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/

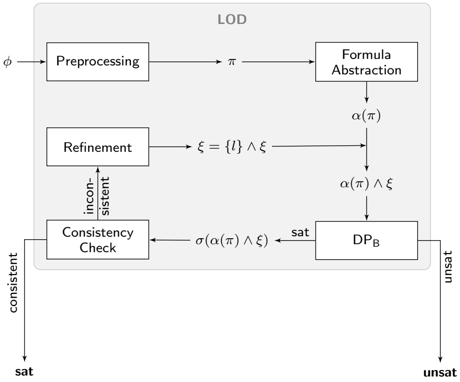
FMCAD 2014
October 21 - 24, 2014
Lausanne, Switzerland

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

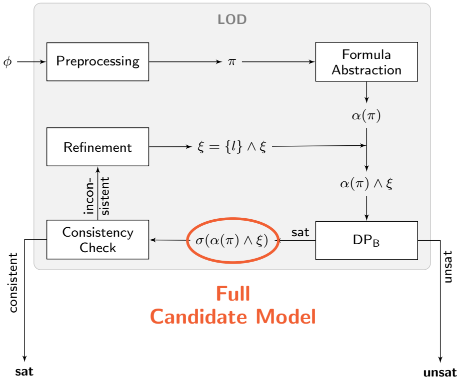
Workflow: Original Procedure LOD



- 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

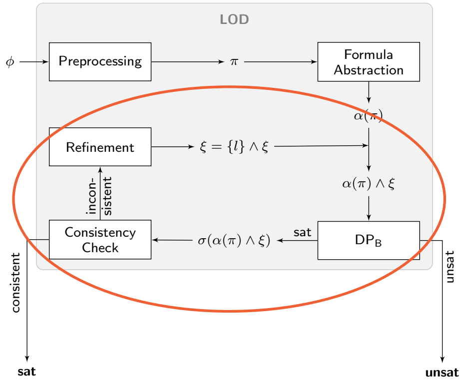


⟶ **each** candidate model is a **full** truth assignment of the formula abstraction

⟶ **full candidate model** needs to be checked for consistency w.r.t. theories

Lemmas on Demand

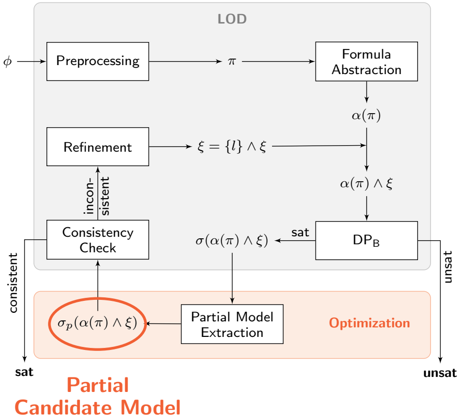
Workflow: Original Procedure LOD



- ⟶ 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

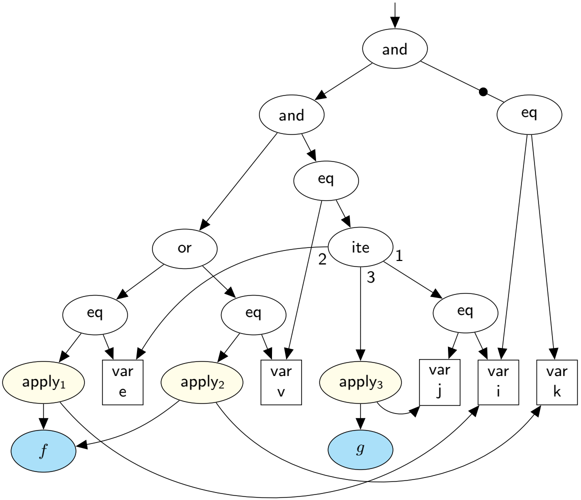
Workflow: Optimized Procedure LOD_{opt}



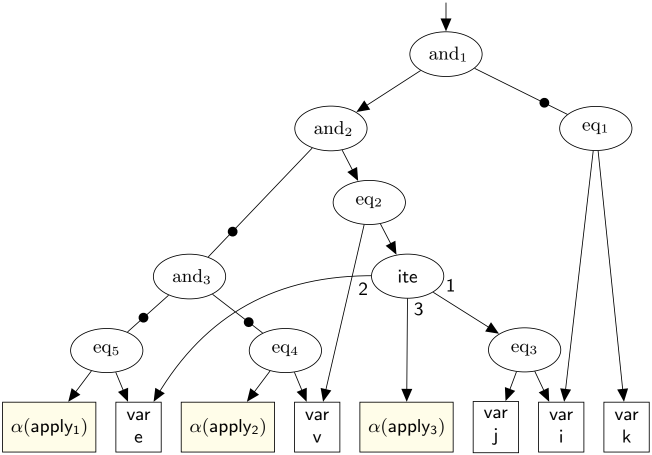
- focus LOD on the **relevant** parts of the input formula
- exploit **a posteriori observability don't cares**
- **partial model extraction** prior to consistency checking

