A Framework to Synergize Partial Order Reduction with State Interpolation

Duc-Hiep Chu and Joxan Jaffar

National University of Singapore (NUS)

Nov 20, 2014

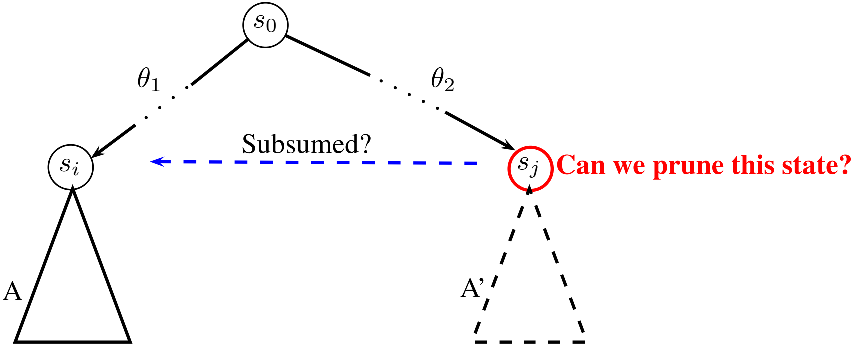
Traditional Partial Order Reduction (POR)

- Weaken the concept of a trace by abstracting the total order into a partial order
  - Two transitions are independent if their consecutive occurrences in a trace can be swapped without changing the final state
  - Two traces are equivalent if one can be transformed into another by repeatedly swapping adjacent independent transitions
  - For each class of equivalent traces, only one representative needs to be checked

Our Contributions

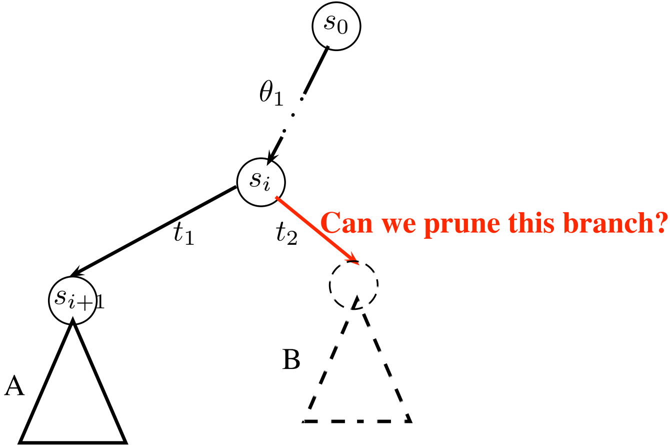
- Enable POR to work with symbolic search
- Synergize POR with State Interpolation (SI)
  - Replace the concept of trace equivalence with *trace coverage*
  - Weaken POR to Property Dependent POR (PDPOR)

State Interpolation: State Pruning



s_i and s_j share the same program point ℓ

POR: Branch Pruning



t_1 and t_2 emanate from the same state s_i

Trace Coverage

Definition (Trace Coverage)

Let ρ_1, ρ_2 be two traces of a concurrent program. We say ρ_1 *covers* ρ_2 wrt. a safety property ψ, denoted as $\rho_1 \sqsupseteq_\psi \rho_2$, iff $\rho_1 \models \psi \rightarrow \rho_2 \models \psi$. $\square$

- To replace the concept of trace equivalence
- The safety of one trace implies the safety of the other

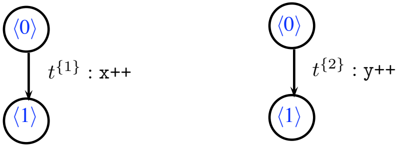
Property Dependent POR

Definition (Semi-Commutativity Relation)

Given a feasible derivation $s_0 \overset{\theta}{\Longrightarrow} s$, for all $t_1, t_2 \in \mathcal{T}$ which cannot dis-schedule each other, we say t_1 *semi-commutes* with t_2 *after state* s wrt. $\sqsupseteq_\psi$, denoted by $\langle s, t_1 \uparrow t_2, \psi \rangle$, iff for all $w_1, w_2 \in \mathcal{T}^*$, if $\theta w_1 t_1 t_2 w_2$ and $\theta w_1 t_2 t_1 w_2$ both are execution traces of the program, then we have $\theta w_1 t_1 t_2 w_2 \sqsupseteq_\psi \theta w_1 t_2 t_1 w_2$. □

