Turbo-Charging Lemmas on Demand with Don’t Care Reasoning

Aina Niemetz, Mathias Preiner and Armin Biere

Institute for Formal Models and Verification (FMV)
Johannes Kepler University, Linz, Austria
http://fmv.jku.at/

FMCAD 2014
October 21 - 24, 2014
Lausanne, Switzerland
Introduction

Lemmas on Demand

- so-called lazy SMT approach
- our SMT solver **Boolector**
  - implements Lemmas on Demand for
  - the quantifier-free theory of
    - fixed-size bit vectors
    - arrays
- recently: Lemmas on Demand for **Lambdas** [DIFTS'13]
  - generalization of Lemmas on Demand for Arrays [JSAT'09]
  - arrays represented as uninterpreted functions
  - array operations represented as lambda-terms
  - reads represented as function applications
Lemmas on Demand
Workflow: Original Procedure LOD

- bit vector formula abstraction (bit vector skeleton)
- enumeration of truth assignments (candidate models)
- iterative refinement with lemmas until convergence
Lemmas on Demand
Workflow: Original Procedure LOD

Each candidate model is a full truth assignment of the formula abstraction.

Full candidate model needs to be checked for consistency w.r.t. theories.
Lemmas on Demand
Workflow: Original Procedure LOD

→ abstraction refinement usually the most **costly** part of LOD
→ cost generally correlates with number of refinements
→ checking the full candidate model often not required
→ small subset responsible for satisfying formula abstraction
Lemmas on Demand
Workflow: Optimized Procedure $\text{LOD}_{\text{opt}}$

- focus LOD on the relevant parts of the input formula
- exploit a posteriori observability don’t cares
- partial model extraction prior to consistency checking
  → subsequently reduces the cost for consistency checking
Example. \[ \psi_1 \equiv i \neq k \land (f(i) = e \lor f(k) = v) \land v = \text{ite}(i = j, e, g(j)) \]
Example. Bit Vector Skeleton

\[ \alpha(apply_1) \quad \text{var} \quad \alpha(apply_2) \quad \text{var} \quad \alpha(apply_3) \quad \text{var} \quad \text{var} \quad \text{var} \quad \text{var} \]
Example.

**Full Candidate Model**

\[
\alpha(\text{apply}_1) \quad \text{var} \quad \alpha(\text{apply}_2) \quad \text{var} \quad \alpha(\text{apply}_3) \quad \text{var} \quad \text{var} \quad \text{var} \quad \text{var}
\]

\[
00 \quad 00 \quad 00 \quad 00 \quad 00 \quad 00 \quad 00 \quad 01
\]
Example. **Full Candidate Model**

Check consistency: \{apply_1, apply_2, apply_3\}
Example. **Partial** Candidate Model

Check consistency: \{apply_1\}
Partial Model Extraction

Most intuitive: use **justification-based** approach

→ Justification-based techniques in the context of

- **SMT**
  - prune the search space of DPLL(T) [ENTCS’05, MSRTR’07]

- **Model checking**
  - prune the search space of BMC [CAV’02]
  - generalize proof obligations in PDR [EënFMCAD’11, ChoFMCAD’11]
  - generalize candidate counter examples (CEGAR) [LPAR’08]
Partial Model Extraction

Our approach: **Dual propagation**-based partial model extraction

- exploiting the duality of a formula abstraction $\psi$
  - assignments satisfying $\psi$ (the **primal** channel)
  - falsify its negation $\neg \psi$ (the **dual** channel)

- motivated by dual propagation techniques in QBF [AAAI'10]
  - one solver with two channels (**online** approach)
  - symmetric propagation between primal and dual channel

- **here**: **offline** dual propagation
  - two solvers, one solver per channel
  - consecutive propagation between primal and dual channel
    - primal generates full assignment before dual enables partial model extraction based on the primal assignment
Example. **Boolean** Level

**Primal** channel: \( \psi_2 \equiv (a \land b) \lor (c \land d) \)

**Dual** channel: \( \neg \psi_2 \equiv (\neg a \lor \neg b) \land (\neg c \lor \neg d) \)
Partial Model Extraction
Dual Propagation-Based Approach

