Lemmas on Demand for the Extensional Theory of Arrays

Robert Brummayer and Armin Biere

Institute for Formal Models and Verification
Johannes Kepler University Linz, Austria

SMT 2008
Princeton, New Jersey, USA

July 8th, 2008
Overview

- Introduction
- Arrays
- Verification example
- Lemmas on demand
- Consistency checking
- Lemma encoding
- Handling equalities on arrays
- Conclusion
Introduction

R. Brummayer, FMV, JKU Linz

- Bit-Vectors
  - Bit-precise modelling of computation
  - Modular arithmetic

- Arrays
  - Modelling of memories
  - Main memory in software
  - Memory components in hardware, e.g. registers

Combination of both: bit-precise modelling and reasoning
Scenarios

- Pipelined vs. non-pipelined hardware
- Equivalence-check memory semantics of software

Let both algorithm start on the same array

Symbolically execute both algorithms

Finally, check whether the resulting arrays are equal or not
• Theory of Arrays [McCarthy62]

\[(A1) \quad a = b \land i = j \implies \text{read}(a, i) = \text{read}(b, j)\]

\[(A2) \quad i = j \implies \text{read}(\text{write}(a, i, e), j) = e\]

\[(A3) \quad i \neq j \implies \text{read}(\text{write}(a, i, e), j) = \text{read}(a, j)\]

• With (A1) to (A3) we cannot handle array inequalities

• We need additional axiom of extensionality (A4) resp. (A4′)

\[(A4) \quad a = b \iff \forall i \,(\text{read}(a, i) = \text{read}(b, i))\]

\[(A4′) \quad a \neq b \implies \exists \lambda \,(\text{read}(a, \lambda) \neq \text{read}(b, \lambda))\]
Verification of Selection Sort for bit-width = 32 and size = 4

R. Brummayer, FMV, JKU Linz
procedure lemmas-on-demand ($\phi$)

encode-to-sat ($\phi$)

loop

$(r, \sigma) \leftarrow \text{sat}(\phi)$

if ($r = \text{unsatisfiable}$) return unsatisfiable

if (consistent ($\phi, \sigma$)) return satisfiable

add-lemma ($\phi, \sigma$)

• Do not eagerly encode all constraints up front

• Explicitly check constraints that have not been encoded
  – Use SAT solver assignment $\sigma$

• Incrementally refine formula
• Translate bit-vector but not array part of the formula

• Let SAT solver “guess” solution
  – If SAT solver cannot find a solution, terminate with \textit{unsatisfiable}

• Explicitly check Array axioms A1 to A3

• If check succeeds then terminate with \textit{satisfiable}

• If check fails
  – Add lemma to refine formula
  – Let SAT solver “guess” a new solution
Checking Array Axioms

- Propagation-based algorithm
  - Direct application of (A1) to (A3)

- Annotate every array expression with its set of reads $\rho$

- For every read $\text{read}(b, i) \in \rho(\text{write}(a, j, e))$
  - (A2): If current assignment $\sigma(i) = \sigma(j)$, check if $\sigma(\text{read}(b, i)) = \sigma(e)$
  - (A3): If current assignment $\sigma(i) \neq \sigma(j)$, add read to $\rho(a)$

- Check read congruence (A1) on all array expressions

- Propagation only downwards and can be implemented with DFS or BFS
Example 1: Inconsistent Assignment

\[ \sigma(i) = \sigma(j), \quad \sigma(k) = \sigma(l2) \]

\[ \sigma(r1) \neq \sigma(r2), \quad \sigma(i) \neq \sigma(l1), \quad \sigma(j) \neq \sigma(l2), \quad \sigma(r3) \neq \sigma(e2) \]
Lemma Construction

- We collect all assignments responsible for propagation

- We add a lemma of the following kind:
  \[ i \neq l_1 \land j \neq l_2 \land \ldots \land i = j \implies r_1 = r_2 \]

- Lemma for inconsistency of \( r_1 \) and \( r_2 \) in example 1
  \[ i \neq l_1 \land j \neq l_2 \land i = j \implies r_1 = r_2 \]

- Lemma for inconsistency of \( r_3 \) and right write in example 1
  \[ k = l_2 \implies r_3 = e_2 \]

