A Concurrency Problem with Exponential DPLL($\mathcal{T}$) Proofs

Liana Hadarean[1] **Alex Horn**[1] Tim King[2]

[1]University of Oxford

[2]Verimag

June 5, 2015

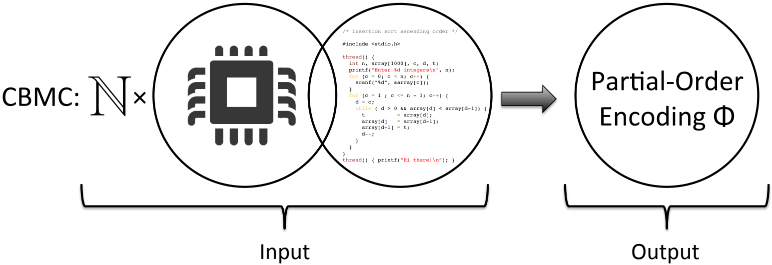
Outline

SAT/SMT-based Verification Techniques

SAT/SMT solvers are highly optimized decision procedures:

$$\phi \rightarrow \square \begin{array}{l} \nearrow \text{SAT} \\ \searrow \text{UNSAT} \end{array}$$

Using these, state-of-the-art symbolic bounded model checker such as CBMC can find concurrency bugs in Apache, PostgreSQL and .

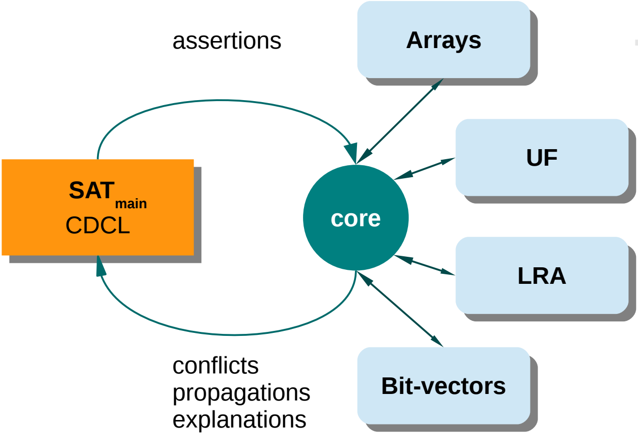


The Problem

Can we pinpoint the challenges (if any) symbolic partial-order encodings of concurrency pose for SAT/SMT solvers?

If yes, what insights can we gain?

SMT Solvers built on DPLL($\mathcal{T}$)



Fixed-Alphabet DPLL($\mathcal{T}$) Proofs

The SMT solvers in our experiments are built on the DPLL($\mathcal{T}$) framework. A simplified form of DPLL($\mathcal{T}$) with only two rules:

- Propositional resolution (Res);
- Learning $\mathcal{T}$-valid clauses over the literals of a fixed alphabet of $\mathcal{T}$-atoms ($\mathcal{T}$-learn).

Fixed-Alphabet DPLL($\mathcal{T}$) Proofs

The SMT solvers in our experiments are built on the DPLL($\mathcal{T}$) framework. A simplified form of DPLL($\mathcal{T}$) with only two rules:

- Propositional resolution (Res);
- Learning $\mathcal{T}$-valid clauses over the literals of a fixed alphabet of $\mathcal{T}$-atoms ($\mathcal{T}$-learn).

Example: $(x < y \vee x = y) \wedge y < x$ where $x, y \in \mathbb{Z}$. This formula is $\mathcal{T}$-unsatisfiable where $\mathcal{T}$ is `QF_LIA`.

Fixed-Alphabet DPLL($\mathcal{T}$) Proofs

Example: $(\overbrace{x < y}^{A} \vee \overbrace{x = y}^{B}) \wedge \overbrace{y < x}^{C}$

Fixed-Alphabet DPLL($\mathcal{T}$) Proofs

Example: $(\overbrace{x < y}^{A} \vee \overbrace{x = y}^{B}) \wedge \overbrace{y < x}^{C}$

$$\rightsquigarrow (\underbrace{\;A\;}_{\top} \vee \underbrace{\;B\;}_{\bot}) \wedge \underbrace{\;C\;}_{\top}$$ (SAT: Yes)

( )

( )