Example. **Boolean Level**

**Primal** channel: \( \psi_2 \equiv (a \land b) \lor (c \land d) \)

**Dual** channel: \( \neg \psi_2 \equiv (\neg a \lor \neg b) \land (\neg c \lor \neg d) \)

**Primal** assignment: \( \sigma(\psi_2) \equiv \{ \sigma(a) = \top, \sigma(b) = \top, \sigma(c) = \top, \sigma(d) = \top \} \)
Example. **Boolean** Level

**Primal** channel: \( \psi_2 \equiv (a \land b) \lor (c \land d) \)

**Dual** channel: \( \neg \psi_2 \equiv (\neg a \lor \neg b) \land (\neg c \lor \neg d) \)

**Primal** assignment: \( \sigma(\psi_2) \equiv \{ \sigma(a) = T, \sigma(b) = T, \sigma(c) = T, \sigma(d) = T \} \)

Fix values of inputs via **assumptions** to the dual solver:
**Dual** assumptions: \( \{ a = T, b = T, c = T, d = T \} \)
Example. **Boolean** Level

**Primal** channel: \( \psi_2 \equiv (a \land b) \lor (c \land d) \)

**Dual** channel: \( \neg \psi_2 \equiv (\neg a \lor \neg b) \land (\neg c \lor \neg d) \)

**Primal** assignment: \( \sigma(\psi_2) \equiv \{ \sigma(a) = \top, \sigma(b) = \top, \sigma(c) = \top, \sigma(d) = \top \} \)

Fix values of inputs via **assumptions** to the dual solver:

**Dual** assumptions: \( \{ a = \top, b = \top, c = \top, d = \top \} \)

**Failed** assumptions: \( \{ a = \top, b = \top \} \)

\( \rightarrow \) sufficient to falsify \( \neg \psi_2 \)

\( \rightarrow \) sufficient to satisfy \( \psi_2 \)
Partial Model Extraction
Dual Propagation-Based Approach

Example. **Boolean Level**

**Primal** channel: \( \psi_2 \equiv (a \land b) \lor (c \land d) \)

**Dual** channel: \( \neg\psi_2 \equiv (\neg a \lor \neg b) \land (\neg c \lor \neg d) \)

**Primal** assignment: \( \sigma(\psi_2) \equiv \{ \sigma(a) = \top, \sigma(b) = \top, \sigma(c) = \top, \sigma(d) = \top \} \)

Fix values of inputs via assumptions to the dual solver:

**Dual** assumptions: \( \{ a = \top, b = \top, c = \top, d = \top \} \)

**Failed** assumptions: \( \{ a = \top, b = \top \} \) → sufficient to falsify \( \neg\psi_2 \)

\( \neg\psi_2 \) → sufficient to satisfy \( \psi_2 \)
Partial Model Extraction
Dual Propagation-Based Approach

→ structural don’t care reasoning simulated via the dual solver
→ no structural SAT solver necessary

Example. (ctd)

Input formula: \( \psi_2 \equiv (a \land b) \lor (c \land d) \equiv \top \)

Primal SAT solver: \( \text{CNF}(\psi_2) \equiv (\neg o \lor x \lor y) \land (\neg x \lor o) \land (\neg y \lor o) \land (\neg x \lor a) \land (\neg x \lor b) \land (\neg a \lor \neg b \lor x) \land (\neg y \lor c) \land (\neg y \lor d) \land (\neg c \lor \neg d \lor y) \equiv ? \)

Dual SAT solver: \( \text{CNF}(\neg \psi_2) \equiv (\neg a \lor \neg b) \land (\neg c \lor \neg d) \equiv \bot \)

Dual assumptions: \( \{ a = \top, b = \top, c = \top, d = \top \} \)

Partial Model: \( \{ a = \top, b = \top \} \)

→ in contrast to partial model extraction techniques based on iterative removal of unnecessary assignments on the CNF level [FMCAD’13]
we lift this approach to the \textbf{word} level

\textbf{Primal} channel: \[ \Gamma \equiv \alpha(\pi) \land \xi \equiv \alpha(\pi) \land l_1 \land \ldots \land l_{i-1} \]

\textbf{Dual} channel: \[ \neg \Gamma \]

