A Concurrency Problem with Exponential DPLL(𝔽) Proofs
A Problem Harder Than Diamonds

Liana Hadarean¹  Alex Horn¹  Tim King²

¹University of Oxford
²Verimag

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. $[x] = 0$.

<table>
<thead>
<tr>
<th>Thread $T_0$</th>
<th>Thread $T_1$</th>
<th>Thread $T_N$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>local</strong> $v_0 := [x]$</td>
<td><strong>local</strong> $v_1 := [x]$</td>
<td><strong>local</strong> $v_N := [x]$</td>
</tr>
<tr>
<td>assert($v_0 \leq N$)</td>
<td>$[x] := v_1 + 1$</td>
<td>$[x] := v_N + 1$</td>
</tr>
</tbody>
</table>
A Concurrency Problem

Example
Let the value at memory location \( x \) be initialized to 0, i.e. \( [x] = 0 \).

<table>
<thead>
<tr>
<th>Thread ( T_0 )</th>
<th>Thread ( T_1 )</th>
<th>Thread ( T_N )</th>
</tr>
</thead>
<tbody>
<tr>
<td>local ( v_0 := [x] )</td>
<td>local ( v_1 := [x] )</td>
<td>local ( v_N := [x] )</td>
</tr>
<tr>
<td>assert( (v_0 \leq N) )</td>
<td>( [x] := v_1 + 1 )</td>
<td>( [x] := v_N + 1 )</td>
</tr>
</tbody>
</table>

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

Example (animation next)

\[
\phi \triangleq (x < y \lor x = y) \land y < x
\]

\(\phi\) 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 \(-^B : \mathcal{A} \to \mathcal{X}\) be an injective function, e.g. \((x < y)^B = A\).

Using \(\mathcal{A}, \mathcal{X}\) and \(-^B\), we now illustrate \(\text{Res}\) and \(\mathcal{T}\)-learn.
Example: Fixed-Alphabet DPLL(\(T\)) Proofs

\[ \phi = (x < y \lor x = y) \land y < x \]

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

\[ \sim (A \lor B) \land C \]

\(A \leftarrow \top\), \(B \leftarrow \bot\), \(C \leftarrow \top\) (Bool Assignment)

(i.e. formula is \(T\)-unsatisfiable) (Res, multiple steps)
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):

\[
\leadsto (A \lor B) \land C \land (\neg A \lor \neg C)
\]

(\(\mathcal{T}\)-Learn)

\[
\leadsto (A \lor B) \land C
\]

(Bool Assignment)

\[
A \leftarrow \top \quad B \leftarrow \bot \quad C \leftarrow \top
\]
Example: Fixed-Alphabet DPLL(\(T\)) Proofs

\[ \phi = (x < y \lor x = y) \land y < x \]

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

\[ \leadsto (A \lor B) \land C \land (\neg A \lor \neg C) \]

(Bool Assignment)

\[ \leadsto (A \lor B) \land C \land (\neg A \lor \neg C) \]

(\(T\)-LEARN)

\[ \leadsto (A \lor B) \land C \land (\neg A \lor \neg C) \]

(Bool Assignment)
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):

\[ \leadsto \left( \begin{array}{c}
A \\
B
\end{array} \right) \lor C \]

\[ A \leftarrow \top \]

\[ B \leftarrow \bot \]

\[ C \leftarrow \top \]

(Bool Assignment)

\[ \leadsto (A \lor B) \land C \land (\neg A \lor \neg C) \]

(\(\mathcal{T}\)-LEARN)

\[ \leadsto \left( \begin{array}{c}
A \\
B
\end{array} \right) \land C \land \left( \begin{array}{c}
\neg A \\
\neg C
\end{array} \right) \]

\[ A \leftarrow \bot \]

\[ B \leftarrow \top \]

\[ C \leftarrow \top \]

\[ \top \]

\[ \bot \]

(Bool Assignment)

(\(\mathcal{T}\)-LEARN)

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

φ = (x < y ∨ x = y) ∧ y < x

An execution of a lazy DPLL(Τ) solver (with a fixed-alphabet):

\[ \sim (A ∨ B) \land C \land (¬A ∨ ¬C) \]

(Bool Assignment)

\[ \sim (A ∨ B) \land C \land (¬A ∨ ¬C) \]

(Τ-LEARN)

\[ \sim (A ∨ B) \land C \land (¬A ∨ ¬C) \land (¬B ∨ ¬C) \]

