Value-Centric Dynamic Partial Order Reduction

Krishnendu Chatterjee **Andreas Pavlogiannis** Viktor Toman

October 25, 2019

IST AUSTRIA
Institute of Science and Technology

AARHUS UNIVERSITY

Concurrency Bugs

Thread 1: Withdraw(x)

if balance $\geqslant x$ **then**
    balance $\leftarrow$ balance $- x$

Concurrency Bugs

Thread 1: Withdraw(x)

```
if balance ⩾ x then
    balance ← balance − x
```

Thread 2: Withdraw(x)

```
if balance ⩾ x then
    balance ← balance − x
```

Concurrency Bugs

Thread 1: Withdraw(x)

if balance $\geqslant x$ **then**
    balance $\leftarrow$ balance $- x$

Thread 2: Withdraw(x)

if balance $\geqslant x$ **then**
    balance $\leftarrow$ balance $- x$

Withdraw(5)   Withdraw(5)

balance $= 8$

Concurrency Bugs

Thread 1: Withdraw(x)

if balance $\geqslant x$ **then**
    balance $\leftarrow$ balance $- x$

Thread 2: Withdraw(x)

if balance $\geqslant x$ **then**
    balance $\leftarrow$ balance $- x$

Withdraw(5)

Withdraw(5)

balance $= 8$

Concurrency Bugs

Thread 1: Withdraw(x)

if balance $\geqslant x$ **then**
    balance $\leftarrow$ balance $- x$

Thread 2: Withdraw(x)

if balance $\geqslant x$ **then**
    balance $\leftarrow$ balance $- x$

Withdraw(5)

Withdraw(5)

balance $= 8$

Concurrency Bugs

Thread 1: Withdraw(x)

if balance $\geqslant x$ **then**
    balance $\leftarrow$ balance $- x$

Thread 2: Withdraw(x)

if balance $\geqslant x$ **then**
    balance $\leftarrow$ balance $- x$

Withdraw(5)

Withdraw(5)

$$\text{balance} = 8 \rightarrow 3$$

Concurrency Bugs

Thread 1: Withdraw(x)

if balance $\geqslant x$ **then**
    balance $\leftarrow$ balance $- x$

Thread 2: Withdraw(x)

if balance $\geqslant x$ **then**
    balance $\leftarrow$ balance $- x$

Withdraw(5)

Withdraw(5)

$$\text{balance} = 8 \rightarrow 3 \rightarrow -2$$

Concurrency Bugs

Thread 1: Withdraw(x)
if balance $\geqslant x$ **then**
balance $\leftarrow$ balance $- x$

Thread 2: Withdraw(x)
if balance $\geqslant x$ **then**
balance $\leftarrow$ balance $- x$

Withdraw(5) Withdraw(5)

$$\text{balance} = 8 \rightarrow 3 \rightarrow -2$$

- No control over scheduling
- "Heisenbugs" lie in scheduling subtleties
- Formal verification to the rescue

Concurrency Setting

- k deterministic threads
  - No randomization
  - Fixed inputs
- All nondeterministic behavior comes from the scheduler
- Goal local-state reachability: catch bugs, e.g. assertion violations

Concurrency Setting

- k deterministic threads
  - No randomization
  - Fixed inputs
- All nondeterministic behavior comes from the scheduler
- Goal local-state reachability: catch bugs, e.g. assertion violations
- Algorithmic problem: visit all local states of each process
- **Stateless:** cannot remember all system states

Concurrency Setting

- k deterministic threads
  - No randomization
  - Fixed inputs
- All nondeterministic behavior comes from the scheduler
- Goal local-state reachability: catch bugs, e.g. assertion violations

- Algorithmic problem: visit all local states of each process
- **Stateless:** cannot remember all system states