- Lemmas can be directly encoded into CNF (no new expressions)
• For every array equality $a = b$
  
  – Introduce fresh Tseitin variable $e_{a,b}$
  
  – Introduce two virtual reads $\text{read}(a, \lambda), \text{read}(b, \lambda)$, for a fresh $\lambda$
  
  – Virtual reads are used as witness for array inequality
  
  – Encode $\overline{e_{a,b}} \Rightarrow \text{read}(a, \lambda) \neq \text{read}(b, \lambda)$

• If $\sigma(e_{a,b}) = 1$, propagate reads over array equalities
  
  – Ensures read congruence over equal arrays
  
  – Propagation can now be cyclic, e.g. $a = b \land b = c \land c = a$
  
  – We need fix-point algorithm
• We must ensure congruence on write values for equal writes (only necessary in a setting without congruence closure)

• For example \( \text{write}(a, i, e_1) = \text{write}(a, i, e_2) \) implies that \( e_1 = e_2 \)

• We can treat every \( \text{write}(a, i, e) \) as \( \text{read}(a, i) \), where \( \text{read}(a, i) = e \)

• Propagate writes as reads

• In order to reach every array equality
  – We must also propagate upwards with respect to (A2) and (A3)
  – Only propagate upwards if value has not been overwritten
Example 2: Propagation upwards

\[
\text{write}(a, i, e_1) = \text{write}(b, j, e_2), \quad i \neq k, \quad j \neq k, \quad \text{read}(a, k) \neq \text{read}(b, k)
\]
## Extensional Experiments (1/2)