\(\longrightarrow\) one SMT solver per channel

\(\longrightarrow\) one \textbf{single} dual solver instance to maintain \(\neg \Gamma\) over all iterations
Example.  **Word Level**

\[\psi_1 \equiv i \neq k \land (f(i) = e \lor f(k) = v) \land v = \text{ite}(i = j, e, g(j))\]

\[\alpha(\psi_1) \equiv i \neq k \land (\alpha(\text{apply}_1) = e \lor \alpha(\text{apply}_2) = v) \land v = \text{ite}(i = j, e, \alpha(\text{apply}_3))\]

**Primal** solver:  \[\alpha(\psi_1)\]

**Dual** solver:  \[\neg\alpha(\psi_1)\]  \}  \text{Formula abstraction and its negation}\n
**Primal** assignment:

\[\sigma(\psi_2) \equiv \{\sigma(i) = 00, \sigma(j) = 00, \sigma(e) = 00, \sigma(v) = 00, \sigma(k) = 01, \alpha(\text{apply}_1) = 00, \alpha(\text{apply}_2) = 00, \alpha(\text{apply}_3) = 00\}\]

Fix values of inputs via assumptions to the dual solver:

**Dual** assumptions:

\[\sigma(\psi_2) \equiv \{i = 00, j = 00, e = 00, v = 00, k = 01, \alpha(\text{apply}_1) = 00, \alpha(\text{apply}_2) = 00, \alpha(\text{apply}_3) = 00\}\]

**Failed** assumptions:

\[\{i = 00, j = 00, e = 00, v = 00, k = 01, \alpha(\text{apply}_1) = 00\}\]
Partial Model Extraction
Dual Propagation-Based Approach

Example. Word Level

\[
\psi_1 \equiv i \neq k \land (f(i) = e \lor f(k) = v) \land v = \text{ite}(i = j, e, g(j))
\]

\[
\alpha(\psi_1) \equiv i \neq k \land (\alpha(\text{apply}_1) = e \lor \alpha(\text{apply}_2) = v) \land v = \text{ite}(i = j, e, \alpha(\text{apply}_3))
\]

Primal solver: \( \alpha(\psi_1) \)
Dual solver: \( \neg \alpha(\psi_1) \)

Formula abstraction and its negation

Primal assignment:
\[
\sigma(\psi_2) \equiv \{\sigma(i) = 00, \sigma(j) = 00, \sigma(e) = 00, \sigma(v) = 00, \sigma(k) = 01, \\
\alpha(\text{apply}_1) = 00, \alpha(\text{apply}_2) = 00, \alpha(\text{apply}_3) = 00\}
\]

Fix values of inputs via assumptions to the dual solver:
Dual assumptions:
\[
\sigma(\psi_2) \equiv \{i = 00, j = 00, e = 00, v = 00, k = 01, \\
\alpha(\text{apply}_1) = 00, \alpha(\text{apply}_2) = 00, \alpha(\text{apply}_3) = 00\}
\]

Failed assumptions:
\[
\{i = 00, j = 00, e = 00, v = 00, k = 01, \alpha(\text{apply}_1) = 00\}
\]
Partial Model Extraction
Dual Propagation-Based Approach

Example. **Word** Level

$$\psi_1 \equiv i \neq k \land (f(i) = e \lor f(k) = v) \land v = \text{ite}(i = j, e, g(j))$$

$$\alpha(\psi_1) \equiv i \neq k \land (\alpha(\text{apply}_1) = e \lor \alpha(\text{apply}_2) = v) \land v = \text{ite}(i = j, e, \alpha(\text{apply}_3))$$

**Primal** solver:  \(\alpha(\psi_1)\)

**Dual** solver:  \(\neg \alpha(\psi_1)\)

**Primal** assignment:

$$\sigma(\psi_2) \equiv \{\sigma(i) = 00, \sigma(j) = 00, \sigma(e) = 00, \sigma(v) = 00, \sigma(k) = 01, \alpha(\text{apply}_1) = 00, \alpha(\text{apply}_2) = 00, \alpha(\text{apply}_3) = 00\}$$

Fix values of inputs via assumptions to the dual solver:

**Dual** assumptions:

$$\sigma(\psi_2) \equiv \{i = 00, j = 00, e = 00, v = 00, k = 01, \alpha(\text{apply}_1) = 00, \alpha(\text{apply}_2) = 00, \alpha(\text{apply}_3) = 00\}$$