Fixed-Alphabet DPLL($\mathcal{T}$) Proofs

Example: $(\overbrace{x < y}^{A} \vee \overbrace{x = y}^{B}) \wedge \overbrace{y < x}^{C}$

$$\rightsquigarrow (\underbrace{\;A\;}_{\top} \vee \underbrace{\;B\;}_{\bot}) \wedge \underbrace{\;C\;}_{\top} \qquad \text{(SAT: Yes)}$$

$$\rightsquigarrow (A \vee B) \wedge C \wedge (\neg A \vee \neg C) \qquad (\mathcal{T}\text{-LEARN})$$

(   )

Fixed-Alphabet DPLL($\mathcal{T}$) Proofs

Example: $(\overbrace{x < y}^{A} \vee \overbrace{x = y}^{B}) \wedge \overbrace{y < x}^{C}$

$$\rightsquigarrow (\underbrace{A}_{\top} \vee \underbrace{B}_{\bot}) \wedge \underbrace{C}_{\top} \qquad \text{(SAT: Yes)}$$

$$\rightsquigarrow (A \vee B) \wedge C \wedge (\neg A \vee \neg C) \qquad (\mathcal{T}\text{-LEARN})$$

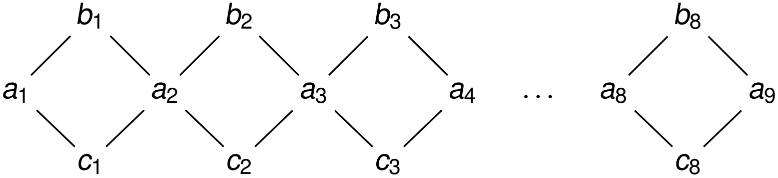
$$\rightsquigarrow (\underbrace{A}_{\bot} \vee \underbrace{B}_{\top}) \wedge \underbrace{C}_{\top} \wedge (\underbrace{\neg A}_{\top} \vee \underbrace{\neg C}_{\bot}) \qquad \text{(SAT: Yes)}$$

$$\rightsquigarrow (A \vee B) \wedge C \wedge (\neg A \vee \neg C) \wedge (\neg B \vee \neg C) \qquad (\mathcal{T}\text{-LEARN})$$

$$\rightsquigarrow \bot \qquad \text{(RES)}$$

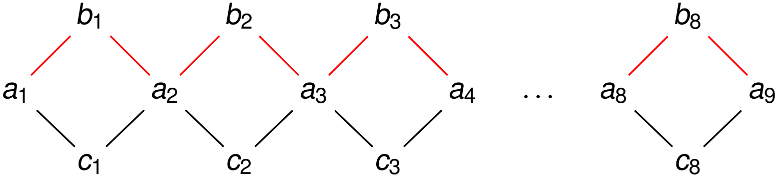
Known Challenges for SMT Solvers

$$\phi_\diamond \triangleq a_1 \neq a_9 \wedge \bigwedge_{i=1}^{8} (a_i = b_i \wedge b_i = a_{i+1}) \vee (a_i = c_i \wedge c_i = a_{i+1})$$



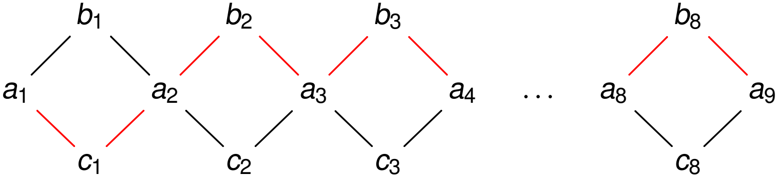
Known Challenges for SMT Solvers

$$\phi_\diamond \triangleq a_1 \neq a_9 \wedge \bigwedge_{i=1}^{8} (a_i = b_i \wedge b_i = a_{i+1}) \vee (a_i = c_i \wedge c_i = a_{i+1})$$