$\longrightarrow$ subsequently **reduces the cost** for consistency checking

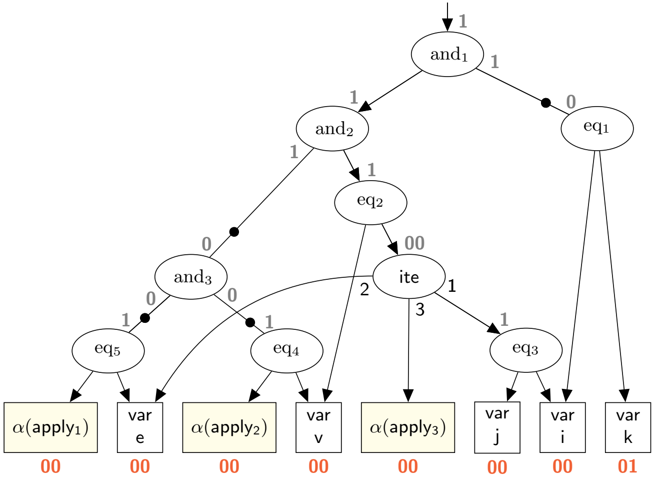
Example. $\psi_1 \equiv i \neq k \wedge (f(i) = e \vee f(k) = v) \wedge v = ite(i = j, e, g(j))$



Example. Bit Vector Skeleton

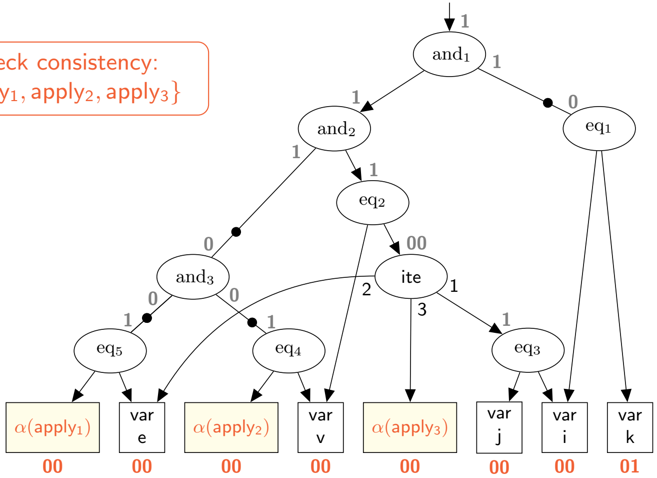


Example. **Full** Candidate Model



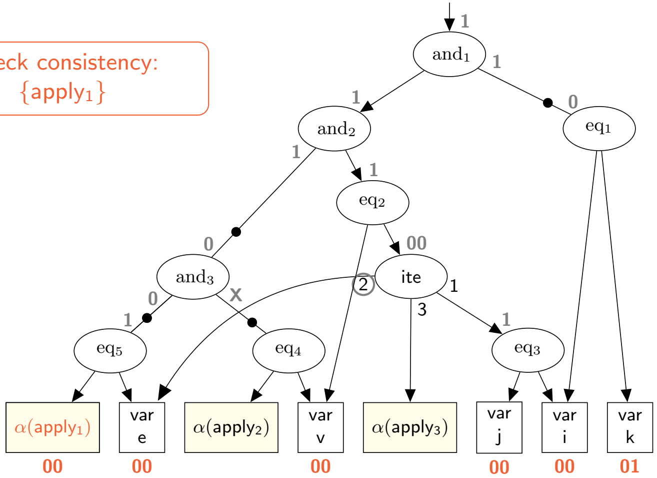
Example. **Full** Candidate Model

Check consistency:
$\{\text{apply}_1, \text{apply}_2, \text{apply}_3\}$



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 [EénFMCAD'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 ψ

  $\longrightarrow$ assignments satisfying ψ (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

    $\longrightarrow$ primal generates full assignment before dual enables partial model extraction based on the primal assignment

Example. **Boolean** Level

Primal channel: $\psi_2 \equiv (a \wedge b) \vee (c \wedge d)$
Dual channel: $\neg\psi_2 \equiv (\neg a \vee \neg b) \wedge (\neg c \vee \neg d)$

