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

A Problem Harder Than Diamonds

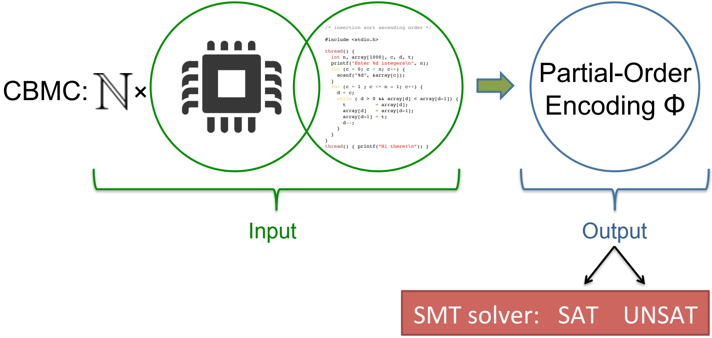
Liana Hadarean[1] **Alex Horn**[1] Tim King[2]

[1]University of Oxford

[2]Verimag

July 19, 2015

SAT/SMT-based Concurrency Verification



[CAV ’13] found bugs in:

Apache PostgreSQL

A Concurrency Problem

Example

Let the value at memory location x be initialized to 0, i.e. $[x] = 0$.

Thread $\mathtt{T}_0$		Thread $\mathtt{T}_1$				Thread $\mathtt{T}_N$
$\textbf{local } v_0 := [x]$ $\textbf{assert}(v_0 \leq N)$	$\parallel$	$\textbf{local } v_1 := [x]$ $[x] := v_1 + 1$	$\parallel$	$\ldots$	$\parallel$	$\textbf{local } v_N := [x]$ $[x] := v_N + 1$

A Concurrency Problem

Example

Let the value at memory location x be initialized to 0, i.e. $[x] = 0$.

Thread $\mathtt{T}_0$		Thread $\mathtt{T}_1$				Thread $\mathtt{T}_N$
local $v_0 := [x]$ **assert**$(v_0 \leq N)$	$\|\|$	**local** $v_1 := [x]$ $[x] := v_1 + 1$	$\|\|$	$\ldots$	$\|\|$	**local** $v_N := [x]$ $[x] := v_N + 1$

Claim (Partial-Order DPLL($\mathcal{T}$) Proof Complexity)

The size of DPLL($\mathcal{T}$) proofs* for this problem is at least $N!$ using partial-order encodings of concurrency**.

Our Contributions

- The concept of *non-interfering critical assignments*;
- A proof complexity theorem for Fixed-Alphabet DPLL($\mathcal{T}$);
- A factorial-size lower bound for the concurrency problem;
- Experiments with multiple SMT solvers & theory combinations.

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

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

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 (animation next)

$$\phi \triangleq (x < y \vee x = y) \wedge y < x$$

ϕ is a `QF_LIA`-unsatisfiable CNF formula.

- Fix $\mathcal{A} = \{x < y, x = y, y < x\}$ to be the alphabet of $\mathcal{T}$-atoms;
- let $\mathcal{X} = \{A, B, C\}$ be propositional variables;
- let $-^{\mathbb{B}} : \mathcal{A} \to \mathcal{X}$ be an injective function, e.g. $(x < y)^{\mathbb{B}} = A$.

Using $\mathcal{A}$, $\mathcal{X}$ and $-^{\mathbb{B}}$, we now illustrate Res and $\mathcal{T}$-learn.

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

$$\phi = (x < y \lor x = y) \land y < x$$

An execution of a lazy DPLL($\mathcal{T}$) solver (with a fixed-alphabet):

$$\rightsquigarrow (\underbrace{A}_{A \leftarrow \top} \lor \underbrace{B}_{B \leftarrow \bot}) \land \underbrace{C}_{C \leftarrow \top} \qquad \text{(Bool Assignment)}$$

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

$$\phi = (x < y \vee x = y) \wedge y < x$$

An execution of a lazy DPLL($\mathcal{T}$) solver (with a fixed-alphabet):

$$\rightsquigarrow (\underbrace{A}_{A \leftarrow \top} \vee \underbrace{B}_{B \leftarrow \bot}) \wedge \underbrace{C}_{C \leftarrow \top} \qquad \text{(Bool Assignment)}$$