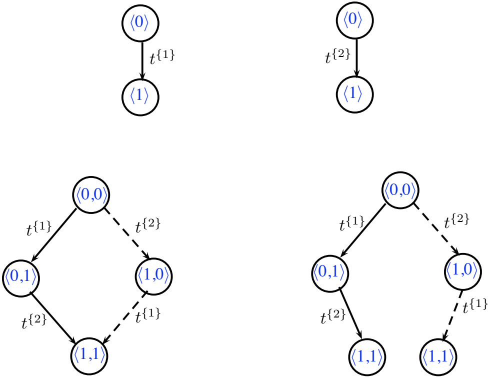
- To replace the concept of transition independence relation
- Traces with t_1 right before t_2 cover traces with t_1 right after t_2

Example: Independence vs. Semi-Commutativity



- $t^{\{1\}}$ is independent with $t^{\{2\}}$ wrt. deadlock verification
- $t^{\{1\}}$ is dependent with $t^{\{2\}}$ wrt. general safety property
- $t^{\{1\}}$ is semi-commutative with $t^{\{2\}}$ and vice versa wrt. safety property $\psi \equiv x + y \leq C$
- $t^{\{1\}}$ is semi-commutative with $t^{\{2\}}$ wrt. safety property $\psi \equiv x - y \leq C$, but not the other way around

Example: Independence vs. Semi-Commutativity



Property Dependent POR

Definition (New Persistent Set)

A set $T \subseteq \mathcal{T}$ of transitions enabled in a state s is *persistent in s* wrt. a property ψ iff, for all feasible derivation $s \xrightarrow{t_1} s_1 \xrightarrow{t_2} s_2 \ldots \xrightarrow{t_{m-1}} s_{m-1} \xrightarrow{t_m} s_m$ including only transitions $t_i \in \mathcal{T}$ and $t_i \notin T, 1 \leq i \leq m$, each transition in T *semi-commutes* with t_i after s wrt. $\sqsupseteq_\psi$. □

- Traces derived with transitions not in the persistent set first are covered by traces derived with transitions in the persistent set first

Property Dependent POR

- Selective search algorithm: at each state, we only consider transitions that belong to its persistent set

Theorem

The selective search algorithm with our new definition for persistent set is sound

- Given the semi-commutativity relation, to compute new persistent sets is similar to computing old persistent sets from the independence relation

Property Dependent POR

Definition (Semi-Commutative After A Program Point)

We say t_1 *semi-commutes* with t_2 *after program point* ℓ wrt. $\sqsupseteq_\psi$ and ϕ, denoted as $\langle \ell, \phi, t_1 \uparrow t_2, \psi \rangle$, iff for all feasible state $s \equiv \langle \ell, [\![s]\!] \rangle$ reachable from the initial state s_0, if $[\![s]\!] \models \phi$ then t_1 semi-commutes with t_2 after state s wrt. $\sqsupseteq_\psi$. □

Definition (Persistent Set Of A Program Point)

A set $T \subseteq \mathcal{T}$ of transitions schedulable at program point ℓ is *persistent at* ℓ under the trace-interpolant $\overline{\Psi}$ wrt. a property ψ iff, for all feasible derivation $s_0 \Longrightarrow s$ such that $s \equiv \langle \ell, [\![s]\!] \rangle$, if $[\![s]\!] \models \overline{\Psi}$ then for all feasible derivations $s \xrightarrow{t_1} s_1 \xrightarrow{t_2} s_2 \ldots \xrightarrow{t_{m-1}} s_{m-1} \xrightarrow{t_m} s_m$ including only transitions $t_i \in \mathcal{T}$ and $t_i \notin T, 1 \leq i \leq m$, each transition in T semi-commutes with t_i after state s wrt. $\sqsupseteq_\psi$. □

Algorithm

Assume safety property ψ and initial state s_0

function Explore($s \equiv \langle \ell, \cdot \rangle$)

$\langle 1 \rangle$ **if** (memoed($s, \overline{\Psi}$)) **return** $\overline{\Psi}$ /* Prune using state interpolation */