R. Brummayer, FMV, JKU Linz

<table>
<thead>
<tr>
<th>benchmark</th>
<th>S</th>
<th>P</th>
<th>R</th>
<th>T</th>
<th>M</th>
<th>(T)</th>
<th>(M)</th>
<th>(T)</th>
<th>(M)</th>
<th>(T)</th>
<th>(M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>swapmem</td>
<td>s</td>
<td>2</td>
<td>16</td>
<td>0.1</td>
<td>0.6</td>
<td>0.0</td>
<td>0.0</td>
<td>0.1</td>
<td>547.6</td>
<td>101.6</td>
<td>554.1</td>
</tr>
<tr>
<td>swapmem</td>
<td>s</td>
<td>6</td>
<td>53</td>
<td>0.4</td>
<td>1.2</td>
<td>0.6</td>
<td>3.3</td>
<td>TO</td>
<td>TO</td>
<td>90.0</td>
<td>71.9</td>
</tr>
<tr>
<td>swapmem</td>
<td>s</td>
<td>8</td>
<td>37</td>
<td>0.3</td>
<td>1.4</td>
<td>1.5</td>
<td>5.4</td>
<td>TO</td>
<td>TO</td>
<td>333.8</td>
<td>175.8</td>
</tr>
<tr>
<td>swapmem</td>
<td>s</td>
<td>10</td>
<td>34</td>
<td>0.5</td>
<td>1.7</td>
<td>1.0</td>
<td>8.3</td>
<td>TO</td>
<td>TO</td>
<td>740.2</td>
<td>305.3</td>
</tr>
<tr>
<td>swapmem</td>
<td>s</td>
<td>12</td>
<td>64</td>
<td>1.0</td>
<td>2.2</td>
<td>3.5</td>
<td>11.2</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>swapmem</td>
<td>s</td>
<td>14</td>
<td>57</td>
<td>1.6</td>
<td>2.4</td>
<td>1.8</td>
<td>15.7</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>wchains</td>
<td>s</td>
<td>6</td>
<td>14</td>
<td>0.0</td>
<td>0.0</td>
<td>0.8</td>
<td>7.0</td>
<td>46.3</td>
<td>11.6</td>
<td>8.5</td>
<td>60.1</td>
</tr>
<tr>
<td>wchains</td>
<td>s</td>
<td>8</td>
<td>16</td>
<td>0.0</td>
<td>0.0</td>
<td>1.9</td>
<td>14.5</td>
<td>572.2</td>
<td>32.5</td>
<td>23.5</td>
<td>117.6</td>
</tr>
<tr>
<td>wchains</td>
<td>s</td>
<td>10</td>
<td>18</td>
<td>0.1</td>
<td>1.3</td>
<td>3.1</td>
<td>20.3</td>
<td>TO</td>
<td>TO</td>
<td>34.2</td>
<td>193.6</td>
</tr>
<tr>
<td>wchains</td>
<td>s</td>
<td>12</td>
<td>20</td>
<td>0.1</td>
<td>1.4</td>
<td>3.5</td>
<td>29.8</td>
<td>TO</td>
<td>TO</td>
<td>65.4</td>
<td>294.6</td>
</tr>
<tr>
<td>wchains</td>
<td>s</td>
<td>15</td>
<td>21</td>
<td>0.1</td>
<td>1.5</td>
<td>4.2</td>
<td>45.8</td>
<td>TO</td>
<td>TO</td>
<td>126.3</td>
<td>472.6</td>
</tr>
<tr>
<td>wchains</td>
<td>s</td>
<td>20</td>
<td>28</td>
<td>0.3</td>
<td>2.0</td>
<td>5.7</td>
<td>80.2</td>
<td>TO</td>
<td>TO</td>
<td>267.8</td>
<td>897.3</td>
</tr>
<tr>
<td>wchains</td>
<td>s</td>
<td>30</td>
<td>36</td>
<td>0.6</td>
<td>2.8</td>
<td>10.9</td>
<td>179.9</td>
<td>TO</td>
<td>TO</td>
<td>MO</td>
<td>MO</td>
</tr>
<tr>
<td>wchains</td>
<td>s</td>
<td>60</td>
<td>70</td>
<td>2.3</td>
<td>4.8</td>
<td>41.8</td>
<td>718.5</td>
<td>TO</td>
<td>TO</td>
<td>MO</td>
<td>MO</td>
</tr>
<tr>
<td>wchains</td>
<td>s</td>
<td>90</td>
<td>96</td>
<td>4.8</td>
<td>6.8</td>
<td>MO</td>
<td>MO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>dubreva</td>
<td>u</td>
<td>2</td>
<td>19</td>
<td>0.1</td>
<td>0.6</td>
<td>0.0</td>
<td>0.0</td>
<td>0.5</td>
<td>5.0</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>dubreva</td>
<td>u</td>
<td>3</td>
<td>41</td>
<td>0.3</td>
<td>0.7</td>
<td>0.1</td>
<td>1.2</td>
<td>0.7</td>
<td>5.6</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>dubreva</td>
<td>u</td>
<td>4</td>
<td>239</td>
<td>12.5</td>
<td>2.2</td>
<td>12.5</td>
<td>6.5</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>dubreva</td>
<td>u</td>
<td>5</td>
<td>379</td>
<td>27.4</td>
<td>3.7</td>
<td>348.3</td>