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

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

$$\phi = (x < y \vee x = y) \wedge y < x$$

An execution of a lazy DPLL($\mathcal{T}$) solver (with a fixed-alphabet):

$$\rightsquigarrow (\underbrace{A}_{A \leftarrow \top} \vee \underbrace{B}_{B \leftarrow \bot}) \wedge \underbrace{C}_{C \leftarrow \top} \qquad \text{(Bool Assignment)}$$

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

$$\rightsquigarrow (\underbrace{A}_{A \leftarrow \bot} \vee \underbrace{B}_{B \leftarrow \top}) \wedge \underbrace{C}_{C \leftarrow \top} \wedge (\underbrace{\neg A}_{\top} \vee \underbrace{\neg C}_{\bot}) \qquad \text{(Bool Assignment)}$$

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

$$\phi = (x < y \vee x = y) \wedge y < x$$

An execution of a lazy DPLL($\mathcal{T}$) solver (with a fixed-alphabet):

$$\rightsquigarrow (\underbrace{A}_{A \leftarrow \top} \vee \underbrace{B}_{B \leftarrow \bot}) \wedge \underbrace{C}_{C \leftarrow \top} \qquad \text{(Bool Assignment)}$$

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

$$\rightsquigarrow (\underbrace{A}_{A \leftarrow \bot} \vee \underbrace{B}_{B \leftarrow \top}) \wedge \underbrace{C}_{C \leftarrow \top} \wedge (\underbrace{\neg A}_{\top} \vee \underbrace{\neg C}_{\bot}) \qquad \text{(Bool Assignment)}$$

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

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

$$\phi = (x < y \vee x = y) \wedge y < x$$

An execution of a lazy DPLL($\mathcal{T}$) solver (with a fixed-alphabet):

$\rightsquigarrow (\underbrace{A}_{A \leftarrow \top} \vee \underbrace{B}_{B \leftarrow \bot}) \wedge \underbrace{C}_{C \leftarrow \top}$ (Bool Assignment)

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

$\rightsquigarrow (\underbrace{A}_{A \leftarrow \bot} \vee \underbrace{B}_{B \leftarrow \top}) \wedge \underbrace{C}_{C \leftarrow \top} \wedge (\underbrace{\neg A}_{\top} \vee \underbrace{\neg C}_{\bot})$ (Bool Assignment)

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

$\rightsquigarrow \bot$ (i.e. formula is $\mathcal{T}$-unsatisfiable) (RES, multiple steps)

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

$$\phi = (x < y \vee x = y) \wedge y < x$$

An execution of a lazy DPLL($\mathcal{T}$) solver (with a fixed-alphabet):

$\rightsquigarrow (\underbrace{A}_{A \leftarrow \top} \vee \underbrace{B}_{B \leftarrow \bot}) \wedge \underbrace{C}_{C \leftarrow \top}$ (Bool Assignment)

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

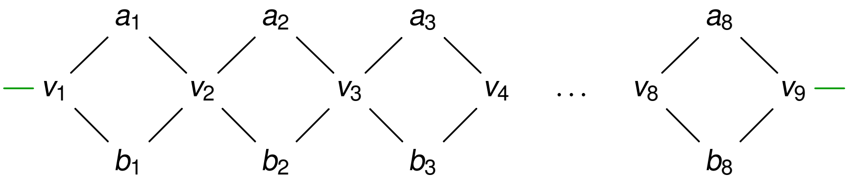
$\rightsquigarrow (\underbrace{A}_{A \leftarrow \bot} \vee \underbrace{B}_{B \leftarrow \top}) \wedge \underbrace{C}_{C \leftarrow \top} \wedge (\underbrace{\neg A}_{\top} \vee \underbrace{\neg C}_{\bot})$ (Bool Assignment)

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

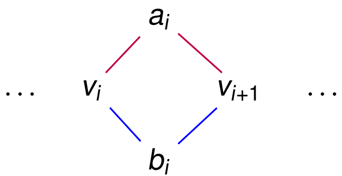
$\rightsquigarrow \bot$ (i.e. formula is $\mathcal{T}$-unsatisfiable) (RES, multiple steps)

Known Challenges for DPLL($\mathcal{T}$): Diamonds

Let $\phi_\diamond$ be $\left(\bigwedge_{i=1}^{8} (v_i = a_i \land a_i = v_{i+1}) \lor (v_i = b_i \land b_i = v_{i+1})\right) \land v_1 \neq v_9$.