(Τ-LEARN)

\[ \sim ⊥ \text{ (i.e. formula is } Τ\text{-unsatisfiable)} \]

(Res, multiple steps)
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):

$\leadsto (A \lor B) \land C \land (\neg A \lor \neg C)$  
$\quad$ (Bool Assignment)

$\quad$ ($\mathcal{T}$-Learn)

$\leadsto (A \lor B) \land C \land (\neg A \lor \neg C)$  
$\quad$ ($\mathcal{T}$-Learn)

$\leadsto (A \lor B) \land C \land (\neg A \lor \neg C) \land (\neg B \lor \neg C)$  
$\quad$ ($\mathcal{T}$-Learn)

$\leadsto \bot$ (i.e. formula is $\mathcal{T}$-unsatisfiable)  
$\quad$ (Res, multiple steps)
Known Challenges for DPLL($T$): Diamonds

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

A diamond:

Let $X = \{/_{i}, \backslash_{i}, \backslash_{i}, /_{i}, - : 1 \leq i \leq 8 \}$, e.g. $/_{i}$ denotes $v_i = a_i$. 
Known Challenges for DPLL(\(\mathcal{T}\)): Diamonds

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

1. Let \(M_1 = \left\{ \frac{v_i}{i}, \frac{a_i}{i} : 1 \leq i \leq 8 \right\} \cup \left\{ \frac{v_1}{1} \right\}.\)

2. Then \(M_1 \models \phi_\diamond^B.\) 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 \((\bigwedge_{i=1}^{8} (v_i = a_i \land a_i = v_{i+1}) \lor (v_i = b_i \land b_i = v_{i+1})) \land v_1 \neq v_9\).

1. Let \(M_2 = \left\{ \begin{array}{c}
  1, \\
  1
\end{array} \right\} \cup \left\{ \begin{array}{c}
  i, \\
  i : 2 \leq i \leq 8
\end{array} \right\} \cup \left\{ \begin{array}{c}
  \_
\end{array} \right\}.
2. Then \(M_2 \models \phi_{\Diamond}^{B}\). 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 \(M \models \phi^B\) and there is exactly one minimal \(\mathcal{T}\)-conflict \(\neg L\) such that \(\neg L^B \subseteq M\). Let \(Q\) be a set of critical assignments for \(\phi\).
Our DPLL(\(\mathcal{T}\)) Proof Complexity Theorem

A strict generalization of the diamonds problem:

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

**Definition (Non-interfering Critical Assignments)**
\(Q\) is *non-interfering* if, for all \(M_i \neq M_j\) in \(Q\), \(\neg L_i^B\) 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 \(M \models \phi^B\) and there is exactly one minimal \(\mathcal{T}\)-conflict \(\neg L\) such that \(\neg L^B \subseteq M\). Let \(Q\) be a set of critical assignments for \(\phi\).

**Definition (Non-interfering Critical Assignments)**
\(Q\) is *non-interfering* if, for all \(M_i \neq M_j\) in \(Q\), \(\neg L_i^B\) isn’t a subset of \(M_j\).

**Theorem (DPLL(\(\mathcal{T}\)) Lower Bound Proof Complexity)**
Let \(\phi\) be an unsatisfiable \(\mathcal{T}\)-formula, and \(Q\) be a non-interfering set of critical assignments for \(\phi\). Every Fixed-Alphabet-DPLL(\(\mathcal{T}\)) proof that \(\phi\) 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.

\[ \Omega(N!) \text{ (Concurrency Problem)} \]
\[ \Omega(2^N) \text{ (Diamonds Problem)} \]
SMT Encodings of Concurrency Problem

<table>
<thead>
<tr>
<th>Thread $T_0$</th>
<th>Thread $T_1$</th>
<th>Thread $T_N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_0$</td>
<td>$r_1$</td>
<td>$r_N$</td>
</tr>
<tr>
<td>$w_1$</td>
<td>...</td>
<td>$w_N$</td>
</tr>
</tbody>
</table>

For $N = 2$, if restricted to $T_1 \parallel T_2$, 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) $r_2; w_2; r_1; w_1$

Symbolically encode all interleavings, e.g. [CAV ’13, FORTE ’15].
SMT Encodings of Concurrency Problem