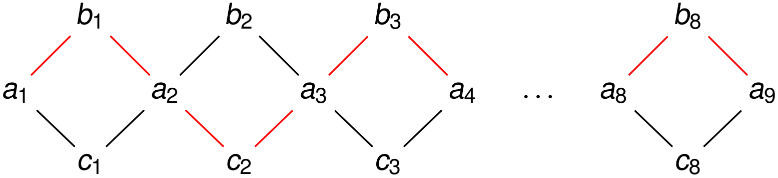
Known Challenges for SMT Solvers

$$\phi_\diamond \triangleq a_1 \neq a_9 \land \bigwedge_{i=1}^{8} (a_i = b_i \land b_i = a_{i+1}) \lor (a_i = c_i \land c_i = a_{i+1})$$



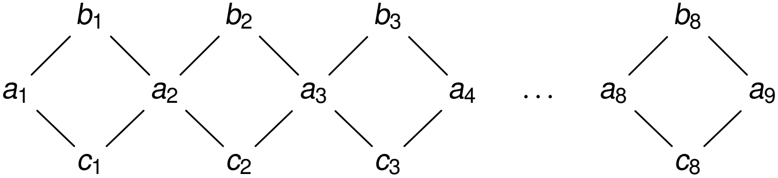
Known Challenges for SMT Solvers

$$\phi_\diamond \triangleq a_1 \neq a_9 \wedge \bigwedge_{i=1}^{8} (a_i = b_i \wedge b_i = a_{i+1}) \vee (a_i = c_i \wedge c_i = a_{i+1})$$



Known Challenges for SMT Solvers

$$\phi_\diamond \triangleq a_1 \neq a_9 \wedge \bigwedge_{i=1}^{8} (a_i = b_i \wedge b_i = a_{i+1}) \vee (a_i = c_i \wedge c_i = a_{i+1})$$

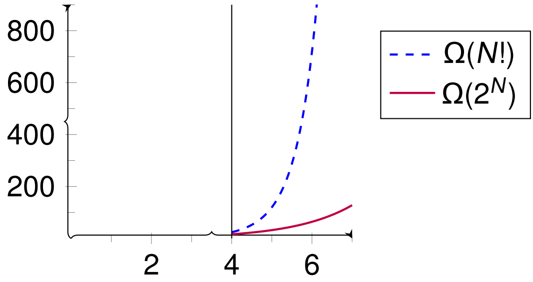


- Note that $\phi_\diamond$ is unsatisfiable because $a_i = a_{i+1}$ for all $1 \leq i \leq 8$.
- But DPLL($\mathcal{T}$) cannot learn these equalities and enumerates 2^8 theory conflicts instead, the infamous *diamonds problem*.[1]

[1] Bjørner, N., Dutertre, B., de Moura, L.: *Accelerating Lemma Learning using Joins - DPLL(Join)*. In: LPAR (2008)

Contributions

- A general framework for establishing lower bounds on the number of $\mathcal{T}$-conflicts in the fixed-alphabet DPLL($\mathcal{T}$) calculus;
- Proof of factorial lower bound proof complexity for two state-of-the-art symbolic partial-order encodings of a simple, yet challenging concurrency problem;
- Experiments that confirm our theoretical lower bound.



DPLL($\mathcal{T}$) Lower Bound Proof Complexity

Informally, *non-interfering set of critical assignments* are propositionally satisfying assignments that contain minimal, disjoint $\mathcal{T}$-conflicts.

Theorem

Let ϕ be an unsatisfiable $\mathcal{T}$-formula, and Q be a non-interfering set of critical assignments for ϕ. Every Fixed-Alphabet-DPLL($\mathcal{T}$) proof that ϕ is UNSAT contains at least |Q| applications of $\mathcal{T}$-LEARN.

This theorem is a theoretical tool for proving lower bound proof complexity results in the DPLL($\mathcal{T}$) framework (as exhibited next).

The Concurrency Problem

The value at memory location x is initialized to zero, i.e. $[x] = 0$.

$$
\begin{array}{l|l|c|l}
\text{Thread } \mathtt{T}_0 & \text{Thread } \mathtt{T}_1 & & \text{Thread } \mathtt{T}_N \\
\hline
\textbf{local } v_0 := [x] & \textbf{local } v_1 := [x] & \ldots & \textbf{local } v_N := [x] \\
\textbf{assert}(v_0 \leq N) & [x] := v_1 + 1 & & [x] := v_N + 1
\end{array}
$$

$$
\begin{array}{c|c|c|c}
\text{Thread } \mathtt{T}_0 & \text{Thread } \mathtt{T}_1 & & \text{Thread } \mathtt{T}_N \\
\hline
r_0 & r_1 & \ldots & r_N \\
 & w_1 & & w_N
\end{array}
$$