- Examine all traces
- n! many
- We can do better: DPOR

Commutative Events

$\underline{t_1}$	$\underline{t_2}$
$w(x)$	$w(x)$
$w(y)$	$w(y)$
$w(z)$	$w(y)$
$w(y)$	$w(z)$
$r(z)$	$r(z)$
$r(x)$	$r(x)$

Commutative Events

Definition

A pair of events (e_1, e_2) is **non-commutative** if

- e_1, e_2 use the same variable, and at last one is a write

t_1	t_2
$w(x)$	$w(x)$
$w(y)$	$w(y)$
$w(z)$	$w(y)$
$w(y)$	$w(z)$
$r(z)$	$r(z)$
$r(x)$	$r(x)$

Commutative Events

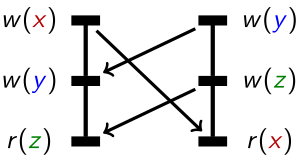
Definition

A pair of events (e_1, e_2) is **non-commutative** if

- e_1, e_2 use the same variable, and at last one is a write

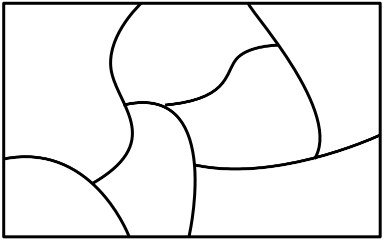
$\underline{t_1}$	$\sim_{\mathcal{HB}}$	$\underline{t_2}$
$w(x)$		$w(x)$
$w(y)$		$w(y)$
$w(z)$		$w(y)$
$w(y)$		$w(z)$
$r(z)$		$r(z)$
$r(x)$		$r(x)$

Happens-Before

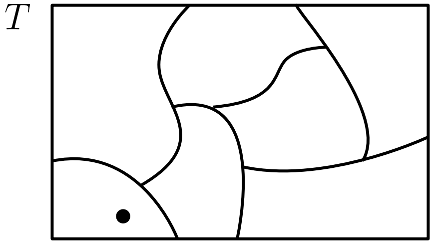


Dynamic Partial Order Reduction

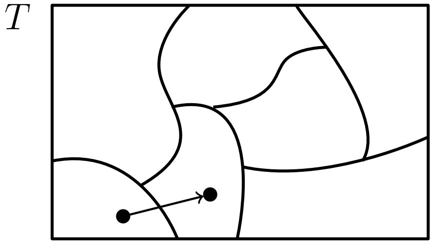
T



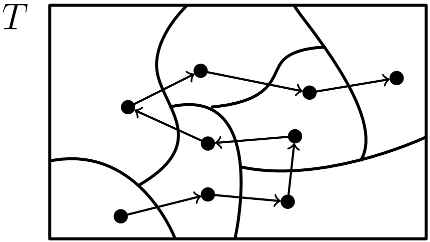
Dynamic Partial Order Reduction



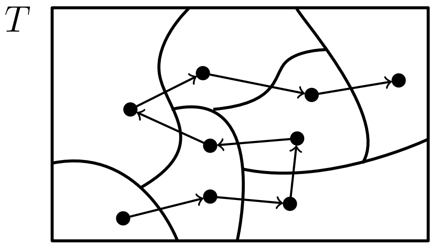
Dynamic Partial Order Reduction



Dynamic Partial Order Reduction



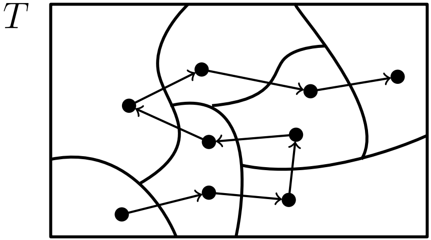
Dynamic Partial Order Reduction



Time: $O(\alpha \cdot \beta)$

- $\alpha = |\mathcal{T}/\sim|$
- $\beta =$ amortized time per class

Dynamic Partial Order Reduction



Time: $O(\alpha \cdot \beta)$

- $\alpha = |\mathcal{T}/\sim|$
- $\beta =$ amortized time per class

Question:

- How coarse can I make $\sim$
- while keeping $\beta = \text{poly}(n)$?