<table>
<thead>
<tr>
<th>Thread $T_0$</th>
<th>Thread $T_1$</th>
<th>Thread $T_N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_0$</td>
<td>$r_1$</td>
<td>$r_N$</td>
</tr>
<tr>
<td>$w_1$</td>
<td>$\ldots$</td>
<td>$w_N$</td>
</tr>
</tbody>
</table>

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

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 $T_0 \parallel T_1 \parallel \ldots \parallel T_N$:

```
<table>
<thead>
<tr>
<th></th>
<th>$w_{init}$</th>
<th>$r_0$</th>
<th>$r_1$</th>
<th>$r_N$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$w_1$</td>
<td>$\ldots$</td>
<td>$w_N$</td>
<td></td>
</tr>
</tbody>
</table>
```

Example
By write consistency, writes in $W$ are totally ordered in $\mathcal{T}_C$, e.g. either $w_1 < w_2$ or $w_2 < w_1$ in $\mathcal{T}_C$. 
Selection and Value Constraints

Example

\[
\begin{array}{cc}
  w_1 & T_1 \\
  r_3 & w_2 \\
  & T_2 \\
  & [x] := 1 \\
  & w_1 \\
  & v := [x] \\
  & T_3 \\
  & [x] := 2 \\
  & w_2 \\
  & r_3
\end{array}
\]
Selection and Value Constraints

Example

To encode that “$r_3$ reads from $w_1$”:

- $s_{r_3} = s_{w_1}$ in $\mathcal{T}_S$
- $v_3 = 1$ in $\mathcal{T}_V$
Intuition: Factorial Lower Bound for Proof Size

Shuffle threads, e.g. for $T_1, T_2$ and $T_3$ we get:

$$T_1; T_2; T_3; T_0$$ (\(\pi_1\))
$$T_2; T_1; T_3; T_0$$ (\(\pi_2\))
$$T_2; T_3; T_1; T_0$$ (\(\pi_3\))
$$\ldots$$ (\(\pi_k\))

Each shuffling is satisfiable in $T_C + T_S$ but leads to a unique minimal $T_V$-conflict:

$v_1 = 0 \land v_2 = v_1 + 1 \land v_3 = v_2 + 1 \land v_{\text{assert}} = v_3 \land v_{\text{assert}} > N$ (\(\pi_1\))
$v_2 = 0 \land v_1 = v_2 + 1 \land v_3 = v_1 + 1 \land v_{\text{assert}} = v_3 \land v_{\text{assert}} > N$ (\(\pi_2\))
$v_2 = 0 \land v_3 = v_2 + 1 \land v_1 = v_3 + 1 \land v_{\text{assert}} = v_3 \land v_{\text{assert}} > N$ (\(\pi_3\))
$$\ldots$$ (\(\pi_k\))
Factorial Lower Bound for Proof Size

Let $\phi^3$ 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($T$) proofs for the problem challenge encoded with $\phi^3$ contain at least $N!$ applications of $T$-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($T$) 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”: $\mathcal{T}_C = \mathcal{T}_S = \mathcal{T}_R$ and $\mathcal{T}_V = \mathcal{T}_Z$
2. “bv-clk-int-val”: $\mathcal{T}_C = \mathcal{T}_S = \mathcal{T}_\text{BV}$ and $\mathcal{T}_V = \mathcal{T}_Z$
3. “real-clk-bv-val”: $\mathcal{T}_C = \mathcal{T}_S = \mathcal{T}_R$ and $\mathcal{T}_V = \mathcal{T}_\text{BV}$
4. “bv-clk-bv-val”: $\mathcal{T}_C = \mathcal{T}_S = \mathcal{T}_\text{BV}$ and $\mathcal{T}_V = \mathcal{T}_\text{BV}$

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 $\mathcal{T}_C = \mathcal{T}_S = \mathcal{T}_\text{BV}$ and $\mathcal{T}_V = \mathcal{T}_Z$.