**Failed** assumptions:

\(\{i = 00, j = 00, e = 00, v = 00, k = 01, \alpha(\text{apply}_1) = 00\}\)

Consistency Check
Experimental Evaluation

Configuration

Four Configurations:

- **Boolector\textsubscript{sc}**
  - \(\rightarrow\) version entering SMTCOMP’12, winner of the QF-AUFBV track

- **Boolector\textsubscript{ba}**
  - \(\rightarrow\) current Boolector base version (new LOD for Lambdas engine)

- **Boolector\textsubscript{dp}**
  - \(\rightarrow\) with dual propagation-based partial model extraction enabled

- **Boolector\textsubscript{ju}**
  - \(\rightarrow\) justification-based partial model extraction approach for comparison
    - determine \textit{a posteriori} observability don’t cares
      - \(\rightarrow\) skip lines that do not influence the output of an \textit{and}-gate under its current assignment
    - if both inputs of an \textit{and}-gate are \textit{controlling} (\(\bot\))
      - \(\rightarrow\) skip \textit{either} one based on a minimum cost heuristic
Experimental Evaluation
Configuration

Two Benchmark Sets:

- **SMT’12**: 149 benchmarks
  all non-extensional QF_AUFBV benchmarks in SMTCOMP’12
- **Selected**: 173 benchmarks
  all non-extensional QF_AUFBV benchmarks (13696) in the SMT-LIB (pre-SMTCOMP’14) for which Boolector_sc required at least 10 seconds

→ 58 benchmarks shared between both sets
→ all experiments on 2.83 GHz Intel Core 2 Quad machines with 8GB RAM running Ubuntu 12.04
→ **time limit**: 1200 seconds, **memory limit**: 7GB
Overall results on sets SMT’12 and Selected.

<table>
<thead>
<tr>
<th></th>
<th>Solver</th>
<th>Solved (sat/unsat)</th>
<th>TO</th>
<th>MO</th>
<th>Time [s]</th>
<th>DS [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMT’12</td>
<td>Boolector_{sc}</td>
<td>140 (83/57)</td>
<td>9</td>
<td>0</td>
<td>15882</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Boolector_{ba}</td>
<td>141 (83/58)</td>
<td>8</td>
<td>0</td>
<td>19312</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Boolector_{ju}</td>
<td>142 (84/58)</td>
<td>7</td>
<td>0</td>
<td>15709</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Boolector_{dp}</td>
<td>142 (84/58)</td>
<td>7</td>
<td>0</td>
<td>20992</td>
<td>5045</td>
</tr>
<tr>
<td>Selected</td>
<td>Boolector_{sc}</td>
<td>116 (72/44)</td>
<td>50</td>
<td>7</td>
<td>85863</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Boolector_{ba}</td>
<td>121 (76/45)</td>
<td>45</td>
<td>7</td>
<td>76104</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Boolector_{ju}</td>
<td>130 (85/45)</td>
<td>36</td>
<td>7</td>
<td>63202</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Boolector_{dp}</td>
<td>130 (85/45)</td>
<td>36</td>
<td>7</td>
<td>66991</td>
<td>4705</td>
</tr>
</tbody>
</table>

TO ... time out   MO ... memory out
Time ... total CPU time   DS ... dual solver overhead
# Experimental Evaluation

## Overview

Overall results on sets **SMT’12** and **Selected**.