$\langle 2 \rangle$ **if** ($s \not\models \psi$) REPORT ERROR and TERMINATE

$\langle 3 \rangle$ $\overline{\Psi} := \psi$

$\langle 4 \rangle$ $\langle T, \overline{\Psi}_{trace} \rangle :=$ Persistent_Set(ℓ)

$\langle 5 \rangle$ **if** ($s \models \overline{\Psi}_{trace}$)

$\langle 6 \rangle$ Ts $:= T$

$\langle 7 \rangle$ $\overline{\Psi} := \overline{\Psi} \wedge \overline{\Psi}_{trace}$

$\langle 8 \rangle$ **else** Ts $:= Schedulable(s)$

$\langle 9 \rangle$ **foreach** t *in* (Ts $\setminus$ $Enabled(s)$) **do**

$\langle 10 \rangle$ $\overline{\Psi} := \overline{\Psi} \wedge \mathrm{pre}(t, \mathit{false})$

$\langle 11 \rangle$ **foreach** t *in* (Ts $\cap$ $Enabled(s)$) **do**

$\langle 12 \rangle$ $s \xrightarrow{t} s'$ /* Execute t */

$\langle 13 \rangle$ $\overline{\Psi}' :=$ Explore(s')

$\langle 14 \rangle$ $\overline{\Psi} := \overline{\Psi} \wedge \mathrm{pre}(t, \overline{\Psi}')$

$\langle 15 \rangle$ memo and **return** ($\overline{\Psi}$)

- We assume a persistent set and an associated trace interpolant $\overline{\Psi}_{trace}$ can be computed for each program point ℓ
- If $s \models \overline{\Psi}_{trace}$, we consider only those transitions in T (line $\langle 6 \rangle$)
- Note how $\overline{\Psi}_{trace}$ affects the final (memoed) interpolant $\overline{\Psi}$ (line $\langle 7 \rangle$)

Implementing Property Dependent POR

- It is about approximating the semi-commutativity relation
  - Syntactic conditions (as in traditional POR)
  - Semantic conditions for some classes of problem and simple properties
    - E.g. Proving bounds on resource usage
    - More in the paper
  - General algorithm (opportunistically) when the weakest preconditions are available (on going)

Experiments: Producers and Consumer

```
Initially x is 0;
N producers increment x; N producers double x;
the consumer consumes value of x; prove x ≤ N * 2^N
```

	POR		SI		POR+SI		PDPOR+SI	
N	States	T(s)	States	T(s)	States	T(s)	States	T(s)
2	449	0.03	514	0.17	85	0.03	10	0.01
3	18745	2.73	6562	2.43	455	0.19	14	0.01
4	986418	586.00	76546	37.53	2313	1.07	18	0.01
5	–	–	–	–	11275	5.76	22	0.01
6	–	–	–	–	53261	34.50	26	0.01
7	–	–	–	–	245775	315.42	30	0.01

Experiments: Sum of Ids

Comparing with the state-of-the-art

	POR = None		Kahlon *et al.* [2009] w. Z3			POR+SI = SI		PDPOR+SI	
N	States	T(s)	Conflicts	Decisions	T(s)	States	T(s)	States	T(s)
6	2676	0.44	1608	1795	0.08	193	0.05	7	0.01
8	149920	28.28	54512	59267	10.88	1025	0.27	9	0.01
10	–	–	–	–	–	5121	1.52	11	0.01
12	–	–	–	–	–	24577	8.80	13	0.01
14	–	–	–	–	–	114689	67.7	15	0.01

Experiments: Dining Philosophers (DP) and Bakery

	None		POR		SI		POR+SI	
Problem	States	T(s)	States	T(s)	States	T(s)	States	T(s)
`din-2(a)`	22	0.01	22	0.01	21	0.01	21	0.01
`din-3(a)`	1773	0.10	646	0.05	153	0.03	125	0.02
`din-4(a)`	–	–	155037	19.48	1001	0.17	647	0.09
`din-5(a)`	–	–	–	–	6113	1.01	4313	0.54
`din-6(a)`	–	–	–	–	35713	22.54	24201	4.16
`din-7(a)`	–	–	–	–	202369	215.63	133161	59.69
`bak-2`	86	0.05	48	0.03	38	0.03	31	0.02
`bak-3`	1755	3.13	1003	1.85	264	0.42	227	0.35
`bak-4`	47331	248.31	27582	145.78	1924	5.88	1678	4.95
`bak-5`	–	–	–	–	14235	73.69	12722	63.60