Motivation

Thread p_1 :	Thread p_2 :
1. $w(x, 1)$	1. $w(x, 1)$
	2. $r(x)$

Motivation

Thread p_1 :	Thread p_2 :
1. $w(x, 1)$	1. $w(x, 1)$
	2. $r(x)$

Happens-Before

t_1	t_2	t_3
w	w	w
w	w	r
r	r	w

Motivation

Thread p_1 :	Thread p_2 :
1. $w(x,1)$	1. $w(x,1)$
	2. $r(x)$

Happens-Before

t_1	t_2	t_3
w	w	w
w	w	r
r	r	w

Data-centric [Chalupa et al '18]

t_1	t_2
w	w
w	w
r	r

Motivation

Thread p_1 :	Thread p_2 :
1. $w(x,1)$	1. $w(x,1)$
	2. $r(x)$

Happens-Before

t_1	t_2	t_3
w	w	w
w	w	r
r	r	w

Data-centric [Chalupa et al '18]

t_1	t_2
w	w
w	w
r	r

This work

t_1
w
w
r

Technical Challenge

$$[A] \quad = \quad \begin{array}{ccc} \underline{t_1} & \underline{t_2} & \underline{t_3} \\ {\color{red}w} & {\color{blue}w} & {\color{blue}w} \\ {\color{blue}w} & {\color{red}w} & {\color{blue}r} \\ {\color{blue}r} & {\color{blue}r} & {\color{red}w} \end{array}$$

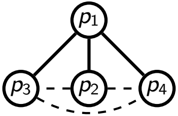
Realizability of Abstraction

Given an abstract object A, decide whether $[A] \neq \varnothing$.

Annotated Partial Orders

$\underline{\mathcal{A} = (X_1, X_2, P, \mathrm{GoodW})}$

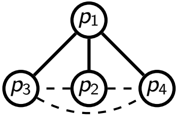
- X_1 has all events of p_1
- $P|X_2$ is a Happens-Before
- GoodW is the *good writes* function
  - $r \mapsto \{w_1, \ldots w_j\}$



Annotated Partial Orders

$\underline{\mathcal{A} = (X_1, X_2, P, \text{GoodW})}$

- X_1 has all events of p_1
- $P|X_2$ is a Happens-Before
- GoodW is the *good writes* function
  - $r \mapsto \{w_1, \ldots w_j\}$



Realizability of $\mathcal{A}$

Find if P can be linearized to a trace t such that every read sees a good write.

Closed Annotated Partial Orders

Definition (Closed Annotated Partial Orders)

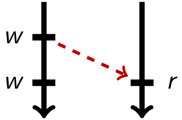
Call $\mathcal{A} = (X_1, X_2, P, \mathrm{GoodW})$ **closed** if for every read r

Closed Annotated Partial Orders

Definition (Closed Annotated Partial Orders)

Call $\mathcal{A} = (X_1, X_2, P, \mathrm{GoodW})$ **closed** if for every read r

1. There is a good write $w <_P r$.

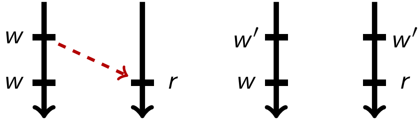


Closed Annotated Partial Orders

Definition (Closed Annotated Partial Orders)

Call $\mathcal{A} = (X_1, X_2, P, \mathrm{GoodW})$ **closed** if for every read r

1. There is a good write $w <_P r$.
2. There is a good maximal write for r.

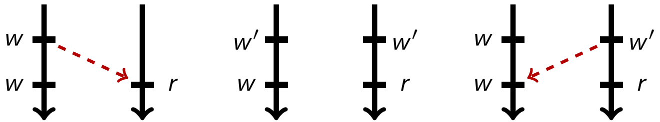


Closed Annotated Partial Orders

Definition (Closed Annotated Partial Orders)

Call $\mathcal{A} = (X_1, X_2, P, \text{GoodW})$ **closed** if for every read r

1. There is a good write $w <_P r$.
2. There is a good maximal write for r.
3. For every minimal bad write $w' <_P r$ there exists a good write w with $w' <_P w$.