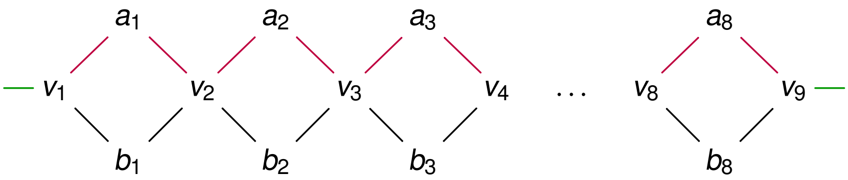
A diamond:



Let $\mathcal{X} = \left\{ \diagup_i, \diagdown_i, \diagdown_i, \diagup_i, — \;:\; 1 \leq i \leq 8 \right\}$, e.g. $\diagup_i$ denotes $v_i = a_i$.

Known Challenges for DPLL($\mathcal{T}$): Diamonds

Let $\phi_\diamond$ be $\left(\bigwedge_{i=1}^{8} (v_i = a_i \land a_i = v_{i+1}) \lor (v_i = b_i \land b_i = v_{i+1})\right) \land v_1 \neq v_9$.



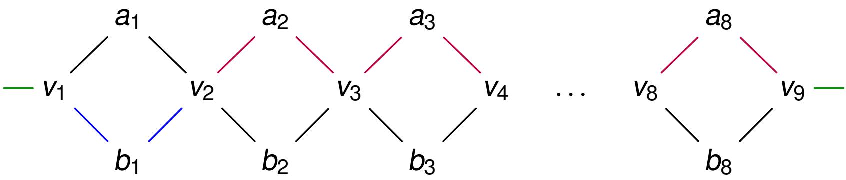
1. Let $M_1 = \left\{ \diagup_i, \diagdown_i : 1 \leq i \leq 8 \right\} \cup \{ - \}$.

2. Then $M_1 \models \phi_\diamond^{\mathbb{B}}$. But M_1 leads to a (unique) $\mathcal{T}$-conflict.

3. DPLL($\mathcal{T}$) learns the $\mathcal{T}$-lemma $\neg M_1$.

Known Challenges for DPLL($\mathcal{T}$): Diamonds

Let $\phi_\diamond$ be $\left(\bigwedge_{i=1}^{8} (v_i = a_i \wedge a_i = v_{i+1}) \vee (v_i = b_i \wedge b_i = v_{i+1})\right) \wedge v_1 \neq v_9$.



1. Let $M_2 = \left\{ \diagdown_1, \diagup_1 \right\} \cup \left\{ \diagup_i, \diagdown_i : 2 \leq i \leq 8 \right\} \cup \{ — \}$.
2. Then $M_2 \models \phi_\diamond^{\mathbb{B}}$. But M_2 leads to a (unique) $\mathcal{T}$-conflict.
3. DPLL($\mathcal{T}$) learns the $\mathcal{T}$-lemma $\neg M_2$.
4. Note that $\neg M_1$ is disjoint from $\neg M_2$.
5. In general, DPLL($\mathcal{T}$) enumerates 2^8 $\mathcal{T}$-conflicts [LPAR '08].

Our DPLL($\mathcal{T}$) Proof Complexity Theorem

A strict generalization of the diamonds problem:

Definition (Critical Assignments)

An assignment M is *critical* if $M \models \phi^{\mathbb{B}}$ and there is exactly one minimal $\mathcal{T}$-conflict $\neg L$ such that $\neg L^{\mathbb{B}} \subseteq M$. Let Q be a set of critical assignments for ϕ.