<table>
<thead>
<tr>
<th>Solver</th>
<th>Solved (sat/unsat)</th>
<th>TO</th>
<th>MO</th>
<th>Time [s]</th>
<th>DS [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SMT’12</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boolector&lt;sub&gt;sc&lt;/sub&gt;</td>
<td>140 (83/57)</td>
<td>9</td>
<td>0</td>
<td>15882</td>
<td>-</td>
</tr>
<tr>
<td>Boolector&lt;sub&gt;ba&lt;/sub&gt;</td>
<td>141 (83/58)</td>
<td>8</td>
<td>0</td>
<td>19312</td>
<td>-</td>
</tr>
<tr>
<td>Boolector&lt;sub&gt;ju&lt;/sub&gt;</td>
<td>142 (84/58)</td>
<td>7</td>
<td>0</td>
<td>15709</td>
<td>-</td>
</tr>
<tr>
<td>Boolector&lt;sub&gt;dp&lt;/sub&gt;</td>
<td>142 (84/58)</td>
<td>7</td>
<td>0</td>
<td>20992</td>
<td>5045</td>
</tr>
<tr>
<td><strong>Selected</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boolector&lt;sub&gt;sc&lt;/sub&gt;</td>
<td>116 (72/44)</td>
<td>50</td>
<td>7</td>
<td>85863</td>
<td>-</td>
</tr>
<tr>
<td>Boolector&lt;sub&gt;ba&lt;/sub&gt;</td>
<td>121 (76/45)</td>
<td>45</td>
<td>7</td>
<td>76104</td>
<td>-</td>
</tr>
<tr>
<td>Boolector&lt;sub&gt;ju&lt;/sub&gt;</td>
<td>130 (85/45)</td>
<td>36</td>
<td>7</td>
<td>63202</td>
<td>-</td>
</tr>
<tr>
<td>Boolector&lt;sub&gt;dp&lt;/sub&gt;</td>
<td>130 (85/45)</td>
<td>36</td>
<td>7</td>
<td>66991</td>
<td>4705</td>
</tr>
</tbody>
</table>

TO ... time out  
MO ... memory out  
Time ... total CPU time  
DS ... dual solver overhead

- **SMT’12**: 1 additional instance (sat)  
- **Selected**: 9 additional instances (all sat)
Experimental Evaluation
Commonly Solved Instances

Results for **commonly solved** instances on sets SMT’12 and Selected.

<table>
<thead>
<tr>
<th>Solver</th>
<th>Time [s]</th>
<th>SAT [s]</th>
<th>DS overhead [s]</th>
<th>LOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMT’12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boolector&lt;sub&gt;sc&lt;/sub&gt;</td>
<td>4129</td>
<td>29</td>
<td>2</td>
<td>3662</td>
</tr>
<tr>
<td>Boolector&lt;sub&gt;ba&lt;/sub&gt;</td>
<td>8564</td>
<td>61</td>
<td>6</td>
<td>7262</td>
</tr>
<tr>
<td>Boolector&lt;sub&gt;ju&lt;/sub&gt;</td>
<td>6362</td>
<td>45</td>
<td>4</td>
<td>5226</td>
</tr>
<tr>
<td>Boolector&lt;sub&gt;dp&lt;/sub&gt;</td>
<td>10145</td>
<td>72</td>
<td>5</td>
<td>4700</td>
</tr>
<tr>
<td>Selected</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boolector&lt;sub&gt;sc&lt;/sub&gt;</td>
<td>15037</td>
<td>133</td>
<td>35</td>
<td>12836</td>
</tr>
<tr>
<td>Boolector&lt;sub&gt;ba&lt;/sub&gt;</td>
<td>10001</td>
<td>88</td>
<td>35</td>
<td>8330</td>
</tr>
<tr>
<td>Boolector&lt;sub&gt;ju&lt;/sub&gt;</td>
<td>8182</td>
<td>72</td>
<td>29</td>
<td>6639</td>
</tr>
<tr>
<td>Boolector&lt;sub&gt;dp&lt;/sub&gt;</td>
<td>10838</td>
<td>95</td>
<td>30</td>
<td>6164</td>
</tr>
</tbody>
</table>

- **SMT’12**: 139 (out of 149) benchmarks, 82 sat, 57 unsat
  → not representative:
  ~50% solved without a single refinement iteration
- **Selected**: 113 (out of 173) benchmarks, 70 sat, 43 unsat

<table>
<thead>
<tr>
<th>Time</th>
<th>total CPU time</th>
<th>SAT</th>
<th>SAT solver runtime (primal solver)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS overhead</td>
<td>dual solver overhead</td>
<td>LOD</td>
<td>number of lemmas generated</td>
</tr>
</tbody>
</table>

- Time: total CPU time
- SAT: SAT solver runtime (primal solver)
- LOD: number of lemmas generated
Experimental Evaluation
Commonly Solved Instances

Results for *commonly solved* instances on sets SMT’12 and Selected.