We have a total of 56 SMT-LIB benchmarks. Timeout is 1 hour.
Experimental Results

Factorial growth of conflicts in $\texttt{fkp2013-unsat}$ benchmark.
Concluding Remarks

• A simple, yet challenging, SMT benchmark:

\[
\begin{align*}
&\text{Thread } T_0 &\quad &\text{Thread } T_1 &\quad &\text{Thread } T_N \\
\text{local } v_0 := [x] &\quad &\text{local } v_1 := [x] &\quad &\text{local } v_N := [x] \\
\text{assert}(v_0 \leq N) &\quad &[x] := v_1 + 1 &\quad &\cdots &\quad &[x] := v_N + 1
\end{align*}
\]

• A new diagnosis tool for SMT encodings:
  1. Proof-size for DPLL(\(T\)) 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 $\phi^3$ be the $O(N^3)$ partial-order encoding of $T_0 \parallel T_1 \parallel \ldots \parallel T_N$:

\[
\begin{align*}
C_{W_{\text{init}}} < C_{r_{\text{assert}}} & \land \bigwedge_{i=1 \ldots N} C_{W_{\text{init}}} < C_{r_i} < C_{W_i} \land \bigwedge_{w, w' \in W, w \neq w'} (C_w < C_{w'} \lor C_{w'} < C_w) \land S_w \neq S_{w'} \land \\
\bigwedge_{w \in W, r \in R} (C_w < C_r \lor C_r < C_w) & \land \bigwedge_{r \in R} \left( \bigvee_{w \in W} S_w = S_r \right) \land v_{r_{\text{assert}}} > N \land \\
\bigwedge_{w \in W, r \in R} (S_w = S_r) & \Rightarrow C_w < C_r \land \bigwedge_{r \in R} (S_{W_{\text{init}}} = S_r) \Rightarrow 0 = v_r \land \bigwedge_{i=1 \ldots N, r \in R} (S_{W_i} = S_r) \Rightarrow v_{r_i} + 1 = v_r \\
\bigwedge_{w, w' \in W, r \in R} (S_w = S_r \land C_w < C_{w'}) & \Rightarrow C_r < C_{w'} \\
\end{align*}
\]

\text{PPO} \quad \text{WW}[x] \quad \text{RW}[x] \quad \text{RF}^3[x] \quad \text{RF}^3[x] \quad \text{RF}^3[x] 

\text{assert}(v_0 \leq N) \quad \text{RF}_{TO}[x]
SC-relaxed Consistency Encoding

Let $E$ be the set of events, $\ll$ be the PPO, $\text{val} : E \rightarrow T_V$-terms, $\text{guard} : E \rightarrow T_V$-formulas and $L$ be the set of memory locations.

$$
PPO \triangleq \bigwedge \{ (\text{guard}(e) \land \text{guard}(e')) \Rightarrow (c_e < c_{e'}) \mid e, e' \in E : e \ll e' \}
$$

$$
WW[x] \triangleq \bigwedge \{ (c_w < c_{w'} \lor c_{w'} < c_w) \land s_w \neq s_{w'} \mid w, w' \in W_x \land w \neq w' \}
$$

$$
RW[x] \triangleq \bigwedge \{ c_w < c_r \lor c_r < c_w \mid w \in W_x \land r \in R_x \}
$$

$$
RF_{\text{TO}}[x] \triangleq \bigwedge \{ \text{guard}(r) \Rightarrow \lor \{ s_w = s_r \mid w \in W_x \} \mid r \in R_x \}
$$

$$
RF^3[x] \triangleq \bigwedge \{ (s_w = s_r) \Rightarrow (\text{guard}(w) \land \text{val}(w) = v_r \land c_w < c_r) \mid r \in R_x \land w \in W_x \}
$$

$$
FR[x] \triangleq \bigwedge \{ (s_w = s_r \land c_w < c_{w'} \land \text{guard}(w')) \Rightarrow (c_r < c_{w'}) \mid w, w' \in W_x \land r \in R_x \}
$$

$$
E^3 \triangleq \bigwedge \{ RF_{\text{TO}}[x] \land RF^3[x] \land FR[x] \land WW[x] \land RW[x] \mid x \in L \} \land PPO
$$

$$
RF^2[x] \triangleq \bigwedge \{ (s_w = s_r) \Rightarrow (c_w = \sup_r \land \text{guard}(w) \land \text{val}(w) = v_r \land c_w < c_r) \mid r \in R_x \land w \in W_x \}
$$

$$
SUP[x] \triangleq \bigwedge \{ (c_w \leq c_r \land \text{guard}(w)) \Rightarrow (c_w \leq \sup_r) \mid r \in R_x \land w \in W_x \}
$$

$$
E^2 \triangleq \bigwedge \{ RF_{\text{TO}}[x] \land RF^2[x] \land SUP[x] \land WW[x] \land RW[x] \mid x \in L \} \land PPO
$$