Closure

Take annotated partial order $\mathcal{A} = (X_1, X_2, P, \mathrm{GoodW})$

Lemma

If $\mathcal{A}$ is closed then it is realizable.

- What if $\mathcal{A}$ is not closed?

Closure

Take annotated partial order $\mathcal{A} = (X_1, X_2, P, \mathrm{GoodW})$

Lemma

If $\mathcal{A}$ is closed then it is realizable.

- What if $\mathcal{A}$ is not closed?

Lemma

Either $\mathcal{A}$ is not realizable, or there is a unique minimal strengthening Q of P such that

- $\mathcal{B} = (X_1, X_2, Q, \mathrm{GoodW})$ *is closed*
- *Any witness for $\mathcal{B}$ is a witness for $\mathcal{A}$*

Definition (Closure)

Call $\mathcal{B}$ the closure of $\mathcal{A}$ (if it exists).

The Realizability of Annotated Partial Orders

Lemma (1)

An annotated partial order is realizable iff it has a closure.

The Realizability of Annotated Partial Orders

Lemma (1)

An annotated partial order is realizable iff it has a closure.

Lemma (2)

We can compute the closure of annotated partial orders in $O(\text{poly}(n))$ *time.*

$$\text{Lemma 1} + \text{Lemma 2} \implies \text{Realizability!}$$

Relax Happens-Before $\mathcal{HB}$ $\quad\mapsto\quad$ Value-Happens Before $\mathcal{VHB}$

Value-Centric Dynamic Partial Order Reduction

Relax Happens-Before $\mathcal{HB}$ $\quad\mapsto\quad$ Value-Happens Before $\mathcal{VHB}$

Theorem

$\sim_{\mathcal{VHB}}$ induces a partitioning that is at least as coarse as

- *the Happens-Before Partitioning [Abdulla et al '14]*
- *the Data-centric Partitioning [Chalupa et al '18]*

and can be exponentially coarser (value-based).

Value-Centric Dynamic Partial Order Reduction

Relax Happens-Before $\mathcal{HB}$ $\quad\mapsto\quad$ Value-Happens Before $\mathcal{VHB}$

Theorem

$\sim_{\mathcal{VHB}}$ *induces a partitioning that is at least as coarse as*

- *the Happens-Before Partitioning [Abdulla et al '14]*
- *the Data-centric Partitioning [Chalupa et al '18]*

and can be exponentially coarser (value-based).

Theorem

VC-DPOR *explores all local states and runs in time* $O(\alpha \cdot \beta)$, *where*

- $\alpha = |\mathcal{T}/ \sim_{\mathcal{VHB}}|$
- $\beta = \mathrm{poly}(n)$, *where* $n =$ *length of the longest trace in* $\mathcal{T}$.

Implementation & Experiments

- Implemented VC-DPOR
- Based on Nidhugg for LLVM IR

How to evaluate coarseness?

- **Source**, **Optimal** P. Abdulla et al. "Optimal Dynamic Partial Order Reduction". In: POPL. 2014
- **Optimal*** S. Aronis et al. "Optimal Dynamic Partial Order Reduction with Observers". In: *TACAS*. 2018
- **DC-DPOR** M. Chalupa et al. "Data-centric Dynamic Partial Order Reduction". In: POPL (2018)