<table>
<thead>
<tr>
<th>Solver</th>
<th>Time [s]</th>
<th>SAT [s]</th>
<th>DS overhead</th>
<th>LOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMT’12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boolector_{sc}</td>
<td>4129</td>
<td>29</td>
<td>2</td>
<td>3662</td>
</tr>
<tr>
<td>Boolector_{ba}</td>
<td>8564</td>
<td>61</td>
<td>6</td>
<td>7262</td>
</tr>
<tr>
<td>Boolector_{ju}</td>
<td>6362</td>
<td>45</td>
<td>4</td>
<td>5226</td>
</tr>
<tr>
<td>Boolector_{dp}</td>
<td>10145</td>
<td>72</td>
<td>5</td>
<td>4700</td>
</tr>
<tr>
<td>Selected</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boolector_{sc}</td>
<td>15037</td>
<td>133</td>
<td>35</td>
<td>12836</td>
</tr>
<tr>
<td>Boolector_{ba}</td>
<td>10001</td>
<td>88</td>
<td>35</td>
<td>8330</td>
</tr>
<tr>
<td>Boolector_{ju}</td>
<td>8182</td>
<td>72</td>
<td>29</td>
<td>6639</td>
</tr>
<tr>
<td>Boolector_{dp}</td>
<td>10838</td>
<td>95</td>
<td>30</td>
<td>6164</td>
</tr>
</tbody>
</table>

- Time ... total CPU time
- SAT ... SAT solver runtime (primal solver)
- DS overhead ... dual solver overhead
- LOD ... number of lemmas generated

- *Boolector_{sc}* implements old LOD engine
  - new engine (*Boolector_{ba}* ) struggles on a small set of benchmarks
  - needs further investigation
## Experimental Evaluation

### Commonly Solved Instances

Results for **commonly solved** instances on sets SMT’12 and Selected.

<table>
<thead>
<tr>
<th>Solver</th>
<th>Time [s]</th>
<th>SAT [s]</th>
<th>DS overhead [s]</th>
<th>LOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMT’12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boolector</td>
<td>4129</td>
<td>29</td>
<td>2</td>
<td>3662</td>
</tr>
<tr>
<td>sc</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boolector</td>
<td>8564</td>
<td>61</td>
<td>6</td>
<td>7262</td>
</tr>
<tr>
<td>ba</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boolector</td>
<td>6362</td>
<td>45</td>
<td>4</td>
<td>5226</td>
</tr>
<tr>
<td>juju</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boolector</td>
<td>10145</td>
<td>72</td>
<td>5</td>
<td>4700</td>
</tr>
<tr>
<td>dp</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Selected</th>
<th>Time [s]</th>
<th>SAT [s]</th>
<th>DS overhead [s]</th>
<th>LOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolector</td>
<td>15037</td>
<td>133</td>
<td>35</td>
<td>12836</td>
</tr>
<tr>
<td>sc</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boolector</td>
<td>10001</td>
<td>88</td>
<td>35</td>
<td>8330</td>
</tr>
<tr>
<td>ba</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boolector</td>
<td>8182</td>
<td>72</td>
<td>29</td>
<td>6639</td>
</tr>
<tr>
<td>ju</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boolector</td>
<td>10838</td>
<td>95</td>
<td>30</td>
<td>6164</td>
</tr>
<tr>
<td>dp</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- sat solver runtime (**SAT**)  
  \[\Rightarrow \text{Boolector}_{dp}\] most notable improvement on both sets

**Note:** Time ... total CPU time  
**DS overhead** ... dual solver overhead  
LOD ... number of lemmas generated
Experimental Evaluation
Commonly Solved Instances

Results for commonly solved instances on sets SMT’12 and Selected.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
<td>Avg.</td>
<td>Med.</td>
<td>Total</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Avg.</td>
</tr>
<tr>
<td>SMT’12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boolector_{sc}</td>
<td>4129</td>
<td>29</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>Boolector_{ba}</td>
<td>8564</td>
<td>61</td>
<td>6</td>
<td>-</td>
</tr>
<tr>
<td>Boolector_{ju}</td>
<td>6362</td>
<td>45</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>Boolector_{dp}</td>
<td>10145</td>
<td>72</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>Selected</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boolector_{sc}</td>
<td>15037</td>
<td>133</td>
<td>35</td>
<td>-</td>
</tr>
<tr>
<td>Boolector_{ba}</td>
<td>10001</td>
<td>88</td>
<td>35</td>
<td>-</td>
</tr>
<tr>
<td>Boolector_{ju}</td>
<td>8182</td>
<td>72</td>
<td>29</td>
<td>-</td>
</tr>
<tr>
<td>Boolector_{dp}</td>
<td>10838</td>
<td>95</td>
<td>30</td>
<td>-</td>
</tr>
</tbody>
</table>

