for $w$

hash
All Different Objects (ADOs)

ADO for \( u \)

\[
\begin{array}{c|c|c}
0 & u_1 & 1 \\
\end{array}
\]

ADO for \( v \)

\[
\begin{array}{c|c|c}
v_2 & v_1 & v_0 \\
\end{array}
\]

ADO for \( w \)

\[
\begin{array}{c|c|c}
w_2 & w_1 & w_0 \\
\end{array}
\]

hash
All Different Objects (ADOs)

hash

move

ADO for $u$

0 $u_1$ 1

ADO for $v$

$v_2$ $v_1$ $v_0$

ADO for $w$

$w_2$ $w_1$ $w_0$
All Different Objects (ADOs)

ADO for $u$

ADO for $v$

ADO for $w$
All Different Objects (ADOs)

ADO for $u$

ADO for $v$

ADO for $w$

hash
All Different Objects (ADOs)

Consistency Checking of ADC(BV) – FMCAD’08

Armin Biere – FMV – JKU Linz
Consistency Checking of ADC(BV) – FMCAD’08  
Armin Biere – FMV – JKU Linz
All Different Objects (ADOs)

hash

ADO for $u$

ADO for $v$

ADO for $w$

Lookup

Consistency Checking of ADC(BV) – FMCAD’08 Armin Biere – FMV – JKU Linz
All Different Objects (ADOs)

Consistency Checking of ADC(BV) – FMCAD’08

ADO for $u$

ADO for $v$

ADO for $w$

conflict $u = v$

hash
Implementaiton Details

- **ADO key is calculated from concrete bit-vector**
  - by for instance XOR’ing bits word by word

- **ADOs watched by variables (not literals)**
  - during backtracking all inserted ADOs need to be removed from hash table
  - save whether variable assignment forced ADO to be inserted
  - stack like insert/remove operations on hash table allow open addressing

- **conflict analysis**
  - all bits of the bit-vectors in conflict are followed
  - can be implemented by temporarily generating a pseudo clause

\[
(u_2 \lor \overline{u}_1 \lor \overline{u}_0 \lor v_2 \lor \overline{v}_1 \lor \overline{v}_0)
\]
<table>
<thead>
<tr>
<th></th>
<th>solved complete</th>
<th>inconclusive unsolved</th>
<th>unsatisfiable satisfiable</th>
<th>time $10^3$sec</th>
<th>space GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>mixed</td>
<td>y</td>
<td>259</td>
<td>85</td>
<td>38</td>
<td>182</td>
</tr>
<tr>
<td>refine</td>
<td>y</td>
<td>250</td>
<td>94</td>
<td>32</td>
<td>179</td>
</tr>
<tr>
<td>sadc</td>
<td>y</td>
<td>244</td>
<td>100</td>
<td>36</td>
<td>171</td>
</tr>
<tr>
<td>eager</td>
<td>y</td>
<td>242</td>
<td>102</td>
<td>27</td>
<td>177</td>
</tr>
<tr>
<td>none</td>
<td>n</td>
<td>267</td>
<td>77</td>
<td>56</td>
<td>179</td>
</tr>
<tr>
<td>base</td>
<td>n</td>
<td>283</td>
<td>61</td>
<td>96</td>
<td>187</td>
</tr>
</tbody>
</table>

only checked up to $k = 100$ (at most 100 steps per instance)

three possible outcomes: inconclusive, satisfiable, or unsatisfiable
Symbolic ADCs versus Refine

Consistency Checking of ADC(BV) – FMCAD’08 Armin Biere – FMV – JKU Linz
Mixed Approach versus Refine Only

Consistency Checking of ADC(BV) – FMCAD’08 Armin Biere – FMV – JKU Linz
• symbolic consistency checker for ADCs over bit-vectors
  – successfully applied to simple path constraints in model checking
  – similar to theory consistency checking in lazy SMT solvers
  – combination with eager refinement approach lemmas on demand

• future work: ADC based BCP for bit-vectors
  – aka theory propagation in lazy SMT solvers
  – extensions to handle Ackermann constraints or even McCarthy axioms
  – one-way to get away from pure bit-blasting in BV