<td>13.1</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>dubreva</td>
<td>u</td>
<td>6</td>
<td>1187</td>
<td>322.2</td>
<td>12.4</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>dubreva</td>
<td>u</td>
<td>7</td>
<td>1637</td>
<td>663.1</td>
<td>18.2</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>MO</td>
<td>MO</td>
</tr>
<tr>
<td>benchmark</td>
<td>S</td>
<td>P</td>
<td>R</td>
<td>T</td>
<td>M</td>
<td>T</td>
<td>M</td>
<td>T</td>
<td>M</td>
<td>T</td>
<td>M</td>
</tr>
<tr>
<td>-----------</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>swapmem</td>
<td>u</td>
<td>2</td>
<td>13</td>
<td>0.1</td>
<td>0.7</td>
<td>0.1</td>
<td>0.8</td>
<td>1.2</td>
<td>5.5</td>
<td>6.1</td>
<td>9.0</td>
</tr>
<tr>
<td>swapmem</td>
<td>u</td>
<td>4</td>
<td>25</td>
<td>1.1</td>
<td>1.0</td>
<td>3.0</td>
<td>2.0</td>
<td>5.2</td>
<td>7.1</td>
<td>159.6</td>
<td>49.8</td>
</tr>
<tr>
<td>swapmem</td>
<td>u</td>
<td>6</td>
<td>37</td>
<td>3.2</td>
<td>1.2</td>
<td>27.9</td>
<td>4.2</td>
<td>16.1</td>
<td>8.6</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>swapmem</td>
<td>u</td>
<td>8</td>
<td>49</td>
<td>9.0</td>
<td>1.5</td>
<td>129.0</td>
<td>8.3</td>
<td>20.7</td>
<td>10.1</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>swapmem</td>
<td>u</td>
<td>10</td>
<td>61</td>
<td>18.3</td>
<td>2.0</td>
<td>411.2</td>
<td>15.4</td>
<td>30.9</td>
<td>11.0</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>swapmem</td>
<td>u</td>
<td>12</td>
<td>73</td>
<td>34.7</td>
<td>2.4</td>
<td>TO</td>
<td>TO</td>
<td>62.1</td>
<td>13.3</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>swapmem</td>
<td>u</td>
<td>14</td>
<td>85</td>
<td>63.4</td>
<td>2.8</td>
<td>TO</td>
<td>TO</td>
<td>102.7</td>
<td>19.4</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>wchains</td>
<td>u</td>
<td>2</td>
<td>17</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>TO</td>
<td>TO</td>
<td>4.1</td>
<td>5.8</td>
</tr>
<tr>
<td>wchains</td>
<td>u</td>
<td>4</td>
<td>33</td>
<td>0.1</td>
<td>0.9</td>
<td>1.7</td>
<td>3.8</td>
<td>10.5</td>
<td>16.4</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>wchains</td>
<td>u</td>
<td>6</td>
<td>49</td>
<td>0.4</td>
<td>1.1</td>
<td>13.5</td>
<td>8.2</td>
<td>1.9</td>
<td>5.5</td>
<td>MO</td>
<td>MO</td>
</tr>
<tr>
<td>wchains</td>
<td>u</td>
<td>8</td>
<td>65</td>
<td>0.6</td>
<td>1.3</td>
<td>54.8</td>
<td>15.0</td>
<td>848.0</td>
<td>725.1</td>
<td>MO</td>
<td>MO</td>
</tr>
<tr>
<td>wchains</td>
<td>u</td>
<td>10</td>
<td>81</td>
<td>1.1</td>
<td>1.5</td>
<td>158.2</td>
<td>23.3</td>
<td>59.7</td>
<td>14.6</td>
<td>MO</td>
<td>MO</td>
</tr>
<tr>
<td>wchains</td>
<td>u</td>
<td>12</td>
<td>97</td>
<td>2.4</td>
<td>1.8</td>
<td>333.5</td>
<td>34.3</td>
<td>93.7</td>
<td>22.4</td>
<td>MO</td>
<td>MO</td>
</tr>
<tr>
<td>wchains</td>
<td>u</td>
<td>14</td>
<td>113</td>
<td>2.9</td>
<td>1.9</td>
<td>588.2</td>
<td>46.7</td>
<td>19.3</td>
<td>6.9</td>
<td>MO</td>
<td>MO</td>
</tr>
<tr>
<td>wchains</td>
<td>u</td>
<td>16</td>
<td>129</td>
<td>4.8</td>
<td>2.1</td>
<td>TO</td>
<td>TO</td>
<td>35.7</td>
<td>8.1</td>
<td>MO</td>
<td>MO</td>
</tr>
<tr>
<td>wchains</td>
<td>u</td>
<td>18</td>
<td>145</td>
<td>6.6</td>
<td>2.3</td>
<td>TO</td>
<td>TO</td>
<td>209.8</td>
<td>26.2</td>
<td>MO</td>
<td>MO</td>
</tr>
</tbody>
</table>
## Non-Extensional Experiments