Example. **Boolean** Level

Primal channel: $\psi_2 \equiv (a \wedge b) \vee (c \wedge d)$
Dual channel: $\neg\psi_2 \equiv (\neg a \vee \neg b) \wedge (\neg c \vee \neg d)$

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

Example. **Boolean** Level

Primal channel: $\psi_2 \equiv (a \wedge b) \vee (c \wedge d)$
Dual channel: $\neg\psi_2 \equiv (\neg a \vee \neg b) \wedge (\neg c \vee \neg d)$

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: $\{a{=}\top, b{=}\top, c{=}\top, d{=}\top\}$

Example. **Boolean** Level

Primal channel: $\psi_2 \equiv (a \wedge b) \vee (c \wedge d)$
Dual channel: $\neg\psi_2 \equiv (\neg a \vee \neg b) \wedge (\neg c \vee \neg d)$

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: $\{a{=}\top,\ b{=}\top\}$

$\longrightarrow$ sufficient to falsify $\neg\psi_2$
$\longrightarrow$ sufficient to satisfy ψ_2

Example. **Boolean** Level

Primal channel: $\psi_2 \equiv (a \wedge b) \vee (c \wedge d)$
Dual channel: $\neg\psi_2 \equiv (\neg a \vee \neg b) \wedge (\neg c \vee \neg d)$

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: $\{a{=}\top,\ b{=}\top\}$ **Partial Model**

$\longrightarrow$ sufficient to falsify $\neg\psi_2$
$\longrightarrow$ sufficient to satisfy ψ_2

$\longrightarrow$ **structural don't care reasoning** simulated via the dual solver

$\longrightarrow$ no structural SAT solver necessary

Example. (ctd)

Input formula:	ψ_2	$\equiv$	$(a \wedge b) \vee (c \wedge d)$	$\equiv \top$
Primal SAT solver:	CNF(ψ_2)	$\equiv$	$(\neg o \vee x \vee y) \wedge (\neg x \vee o) \wedge$ $(\neg y \vee o) \wedge (\neg x \vee a) \wedge$ $(\neg x \vee b) \wedge (\neg a \vee \neg b \vee x) \wedge$ $(\neg y \vee c) \wedge (\neg y \vee d) \wedge$ $(\neg c \vee \neg d \vee y)$	$\equiv$?
Dual SAT solver:	CNF($\neg\psi_2$)	$\equiv$	$(\neg a \vee \neg b) \wedge (\neg c \vee \neg d)$	$\equiv \bot$

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

Partial Model: $\{a = \top, b = \top\}$

$\longrightarrow$ in contrast to partial model extraction techniques based on iterative removal of unnecessary assignments on the CNF level [FMCAD'13]

$\longrightarrow$ we lift this approach to the **word** level

Primal channel: $\Gamma \equiv \alpha(\pi) \wedge \xi \equiv \alpha(\pi) \wedge l_1 \wedge ... \wedge l_{i-1}$

Dual channel: $\neg\Gamma$

$\longrightarrow$ one SMT solver per channel

$\longrightarrow$ one single dual solver instance to maintain $\neg\Gamma$ over all iterations

Example. **Word** Level

$$\psi_1 \equiv i \neq k \wedge (f(i) = e \vee f(k) = v) \wedge v = ite(i = j,\, e,\, g(j))$$
$$\alpha(\psi_1) \equiv i \neq k \wedge (\alpha(\mathsf{apply}_1) = e \vee \alpha(\mathsf{apply}_2) = v) \wedge v = ite(i = j,\, e,\, \alpha(\mathsf{apply}_3))$$

Primal solver: $\alpha(\psi_1)$
Dual solver: $\neg\alpha(\psi_1)$
} Formula abstraction and its negation

Primal assignment:

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

Fix values of inputs via assumptions to the dual solver:

Dual assumptions:

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

Failed assumptions:

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

Example. **Word** Level

$$\psi_1 \equiv i \neq k \wedge (f(i) = e \vee f(k) = v) \wedge v = ite(i = j,\, e,\, g(j))$$
$$\alpha(\psi_1) \equiv i \neq k \wedge (\alpha(\mathsf{apply}_1) = e \vee \alpha(\mathsf{apply}_2) = v) \wedge v = ite(i = j,\, e,\, \alpha(\mathsf{apply}_3))$$

Primal solver: $\alpha(\psi_1)$
Dual solver: $\neg\alpha(\psi_1)$
} Formula abstraction and its negation

Primal assignment:

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

Fix values of inputs via assumptions to the dual solver:

Dual assumptions:

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

Failed assumptions:

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

Partial Model

Example. **Word** Level

$$\psi_1 \equiv i \neq k \wedge (f(i) = e \vee f(k) = v) \wedge v = ite(i = j,\, e,\, g(j))$$
$$\alpha(\psi_1) \equiv i \neq k \wedge (\alpha(\mathsf{apply}_1) = e \vee \alpha(\mathsf{apply}_2) = v) \wedge v = ite(i = j,\, e,\, \alpha(\mathsf{apply}_3))$$

Primal solver: $\alpha(\psi_1)$
Dual solver: $\neg\alpha(\psi_1)$
} Formula abstraction and its negation

Primal assignment:

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

Fix values of inputs via assumptions to the dual solver:

Dual assumptions:

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

Failed assumptions:

Consistency Check

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

Four Configurations:

- **Boolector$_{\mathbf{sc}}$**
  ⟶ version entering SMTCOMP'12, winner of the QF_AUFBV track
- **Boolector$_{\mathbf{ba}}$**
  ⟶ current Boolector base version (new LOD for Lambdas engine)
- **Boolector$_{\mathbf{dp}}$**
  ⟶ with dual propagation-based partial model extraction enabled
- **Boolector$_{\mathbf{ju}}$**
  ⟶ **justification**-based partial model extraction approach for comparison
  - determine **a posteriori** observability don't cares
    ⟶ skip lines that do not influence the output of an **and**-gate under its current assignment
  - if both inputs of an **and**-gate are **controlling** ($\perp$)
    ⟶ skip **either** one based on a minimum cost **heuristic**

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**

$\longrightarrow$ 58 benchmarks shared between both sets

$\longrightarrow$ all experiments on 2.83 GHz Intel Core 2 Quad machines with 8GB RAM running Ubuntu 12.04

$\longrightarrow$ **time limit**: 1200 seconds, **memory limit**: 7GB

Overall results on sets SMT'12 and Selected.

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

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

Overall results on sets SMT'12 and Selected.

	Solver	Solved (sat/unsat)	TO	MO	Time [s]	DS [s]
SMT'12	Boolector$_{sc}$	140 (83/57)	9	0	15882	-
	Boolector$_{ba}$	141 (83/58)	8	0	19312	-
	Boolector$_{ju}$	142 (84/58)	7	0	15709	-
	Boolector$_{dp}$	142 (84/58)	7	0	20992	5045
Selected	Boolector$_{sc}$	116 (72/44)	50	7	85863	-
	Boolector$_{ba}$	121 (76/45)	45	7	76104	-
	Boolector$_{ju}$	130 (85/45)	36	7	63202	-
	Boolector$_{dp}$	130 (85/45)	36	7	66991	4705

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)

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

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

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

- SMT'12: 139 (out of 149) benchmarks, 82 sat, 57 unsat

  $\longrightarrow$ not representative:
  $\sim$50% solved without a single refinement iteration

- Selected: 113 (out of 173) benchmarks, 70 sat, 43 unsat

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

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

Time ... total CPU time
DS overhead ... dual solver overhead
SAT ... SAT solver runtime (primal solver)
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

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

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

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

- sat solver runtime (**SAT**)
  $\longrightarrow$ **Boolector_{dp}** most notable improvement on both sets

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

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

Time ... total CPU time
DS overhead ... dual solver overhead
SAT ... SAT solver runtime (primal solver)
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
      $\longrightarrow$ on 14 instances 1.5-2.6 x more lemmas than Boolector_{ba}
  - Selected: **Boolector_{dp}** most notable improvement

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

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

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

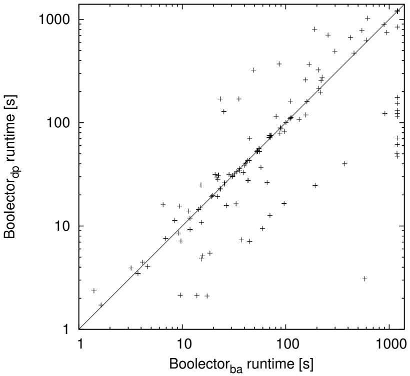
- dual solver overhead ∼30-40% in total
  - on ≤10% of the benchmarks 50-70% of the total runtime
  - on >50% of the benchmarks <10% of the total runtime