Controlled Value Reduction

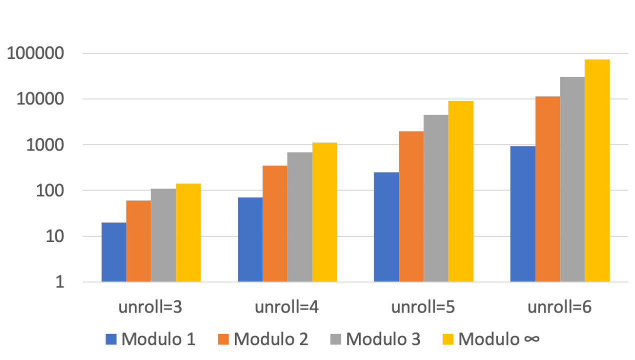


Figure: # Partitioning

Experiments 1: Mutual Exclusion

Benchmark	# Partitioning				
	VC-DPOR	Source	Optimal	Optimal*	DC-DPOR
tsay(2)	**2488**	7469	7469	7469	7469
tsay(3)	**241822**	1414576	1414576	1414576	1414576
tsay(4)	**24609389**	-	-	-	-
pet_fis(2)	**1371**	4386	4386	4386	4386
pet_fis(3)	**70448**	430004	430004	430004	430004
pet_fis(4)	**3747718**	-	-	-	-
burns(1)	**67**	849	849	849	849
burns(2)	**11297**	1490331	1490331	1490331	1490331
burns(3)	**1638338**	-	-	-	-

Benchmark	Time				
	VC-DPOR	Source	Optimal	Optimal*	DC-DPOR
tsay(2)	**0.81s**	2.46s	2.76s	2.99s	1.82s
tsay(3)	**1m38s**	10m2s	10m54s	12m1s	7m42s
tsay(4)	**3h51m**	-	-	-	-
pet_fis(2)	**0.69s**	1.56s	1.61s	1.73s	1.16s
pet_fis(3)	**34.03s**	2m54s	3m10s	3m31s	2m20s
pet_fis(4)	**41m31s**	-	-	-	-
burns(1)	**0.09s**	0.45s	0.40s	0.44s	0.49s
burns(2)	**16.27s**	16m49s	17m32s	20m4s	26m4s
burns(3)	**1h0m**	-	-	-	-

Experiments 2: SV-COMP

Benchmark	# Partitioning				
	VC-DPOR	Source	Optimal	Optimal*	DC-DPOR
parker(6)	**38670**	1100917	1100917	1023567	985807
parker(7)	**52465**	1735432	1735432	1613807	1554237
parker(8)	**68360**	2576147	2576147	2395947	2307467
Boop(6)	**248212**	35079696	35079696	4750426	1468774
Boop(7)	**420033**	-	-	10134616	2874202
Boop(8)	**677870**	-	-	20003512	5268064
scull_true(3)	**3426**	617706	617706	436413	172931
scull_true(4)	**8990**	2732933	2732933	1840022	656100
scull_true(5)	**19881**	9488043	9488043	6070688	1988798

Benchmark	Time				
	VC-DPOR	Source	Optimal	Optimal*	DC-DPOR
parker(6)	**1m29s**	23m5s	24m29s	24m54s	46m41s
parker(7)	**2m23s**	41m28s	44m41s	45m13s	1h27m
parker(8)	**3m35s**	1h9m	1h15m	1h17m	2h29m
Boop(6)	**3m26s**	2h54m	2h49m	26m22s	12m33s
Boop(7)	**6m33s**	-	-	1h0m	27m21s
Boop(8)	**11m54s**	-	-	2h7m	56m13s
scull_true(3)	**19.77s**	9m46s	10m22s	9m7s	4m46s
scull_true(4)	**1m7s**	51m37s	54m33s	46m12s	25m56s
scull_true(5)	**3m8s**	3h29m	3h42m	2h54m	1h47m