R. Brummayer, FMV, JKU Linz

<table>
<thead>
<tr>
<th>benchmark</th>
<th>S</th>
<th>R</th>
<th>T</th>
<th>M</th>
<th>T</th>
<th>M</th>
<th>T</th>
<th>M</th>
<th>T</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>bubsort4</td>
<td>u</td>
<td>10</td>
<td>0.9</td>
<td>0.8</td>
<td>0.2</td>
<td>0.8</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>bubsort6</td>
<td>u</td>
<td>23</td>
<td>4.6</td>
<td>1.1</td>
<td>2.3</td>
<td>1.5</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>bubsort8</td>
<td>u</td>
<td>40</td>
<td>66.1</td>
<td>1.8</td>
<td>15.8</td>
<td>2.7</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>bubsort10</td>
<td>u</td>
<td>61</td>
<td>461.4</td>
<td>3.1</td>
<td>514.0</td>
<td>10.1</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>bubsort12</td>
<td>u</td>
<td>-</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>selsort2</td>
<td>u</td>
<td>8</td>
<td>0.2</td>
<td>0.6</td>
<td>0.1</td>
<td>0.7</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>selsort3</td>
<td>u</td>
<td>21</td>
<td>4.2</td>
<td>0.8</td>
<td>0.8</td>
<td>1.1</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>selsort4</td>
<td>u</td>
<td>42</td>
<td>35.6</td>
<td>1.2</td>
<td>5.4</td>
<td>1.7</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>selsort5</td>
<td>u</td>
<td>66</td>
<td>216.8</td>
<td>2.3</td>
<td>34.7</td>
<td>3.1</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>selsort6</td>
<td>u</td>
<td>-</td>
<td>TO</td>
<td>TO</td>
<td>407.2</td>
<td>7.6</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>selsort7</td>
<td>u</td>
<td>-</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>grep0084</td>
<td>s</td>
<td>191</td>
<td>31.1</td>
<td>27.4</td>
<td>23.7</td>
<td>142.7</td>
<td>81.1</td>
<td>73.2</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>grep0095</td>
<td>s</td>
<td>134</td>
<td>24.6</td>
<td>26.3</td>
<td>26.6</td>
<td>153.7</td>
<td>101.5</td>
<td>78.6</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>grep0106</td>
<td>s</td>
<td>300</td>
<td>41.6</td>
<td>30.6</td>
<td>33.5</td>
<td>164.9</td>
<td>103.2</td>
<td>88.0</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>grep0117</td>
<td>s</td>
<td>205</td>
<td>35.1</td>
<td>29.9</td>
<td>34.7</td>
<td>176.4</td>
<td>149.9</td>
<td>90.2</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>grep0777</td>
<td>s</td>
<td>474</td>
<td>481.4</td>
<td>85.1</td>
<td>MO</td>
<td>MO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>testcase15</td>
<td>s</td>
<td>0</td>
<td>99.5</td>
<td>457.9</td>
<td>98.0</td>
<td>458.8</td>
<td>32.4</td>
<td>594.6</td>
<td>MO</td>
<td>MO</td>
</tr>
<tr>
<td>testcase16</td>
<td>s</td>
<td>0</td>
<td>73.4</td>
<td>476.5</td>
<td>73.4</td>
<td>476.5</td>
<td>33.3</td>
<td>627.4</td>
<td>MO</td>
<td>MO</td>
</tr>
<tr>
<td>testcase20</td>
<td>s</td>
<td>-</td>
<td>TO</td>
<td>TO</td>
<td>MO</td>
<td>MO</td>
<td>TO</td>
<td>TO</td>
<td>MO</td>
<td>MO</td>
</tr>
<tr>
<td>tbnail-sp1</td>
<td>s</td>
<td>0</td>
<td>58.9</td>
<td>1173.7</td>
<td>62.3</td>
<td>1178.2</td>
<td>MO</td>
<td>MO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>610dd9dc</td>
<td>s</td>
<td>319</td>
<td>72.1</td>
<td>12.5</td>
<td>33.6</td>
<td>148.2</td>
<td>TO</td>
<td>TO</td>
<td>MO</td>
<td>MO</td>
</tr>
</tbody>
</table>

Columns:
- **lemmas**: Eager solver
- **eager**: z3
- **z3**: cvc3
- **cvc3**: stp
• Lemmas on demand for **Extensional Theory of Arrays**
  - In our examples with Bit-vectors, but approach is more general
  - SAT solver is used **offline** as black box

• Algorithm based on **propagation** and direct application of array axioms
  - Non-extensional algorithm with DFS or BFS
  - Introducing equality on arrays requires **fix-point algorithm**
  - Virtual reads as witnesses for array **inequalities**

• Algorithm implemented in our SMT solver Boolector