- Method by Kahlon *et al.* [2009] also performs safety verification on DP with a simpler property: Our approach is about 3 times faster
- To disprove a trivially unsafe property (b), we require only one trace ($<$ 0.1 seconds) while they, due to SMT encoding, required a similar amount of time compared to (a)

Experiments: Concurrent Programs from Cordeiro and Fischer [2011]

Comparing with SMT-based context-bounded (column C) model checking

		Cordeiro and Fischer [2011]		SI		PDPOR+SI	
Problem	LOC	C	T(s)	States	T(s)	States	T(s)
micro_2	247	17	1095	20201	10.88	201	0.04
stack	105	12	225	529	0.26	529	0.26
circular_buffer	111	∞	477	29	0.03	29	0.03
stateful20	60	10	95	1681	1.13	41	0.01

Questions & Answers

Trace Coverage

Definition (Equivalence)

Two traces are (Mazurkiewicz) *equivalent* iff one trace can be transformed into another by repeatedly swapping adjacent independent transitions. □

Definition (Trace Coverage)

Let ρ_1, ρ_2 be two traces of a concurrent program. We say ρ_1 *covers* ρ_2 wrt. a safety property ψ, denoted as $\rho_1 \sqsupseteq_\psi \rho_2$, iff $\rho_1 \models \psi \rightarrow \rho_2 \models \psi$. □

Property Dependent POR

Definition (Independence Relation)

$\mathcal{R} \subseteq \mathcal{T} \times \mathcal{T}$ is an *independence relation* iff for each $\langle t_1, t_2 \rangle \in \mathcal{R}$ the following properties hold for every state s:

1. if t_1 is enabled in s and $s \xrightarrow{t_1} s'$, then t_2 is enabled in s iff t_2 is enabled in s' ; and
2. if t_1 and t_2 are enabled in s, then there is a unique state s'' such that $s \overset{t_1 t_2}{\Longrightarrow} s''$ and $s \overset{t_2 t_1}{\Longrightarrow} s''$. □

Definition (Semi-Commutativity Relation)

Given a feasible derivation $s_0 \overset{\theta}{\Longrightarrow} s$, for all $t_1, t_2 \in \mathcal{T}$ which cannot dis-schedule each other, we say t_1 *semi-commutes* with t_2 *after state* s wrt. $\sqsupseteq_\psi$, denoted by $\langle s, t_1 \uparrow t_2, \psi \rangle$, iff for all $w_1, w_2 \in \mathcal{T}^*$, if $\theta w_1 t_1 t_2 w_2$ and $\theta w_1 t_2 t_1 w_2$ both are execution traces of the program, then we have $\theta w_1 t_1 t_2 w_2 \sqsupseteq_\psi \theta w_1 t_2 t_1 w_2$. □

Property Dependent POR

Definition (Old Persistent Set)

A set $T \subseteq \mathcal{T}$ of transitions enabled in a state s is *persistent in s* iff, for all feasible derivations $s \xrightarrow{t_1} s_1 \xrightarrow{t_2} s_2 \ldots \xrightarrow{t_{m-1}} s_{m-1} \xrightarrow{t_m} s_m$ including only transitions $t_i \in \mathcal{T}$ and $t_i \notin T$, $1 \leq i \leq m$, t_i is *independent* with all the transitions in T. □

Definition (New Persistent Set)

A set $T \subseteq \mathcal{T}$ of transitions enabled in a state s is *persistent in s* wrt. a property ψ iff, for all feasible derivation $s \xrightarrow{t_1} s_1 \xrightarrow{t_2} s_2 \ldots \xrightarrow{t_{m-1}} s_{m-1} \xrightarrow{t_m} s_m$ including only transitions $t_i \in \mathcal{T}$ and $t_i \notin T, 1 \leq i \leq m$, each transition in T *semi-commutes* with t_i after s wrt. $\sqsupseteq_\psi$. □

L. Cordeiro and B. Fischer. Verifying multi-threaded software using smt-based context-bounded model checking. In *ICSE*, 2011.

V. Kahlon, C. Wang, and A. Gupta. Monotonic partial order reduction: An optimal symbolic partial order reduction technique. In *CAV*, 2009.