Our DPLL($\mathcal{T}$) Proof Complexity Theorem

A strict generalization of the diamonds problem:

Definition (Critical Assignments)

An assignment M is *critical* if $M \models \phi^{\mathbb{B}}$ and there is exactly one minimal $\mathcal{T}$-conflict $\neg L$ such that $\neg L^{\mathbb{B}} \subseteq M$. Let Q be a set of critical assignments for ϕ.

Definition (Non-interfering Critical Assignments)

Q is *non-interfering* if, for all $M_i \neq M_j$ in Q, $\neg L_i^{\mathbb{B}}$ isn't a subset of M_j.

Our DPLL($\mathcal{T}$) Proof Complexity Theorem

A strict generalization of the diamonds problem:

Definition (Critical Assignments)

An assignment M is *critical* if $M \models \phi^{\mathbb{B}}$ and there is exactly one minimal $\mathcal{T}$-conflict $\neg L$ such that $\neg L^{\mathbb{B}} \subseteq M$. Let Q be a set of critical assignments for ϕ.

Definition (Non-interfering Critical Assignments)

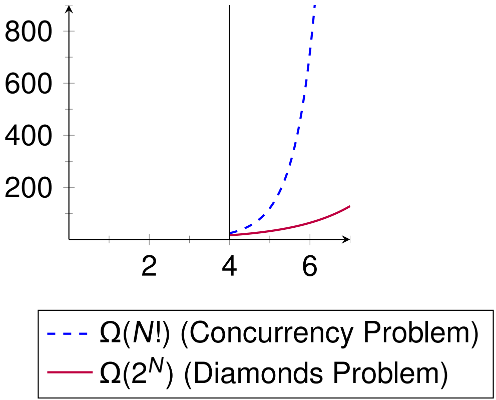
Q is *non-interfering* if, for all $M_i \neq M_j$ in Q, $\neg L_i^{\mathbb{B}}$ isn't a subset of M_j.

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

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.

A New Challenge Problem for SMT Community

We use the previous theorem to establish the factorial-size lower bound proof complexity of our concurrency problem challenge.



SMT Encodings of Concurrency Problem

Thread $\mathtt{T}_0$		Thread $\mathtt{T}_1$				Thread $\mathtt{T}_N$
r_0	$\parallel$	r_1 w_1	$\parallel$	$\ldots$	$\parallel$	r_N w_N

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

$$
\begin{array}{lll}
(1)\ r_1\,;w_1\,;r_2;w_2 & (2)\ r_1\,;r_2;w_1\,;w_2 & (3)\ r_1\,;r_2;w_2;w_1 \\
(4)\ r_2;r_1\,;w_1\,;w_2 & (5)\ r_2;r_1\,;w_2;w_1 & (6)\ r_2;w_2;r_1\,;w_1\,.
\end{array}
$$

Symbolically encode all interleavings, e.g. [CAV ’13, FORTE ’15].

SMT Encodings of Concurrency Problem

Thread $\texttt{T}_0$		Thread $\texttt{T}_1$				Thread $\texttt{T}_N$
r_0	$\|$	r_1 w_1	$\|$	$\dots$	$\|$	r_N w_N

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

Our encodings are parameterized by three SMT theories:

- $\mathcal{T}_C$: clocks
- $\mathcal{T}_S$: selections
- $\mathcal{T}_V$: value

Using $\mathcal{T}_C$, $\mathcal{T}_S$ and $\mathcal{T}_V$, we encode partial-order axioms (see next).

Clock Constraints

Example

Preserved-program order (PPO) for $\mathtt{T}_0 \parallel \mathtt{T}_1 \parallel \ldots \parallel \mathtt{T}_N$:

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

Example

By write consistency, writes in W are totally ordered in $\mathcal{T}_C$, e.g. either $w_1 \prec w_2$ or $w_2 \prec w_1$ in $\mathcal{T}_C$.