Time ... total CPU time
SAT ... SAT solver runtime (primal solver)
DS overhead ... dual solver overhead
LOD ... number of lemmas generated

- number of lemmas generated (LOD)
  - SMT’12:
    - Boolector_{ju} least number of lemmas
    - Boolector_{dp} and Boolector_{ba} approx. the same
      → on 14 instances 1.5-2.6 × more lemmas than Boolector_{ba}
  - Selected: Boolector_{dp} most notable improvement
## Experimental Evaluation

### Commonly Solved Instances

Results for **commonly solved** instances on sets SMT’12 and Selected.

<table>
<thead>
<tr>
<th>Solver</th>
<th>Time [s]</th>
<th>SAT [s]</th>
<th>DS overhead [s]</th>
<th>LOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMT’12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boolector\textsubscript{sc}</td>
<td>4129 29 2</td>
<td>3662 26 0</td>
<td>- - -</td>
<td>30741 221 0</td>
</tr>
<tr>
<td>Boolector\textsubscript{ba}</td>
<td>8564 61 6</td>
<td>7262 52 1</td>
<td>- - -</td>
<td>33013 237 0</td>
</tr>
<tr>
<td>Boolector\textsubscript{ju}</td>
<td>6362 45 4</td>
<td>5226 37 0</td>
<td>- - -</td>
<td>23660 170 0</td>
</tr>
<tr>
<td>Boolector\textsubscript{dp}</td>
<td>10145 72 5</td>
<td>4700 33 0</td>
<td>4109 29 0</td>
<td>33492 240 0</td>
</tr>
<tr>
<td>Selected</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boolector\textsubscript{sc}</td>
<td>15037 133 35</td>
<td>12836 113 34</td>
<td>- - -</td>
<td>104646 926 175</td>
</tr>
<tr>
<td>Boolector\textsubscript{ba}</td>
<td>10001 88 35</td>
<td>8330 73 22</td>
<td>- - -</td>
<td>31752 280 88</td>
</tr>
<tr>
<td>Boolector\textsubscript{ju}</td>
<td>8182 72 29</td>
<td>6639 58 19</td>
<td>- - -</td>
<td>28215 249 28</td>
</tr>
<tr>
<td>Boolector\textsubscript{dp}</td>
<td>10838 95 30</td>
<td>6164 54 15</td>
<td>3036 26 0</td>
<td>24866 220 29</td>
</tr>
</tbody>
</table>

**Time** ... total CPU time
**DS overhead** ... dual solver overhead
**LOD** ... number of lemmas generated

- **dual solver overhead** $\sim$30-40\% in total
  - on $\leq$10\% of the benchmarks 50-70\% of the total runtime
  - on $>$50\% of the benchmarks $<$10\% of the total runtime

$\rightarrow$ **Boolector\textsubscript{dp}** outperforms others disregarding DS overhead

$\rightarrow$ **online** dual propagation approach: DS overhead negligible
Experimental Evaluation

Booleterm dp vs Booleterm ba

DS overhead included

DS overhead not included
Conclusion

→ **dual propagation-based** optimization for Lemmas on Demand

- don’t care reasoning on full candidate models improves performance
- our offline dual propagation-based approach competitive (in spite of introducing considerable overhead)
  → Boolector$_{ju}$ won QF-ABV track of SMTCOMP'14
  → Boolector$_{dp}$ came in close second

**Future work:** **online** dual propagation approach, promises

- negligible or no dual solver overhead
- further improvement of overall performance by enabling partial model extraction even before a full candidate model has been generated
- requires interleaved execution between primal and dual solver
Appendix

Boolector$_{dp}$ vs Boolector$_{ju}$

DS overhead included

DS overhead not included
Appendix

Boolector<sub>dp</sub> vs Boolector<sub>sc</sub>

DS overhead included

DS overhead <strong>not</strong> included


R. Brummayer and A. Biere. Lemmas on demand for the extensional theory of arrays. JSAT, 6(1-3), 2009.