Experiments 3: Dynamic Programming

Benchmark	# Partitioning				
	VC-DPOR	Source	Optimal	Optimal*	DC-DPOR
rod_cut(7)	**4324**	102128	102128	51974	23143
rod_cut(8)	**14744**	508646	508646	257707	114624
rod_cut(9)	**50320**	2574752	-	1300067	577682
coin_min(8)	**46535**	1902262	1902262	981936	382275
coin_min(9)	**154663**	-	-	-	1634899
coin_min(11)	**1312252**	-	-	-	-
bin_nocon(7)	**13202**	1664672	1664672	471151	121350
bin_nocon(8)	**44802**	-	-	2825725	603668
bin_nocon(11)	**922114**	-	-	-	-

Benchmark	Time				
	VC-DPOR	Source	Optimal	Optimal*	DC-DPOR
rod_cut(7)	**33.23s**	4m14s	7m43s	3m47s	1m28s
rod_cut(8)	**3m4s**	27m32s	57m42s	28m2s	12m9s
rod_cut(9)	**17m24s**	3h0m	-	3h27m	1h39m
coin_min(8)	**3m0s**	1h13m	2h12m	1h12m	14m0s
coin_min(9)	**11m36s**	-	-	-	1h8m
coin_min(11)	**2h4m**	-	-	-	-
bin_nocon(7)	**29.57s**	48m34s	1h26m	26m11s	2m4s
bin_nocon(8)	**1m54s**	-	-	3h17m	12m32s
bin_nocon(11)	**1h0m**	-	-	-	-

Little Coarsening

Benchmark	# Partitioning				
	VC-DPOR	Source	Optimal	Optimal*	DC-DPOR
X2Tv9(3)	**7234**	7304	7304	7304	7304
X2Tv9(4)	**150535**	153725	153725	153725	153725
X2Tv9(5)	**3261067**	3324991	3324991	3324991	3324991
pthread5(1)	**20**	24	24	24	**20**
pthread5(2)	**1470**	1890	1890	1806	**1470**
pthread5(3)	**226800**	302400	302400	280800	**226800**

Benchmark	# Partitioning				
	VC-DPOR	Source	Optimal	Optimal*	DC-DPOR
X2Tv9(3)	2.53s	**2.11s**	2.23s	2.49s	2.41s
X2Tv9(4)	1m3s	**52.80s**	56.85s	1m3s	56.86s
X2Tv9(5)	29m53s	**22m17s**	24m10s	27m11s	27m10s
pthread5(1)	**0.05s**	**0.04s**	**0.04s**	0.06s	0.06s
pthread5(2)	0.67s	**0.38s**	0.45s	0.54s	0.67s
pthread5(3)	2m30s	**1m14s**	1m17s	1m17s	2m21s

Conclusion

- Stateless bounded model checking of concurrent programs
- New algorithm VC-DPOR
- A combination of Value-based + Happens-Before
- Efficient (poly-time) amortized exploration time
- Practical speedups

Thank you!

Questions?

Definition (CHB)

The Causally-Happens-Before partial order of a trace t is the weakest partial order $\mapsto_t$ s.t.

- $\mapsto_t \sqsubseteq \mathsf{TO}$
- $\mathsf{RF}_t(r) \mapsto_t r$

Definition (Side Function)

The side function S_t of a trace t is defined over the reads of the root thread, s.t.

$$S_t(r) = \begin{cases} 1, & \text{if } \mathsf{RF}_t(r) \text{ is local to } r \\ 2, & \text{otherwise} \end{cases}$$

VHB

Definition (VHB)

We have $t_1 \sim_{\mathcal{VHB}} t_2$ if

- $\mathsf{Events}(t_1) = \mathsf{Events}(t_2)$, $\mathsf{value}_{t_1} = \mathsf{value}_{t_2}$ and $S_{t_1} = S_{t_2}$
- $\mapsto_{t_1}|\mathsf{Reads} = \mapsto_{t_2}|\mathsf{Reads}$
- $\rightarrow_{t_1}|\mathcal{E}_{\neq p_1} = \rightarrow_{t_2}|\mathcal{E}_{\neq p_1}$ threads.