Selection and Value Constraints

Example

w_1

r_3 $\quad$ w_2

$$\begin{array}{c|c|c} \mathtt{T}_1 & \mathtt{T}_2 & \mathtt{T}_3 \\ \hline \underbrace{[x] := 1}_{w_1} & \underbrace{[x] := 2}_{w_2} & \underbrace{v := [x]}_{r_3} \end{array}$$

Selection and Value Constraints

Example

$$\begin{tikzcd} & w_1 \arrow[dl, "\mathbf{rf}"'] \\ r_3 & w_2 \end{tikzcd}$$

$$\begin{array}{ccccc} \mathtt{T}_1 & & \mathtt{T}_2 & & \mathtt{T}_3 \\ \hline \underbrace{[x] := 1}_{w_1} & \| & \underbrace{[x] := 2}_{w_2} & \| & \underbrace{v := [x]}_{r_3} \end{array}$$

To encode that "r_3 reads from w_1":

- $\mathrm{s}_{r_3} = \mathrm{s}_{w_1}$ in $\mathcal{T}_S$
- $\mathrm{v}_3 = 1$ in $\mathcal{T}_V$

Intuition: Factorial Lower Bound for Proof Size

Shuffle threads, e.g. for $\mathtt{T}_1, \mathtt{T}_2$ and $\mathtt{T}_3$ we get:

$$\mathtt{T}_1;\mathtt{T}_2;\mathtt{T}_3;\mathtt{T}_0 \qquad (\pi_1)$$
$$\mathtt{T}_2;\mathtt{T}_1;\mathtt{T}_3;\mathtt{T}_0 \qquad (\pi_2)$$
$$\mathtt{T}_2;\mathtt{T}_3;\mathtt{T}_1;\mathtt{T}_0 \qquad (\pi_3)$$
$$\ldots \qquad (\pi_k)$$

Each shuffling is satisfiable in $\mathcal{T}_C + \mathcal{T}_S$ but leads to a unique minimal $\mathcal{T}_V$-conflict:

$$\mathrm{v}_1 = 0 \land \mathrm{v}_2 = \mathrm{v}_1 + 1 \land \mathrm{v}_3 = \mathrm{v}_2 + 1 \land \mathrm{v}_{\mathrm{assert}} = \mathrm{v}_3 \land \mathrm{v}_{\mathrm{assert}} > N \qquad (\pi_1)$$
$$\mathrm{v}_2 = 0 \land \mathrm{v}_1 = \mathrm{v}_2 + 1 \land \mathrm{v}_3 = \mathrm{v}_1 + 1 \land \mathrm{v}_{\mathrm{assert}} = \mathrm{v}_3 \land \mathrm{v}_{\mathrm{assert}} > N \qquad (\pi_2)$$
$$\mathrm{v}_2 = 0 \land \mathrm{v}_3 = \mathrm{v}_2 + 1 \land \mathrm{v}_1 = \mathrm{v}_3 + 1 \land \mathrm{v}_{\mathrm{assert}} = \mathrm{v}_3 \land \mathrm{v}_{\mathrm{assert}} > N \qquad (\pi_3)$$
$$\ldots \qquad (\pi_k)$$

Factorial Lower Bound for Proof Size

Let ϕ^3 be a variant of the cubic-size encoding in [CAV '13] by our colleagues Alglave, Kroening and Tautschnig.

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 studied a quadratic-size partial-order encoding [FORTE '15]. Here, we show that this asymptotically smaller encoding has also at least factorial-sized DPLL($\mathcal{T}$) proofs!

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 $\mathcal{T}_V$.

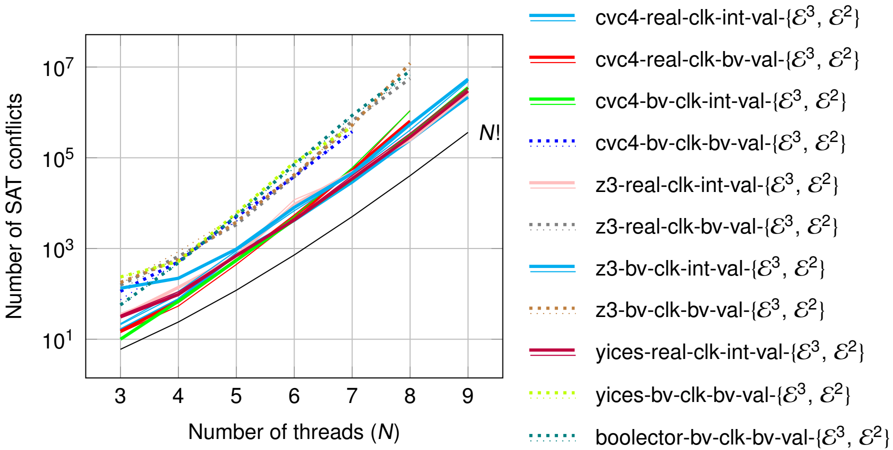
We instantiate $\langle \mathcal{T}_C, \mathcal{T}_S, \mathcal{T}_V \rangle$ to four configurations:

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

- A simple, yet challenging, SMT benchmark:

Thread $\mathtt{T}_0$		Thread $\mathtt{T}_1$				Thread $\mathtt{T}_N$
local v_0 := $[x]$ **assert**$(v_0 \leq N)$	$\parallel$	**local** v_1 := $[x]$ $[x]$:= $v_1 + 1$	$\parallel$	$\ldots$	$\parallel$	**local** v_N := $[x]$ $[x]$:= $v_N + 1$

- A new diagnosis tool for SMT encodings:
  1. Proof-size for DPLL($\mathcal{T}$) via non-interfering critical assignments
  2. $N!$ lower bound for two state-of-the-art partial-order encodings
  3. Theory and experiments pinpoint value constraints as culprit

Morgan Deters

Cubic-size Encoding of Concurrency Problem

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

$$\underbrace{\mathsf{c}_{w_{init}} \prec \mathsf{c}_{r_{assert}} \wedge \bigwedge_{i=1\ldots N} \mathsf{c}_{w_{init}} \prec \mathsf{c}_{r_i} \prec \mathsf{c}_{w_i}}_{\textbf{PPO}} \wedge \underbrace{\bigwedge_{w,w' \in \mathrm{W}, w \neq w'} (\mathsf{c}_w \prec \mathsf{c}_{w'} \vee \mathsf{c}_{w'} \prec \mathsf{c}_w) \wedge \mathsf{s}_w \neq \mathsf{s}_{w'}}_{\textbf{WW}[x]} \wedge$$

$$\underbrace{\bigwedge_{w \in \mathrm{W}, r \in \mathrm{R}} (\mathsf{c}_w \prec \mathsf{c}_r \vee \mathsf{c}_r \prec \mathsf{c}_w)}_{\textbf{RW}[x]} \wedge \underbrace{\bigwedge_{r \in \mathrm{R}} \left(\bigvee_{w \in \mathrm{W}} \mathsf{s}_w = \mathsf{s}_r \right)}_{\textbf{RF}_{\textbf{TO}}[x]} \wedge \underbrace{\mathsf{v}_{r_{assert}} > N}_{\textbf{assert}(v_0 \leq N)} \wedge$$

$$\underbrace{\bigwedge_{w \in \mathrm{W}, r \in \mathrm{R}} (\mathsf{s}_w = \mathsf{s}_r) \Rightarrow \mathsf{c}_w \prec \mathsf{c}_r}_{\textbf{RF}^3[x]} \wedge \underbrace{\bigwedge_{r \in \mathrm{R}} (\mathsf{s}_{w_{init}} = \mathsf{s}_r) \Rightarrow 0 = \mathsf{v}_r}_{\textbf{RF}^3[x]} \wedge \underbrace{\bigwedge_{i=1\ldots N, r \in \mathrm{R}} (\mathsf{s}_{w_i} = \mathsf{s}_r) \Rightarrow \mathsf{v}_{r_i} + 1 = \mathsf{v}_r}_{\textbf{RF}^3[x]}$$

$$\underbrace{\bigwedge_{w,w' \in \mathrm{W}, r \in \mathrm{R}} (\mathsf{s}_w = \mathsf{s}_r \wedge \mathsf{c}_w \prec \mathsf{c}_{w'}) \Rightarrow \mathsf{c}_r \prec \mathsf{c}_{w'}}_{\textbf{FR}[x]}$$

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.

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

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

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

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

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

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

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

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

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

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