For $N = 2$, if restricted to $\mathtt{T}_1 \parallel \mathtt{T}_2$, we get the following interleavings:

(1) $r_1 ; w_1 ; r_2 ; w_2$ (2) $r_1 ; r_2 ; w_1 ; w_2$ (3) $r_1 ; r_2 ; w_2 ; w_1$
(4) $r_2 ; r_1 ; w_1 ; w_2$ (5) $r_2 ; r_1 ; w_2 ; w_1$ (6) $r_2 ; w_2 ; r_1 ; w_1$.

In general, we get $(2N + 1)! \div 2^N$ interleavings for $\mathtt{T}_0 \parallel \mathtt{T}_1 \parallel \ldots \parallel \mathtt{T}_N$.

Concurrency Problem Encoding

We define the following preserved-program order (PPO) for the problem challenge $\mathtt{T}_0 \parallel \mathtt{T}_1 \parallel \ldots \parallel \mathtt{T}_N$:

$$\begin{tikzcd}
 & w_{init} \arrow[dl] \arrow[d] \arrow[drr] & & \\
r_0 & r_1 \arrow[d] & \dots & r_N \arrow[d] \\
 & w_1 & \dots & w_N
\end{tikzcd}$$

Let $\mathrm{R} \triangleq \{r_0, \ldots, r_N\}$ and $\mathrm{W} \triangleq \{w_{init}, w_0, \ldots, w_N\}$.

Our partial-order encodings are parameterized by three theories:

- $\mathcal{T}_C$: clock constraints, e.g. $\mathrm{c}_r \prec \mathrm{c}_w$ for $r \in \mathrm{R}$ and $w \in \mathrm{W}$,
- $\mathcal{T}_S$: selection constraints, e.g. $\mathrm{s}_r = \mathrm{s}_w$ for $r \in \mathrm{R}$ and $w \in \mathrm{W}$,
- $\mathcal{T}_V$: read constraints, e.g. rv_r is a unique $\mathcal{T}_V$-variable for $r \in \mathrm{R}$.

Cubic-size Encoding of Concurrency Problem

Let ϕ^3 be the $O(N^3)$ partial-order encoding of $\mathtt{T}_0 \parallel \mathtt{T}_1 \parallel \ldots \parallel \mathtt{T}_N$:

$$\underbrace{c_{w_{init}} \prec c_{r_{assert}} \wedge \bigwedge_{i=1\ldots N} c_{w_{init}} \prec c_{r_i} \prec c_{w_i}}_{\textbf{PPO}} \wedge \underbrace{\bigwedge_{w,w' \in W, w \neq w'} (c_w \prec c_{w'} \vee c_{w'} \prec c_w) \wedge s_w \neq s_{w'}}_{\textbf{WW}[x]} \wedge$$

$$\underbrace{\bigwedge_{w \in W, r \in R} (c_w \prec c_r \vee c_r \prec c_w)}_{\textbf{RW}[x]} \wedge \underbrace{\bigwedge_{r \in R} \left(\bigvee_{w \in W} s_w = s_r \right)}_{\textbf{RF}_{\textbf{TO}}[x]} \wedge \underbrace{rv_{r_{assert}} > N}_{\textbf{assert}(v_0 \leq N)} \wedge$$

$$\underbrace{\bigwedge_{w \in W, r \in R} (s_w = s_r) \Rightarrow c_w \prec c_r}_{\textbf{RF}^3[x]} \wedge \underbrace{\bigwedge_{r \in R} (s_{w_{init}} = s_r) \Rightarrow 0 = rv_r}_{\textbf{RF}^3[x]} \wedge \underbrace{\bigwedge_{i=1\ldots N, r \in R} (s_{w_i} = s_r) \Rightarrow rv_{r_i} + 1 = rv_r}_{\textbf{RF}^3[x]}$$

$$\underbrace{\bigwedge_{w,w' \in W, r \in R} (s_w = s_r \wedge c_w \prec c_{w'}) \Rightarrow c_r \prec c_{w'}}_{\textbf{FR}[x]}$$

Factorial Lower Bound DPLL($\mathcal{T}$) Proof Complexity

Theorem (Lower Bound for Cubic Partial-Order Encoding)
All Fixed-Alphabet-DPLL($\mathcal{T}$) proofs for the problem challenge encoded with ϕ^3 contain at least $N!$ applications of $\mathcal{T}$-LEARN.