⟶ Boolector$_{dp}$ outperforms others disregarding DS overhead

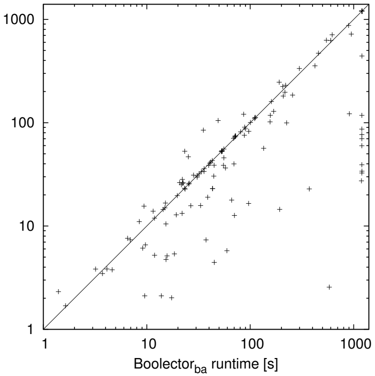
⟶ **online** dual propagation approach: DS overhead negligible

Experimental Evaluation

Boolector$_{dp}$ vs Boolector$_{ba}$



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)

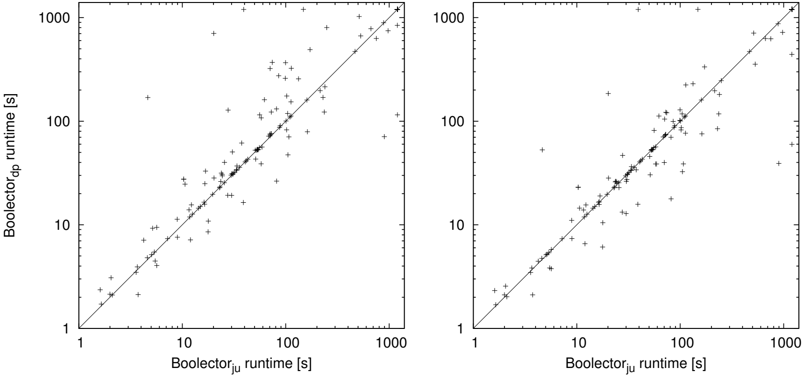
⟶ **Boolector$_{\mathbf{ju}}$** won QF_ABV track of SMTCOMP'14
⟶ **Boolector$_{\mathbf{dp}}$** came in close second

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

- negligible or no dual solver overhead
- further improvment 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$_{ju}$

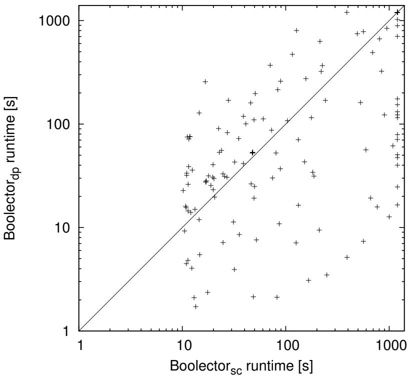


DS overhead included

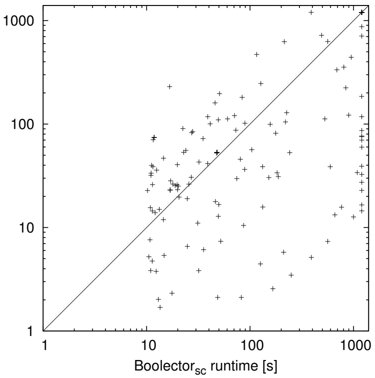
DS overhead **not** included

Appendix

Boolector$_{dp}$ vs Boolector$_{sc}$



DS overhead included



DS overhead **not** included

References I

- J. D. Bingham and A. J. Hu. Semi-formal bounded model checking. In CAV'02, volume 2404 of LNCS. Springer, 2002.
- C. Barrett and J. Donham. Combining sat methods with non-clausal decision heuristics. ENTCS, 125(3), 2005.
- L. de Moura and N. Bjørner. Relevancy propagation. Technical Report MSR-TR-2007-140, Microsoft Research, 2007.
- Z. S. Andraus, M. H. Liffiton, and K. A. Sakallah. Reveal: A formal verification tool for Verilog designs. In LPAR'08, volume 5330 of LNCS. Springer, 2008.
- R. Brummayer and A. Biere. Lemmas on demand for the extensional theory of arrays. JSAT, 6(1-3), 2009.
- H. Chockler, A. Ivrii, A. Matsliah, S. Moran, and Z. Nevo. Incremental formal verification of hardware. In FMCAD'11. FMCAD Inc., 2011.
- N. Eén, A. Mishchenko, and R. K. Brayton. Efficient implementation of property directed reachability. In FMCAD'11. FMCAD Inc., 2011.

References II

- D. Déharbe, P. Fontaine, D. Le Berre and B. Mazure. Computing prime implicants. In FMCAD'13. IEEE, 2013.
- A. Goultiaeva and F. Bacchus. Exploiting QBF duality on a circuit representation. In AAAI'10. AAAI Press, 2010.
- M. Preiner, A. Niemetz and A. Biere. Lemmas on Demand for Lambdas. In DIFTS'13, volume 1130 of CEUR Workshop Proceedings, 2013.