We also have a quadratic-size partial-order encoding.[2]

This asymptotically smaller encoding has also at least factorial-sized DPLL($\mathcal{T}$) proofs for our challenge problem!

---

[2] *Horn and Kroening. On Partial Order Semantics for SAT/SMT-based Symbolic Encodings of Weak Memory Concurrency*. In: FORTE'15.

Experiments with Two Partial-Order Encodings

$\mathcal{E}^3$ and $\mathcal{E}^2$ are partial-order encodings of asymptotically different size, parameterized by three theories $\mathcal{T}_C$, $\mathcal{T}_S$ and $\mathcal{T}_V$.

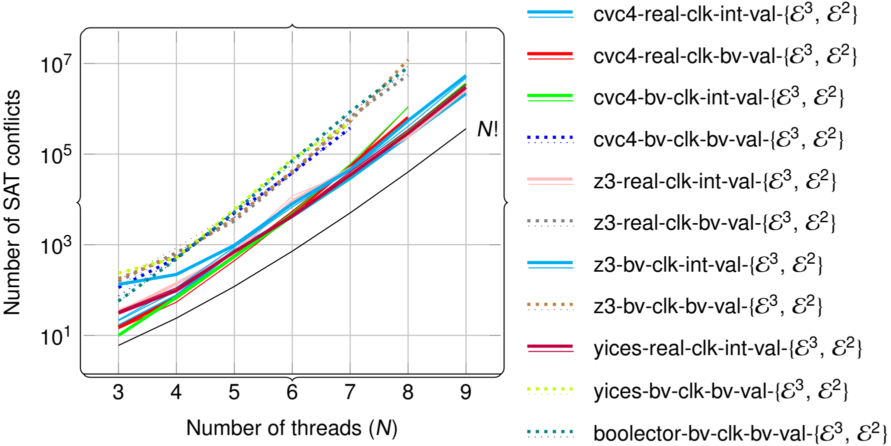
We instantiate $\langle \mathcal{T}_C, \mathcal{T}_S, \mathcal{T}_V \rangle$ to four configurations:

1. "real-clk-int-val": $\mathcal{T}_C = \mathcal{T}_S = \mathcal{T}_{\mathbb{R}}$ and $\mathcal{T}_V = \mathcal{T}_{\mathbb{Z}}$
2. "bv-clk-int-val": $\mathcal{T}_C = \mathcal{T}_S = \mathcal{T}_{\mathbb{BV}}$ and $\mathcal{T}_V = \mathcal{T}_{\mathbb{Z}}$
3. "real-clk-bv-val": $\mathcal{T}_C = \mathcal{T}_S = \mathcal{T}_{\mathbb{R}}$ and $\mathcal{T}_V = \mathcal{T}_{\mathbb{BV}}$
4. "bv-clk-bv-val": $\mathcal{T}_C = \mathcal{T}_S = \mathcal{T}_{\mathbb{BV}}$ and $\mathcal{T}_V = \mathcal{T}_{\mathbb{BV}}$

We use the following SMT solvers: Boolector, CVC4, Yices2, Z3.

Example: "z3-bv-clk-int-val-$\mathcal{E}^2$" denotes experiments with the $O(N^2)$ encoding using Z3 where $\mathcal{T}_C = \mathcal{T}_S = \mathcal{T}_{\mathbb{BV}}$ and $\mathcal{T}_V = \mathcal{T}_{\mathbb{Z}}$. We have a total of 56 SMT-LIB benchmarks. Timeout is 1 hour.

Experimental Results



Factorial growth of conflicts in `fkp2013-unsat` benchmark.

Concluding Remarks

- We have studied a simple, yet challenging concurrency problem.
- Our experiments provide an important diagnostic practice in the development of SMT encodings.
- The proofs we have manually inspected in CVC4 pinpoint value constraints as culprits.
- This way, our experiments can guide research into improving the performance of SMT solvers on such benchmarks.

The results of our work will shortly be published in SMT'15.

Concluding Remarks

- We have studied a simple, yet challenging concurrency problem.
- Our experiments provide an important diagnostic practice in the development of SMT encodings.
- The proofs we have manually inspected in CVC4 pinpoint value constraints as culprits.
- This way, our experiments can guide research into improving the performance of SMT solvers on such benchmarks.

The results of our work will shortly be published in SMT'15.

Thank you!

SC-relaxed Consistency Encoding

Let E be the set of events, $\ll$ be the PPO, $val : \mathrm{E} \to \mathcal{T}_V$-terms, $guard : \mathrm{E} \to \mathcal{T}_V$-formulas and L be the set of memory locations.

$$\mathbf{PPO} \triangleq \bigwedge \{(guard(e) \wedge guard(e')) \Rightarrow (\mathsf{c}_e \prec \mathsf{c}_{e'}) \mid e, e' \in \mathrm{E} : e \ll e'\}$$

$$\mathbf{WW}[x] \triangleq \bigwedge \{(\mathsf{c}_w \prec \mathsf{c}_{w'} \vee \mathsf{c}_{w'} \prec \mathsf{c}_w) \wedge \mathsf{s}_w \neq \mathsf{s}_{w'} \mid w, w' \in \mathrm{W}_x \wedge w \neq w'\}$$

$$\mathbf{RW}[x] \triangleq \bigwedge \{\mathsf{c}_w \prec \mathsf{c}_r \vee \mathsf{c}_r \prec \mathsf{c}_w \mid w \in \mathrm{W}_x \wedge r \in \mathrm{R}_x\}$$

$$\mathbf{RF_{TO}}[x] \triangleq \bigwedge \{guard(r) \Rightarrow \bigvee \{\mathsf{s}_w = \mathsf{s}_r \mid w \in \mathrm{W}_x\} \mid r \in \mathrm{R}_x\}$$

$$\mathbf{RF}^3[x] \triangleq \bigwedge \{(\mathsf{s}_w = \mathsf{s}_r) \Rightarrow (guard(w) \wedge val(w) = \mathsf{rv}_r \wedge \mathsf{c}_w \prec \mathsf{c}_r) \mid r \in \mathrm{R}_x \wedge w \in \mathrm{W}_x\}$$

$$\mathbf{FR}[x] \triangleq \bigwedge \{(\mathsf{s}_w = \mathsf{s}_r \wedge \mathsf{c}_w \prec \mathsf{c}_{w'} \wedge guard(w')) \Rightarrow (\mathsf{c}_r \prec \mathsf{c}_{w'}) \mid w, w' \in \mathrm{W}_x \wedge r \in \mathrm{R}_x\}$$

$$\mathcal{E}^3 \triangleq \bigwedge \left\{\mathbf{RF_{TO}}[x] \wedge \mathbf{RF}^3[x] \wedge \mathbf{FR}[x] \wedge \mathbf{WW}[x] \wedge \mathbf{RW}[x] \mid x \in L\right\} \wedge \mathbf{PPO}$$

$$\mathbf{RF}^2[x] \triangleq \bigwedge \{(\mathsf{s}_w = \mathsf{s}_r) \Rightarrow (\mathsf{c}_w = \mathsf{sup}_r \wedge guard(w) \wedge val(w) = \mathsf{rv}_r \wedge \mathsf{c}_w \prec \mathsf{c}_r) \mid r \in \mathrm{R}_x \wedge w \in \mathrm{W}_x\}$$

$$\mathbf{SUP}[x] \triangleq \bigwedge \{(\mathsf{c}_w \preceq \mathsf{c}_r \wedge guard(w)) \Rightarrow (\mathsf{c}_w \preceq \mathsf{sup}_r) \mid r \in \mathrm{R}_x \wedge w \in \mathrm{W}_x\}$$

$$\mathcal{E}^2 \triangleq \bigwedge \left\{\mathbf{RF_{TO}}[x] \wedge \mathbf{RF}^2[x] \wedge \mathbf{SUP}[x] \wedge \mathbf{WW}[x] \wedge \mathbf{RW}[x] \mid x \in L\right\} \wedge \mathbf{PPO}$$