Compositional Correctness and Completeness for Symbolic Partial Order Reduction

Åsmund Aqissiaq Arild Kløvstad* Eduard Kamburjan* Einar Broch Johnsen*

*University of Oslo
Department of Informatics

CONCUR, 2023-09-22

UNIVERSITY
OF OSLO

Some Motivation

- Program analysis and verification

Some Motivation

- Program analysis and verification
- Concurrent systems

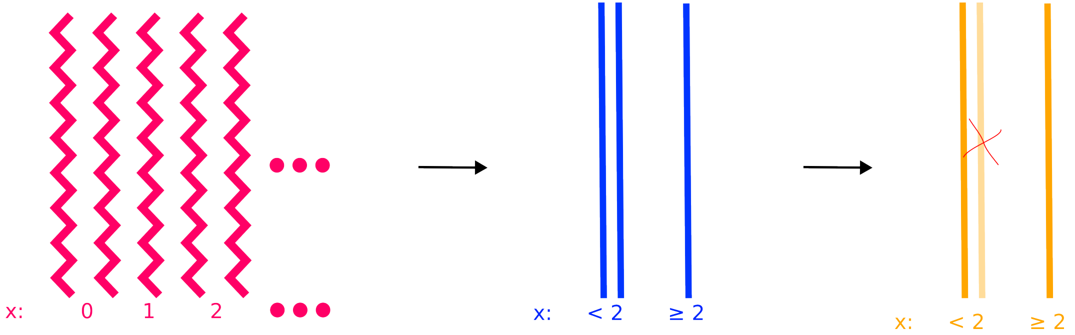
Some Motivation

- Program analysis and verification
- Concurrent systems
- Formalisms and correctness

Some Motivation

- Program analysis and verification
- Concurrent systems
- Formalisms and correctness
- State space explosion

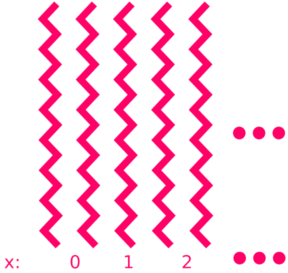
State Space Reduction

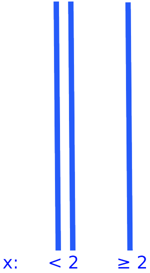


Program Analysis

- Formal
- Correct
- Efficient
- Compositional

Symbolic Execution

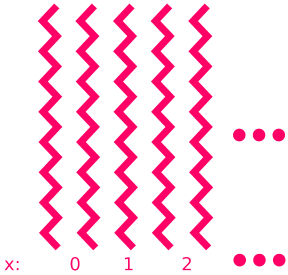


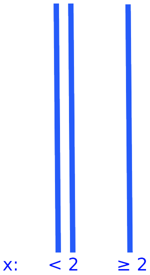


Idea

- Symbolic values
- Cover many inputs at once

Symbolic Execution





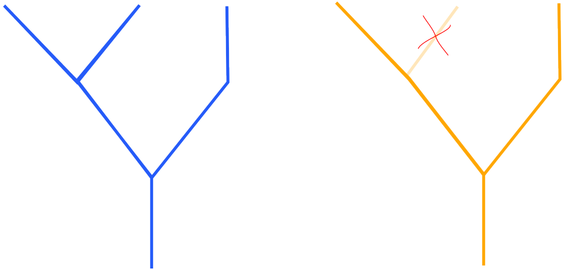
Idea

- Symbolic values
- Cover many inputs at once

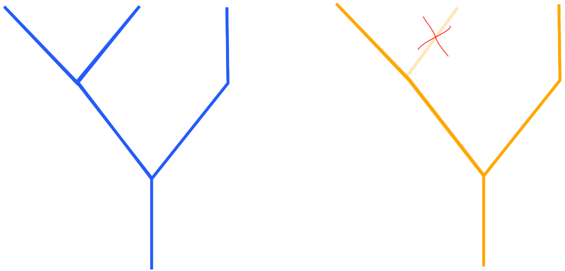
Uses

- Static analysis
- Directed testing
- Program synthesis

Partial Order Reduction



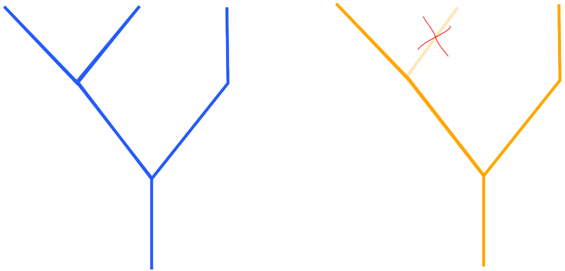
Partial Order Reduction



Idea

- Ignore equivalent paths
- Underapproximate by independence
- Dynamic and static approaches

Partial Order Reduction



Idea

- Ignore equivalent paths
- Underapproximate by independence
- Dynamic and static approaches

Uses

- Model checking
- Concurrency
- Concrete systems

Prior Work

$$
\begin{tikzcd}
\text{concrete semantics} \arrow[d, shift right, harpoon, "(1)"'] \\
\text{symbolic semantics} \arrow[u, shift right, harpoon] \arrow[r, shift left, harpoon] & \text{symbolic POR} \arrow[l, shift left, harpoon, "(2)"]
\end{tikzcd}
$$

1. De Boer & Bonsangue[1]
2. SymPaths[2]

[1] Symbolic Execution Formally Explained, Boer and Bonsangue (2021)
[2] SymPaths: Symbolic Execution Meets Partial Order Reduction, Boer, Bonsangue, et al. (2020)

Contribution

$$
\begin{tikzcd}
\text{concrete semantics} \arrow[r, shift left, harpoon] \arrow[d, shift left, harpoon, "(1)"'] \arrow[dr, shift left, harpoon, red] & \text{concrete POR} \arrow[l, shift left, harpoon] \arrow[d, shift left, harpoon] \\
\text{symbolic semantics} \arrow[r, shift left, harpoon, "(2)"'] \arrow[u, shift left, harpoon] & \text{symbolic POR} \arrow[l, shift left, harpoon] \arrow[u, shift left, harpoon] \arrow[ul, shift left, harpoon, red]
\end{tikzcd}
$$

- Formal
- Compositional
- Mechanized (in Coq)

A Small Language

`While` Language

- Arithmetic and boolean expressions

Expressions

$$e ::= x \mid n \mid e + e$$
$$b ::= x \mid \neg b \mid b \wedge b \mid e \leq e$$

A Small Language

While Language

- Arithmetic and boolean expressions
- Assignment, choice, loop

Expressions

$$e ::= x \mid n \mid e + e$$

$$b ::= x \mid \neg b \mid b \wedge b \mid e \leq e$$

Statements

$$s ::= \texttt{skip} \mid \texttt{x:= e} \mid \texttt{if}\ b\ \{s\}\ \{s\} \mid \texttt{while}\ b\ \{s\} \mid \texttt{s ; s} \mid \texttt{s} \parallel \texttt{s}$$

A Small Language

While Language

- Arithmetic and boolean expressions
- Assignment, choice, loop
- Sequential and parallel composition

Expressions

$$e ::= x \mid n \mid e + e$$
$$b ::= x \mid \neg b \mid b \wedge b \mid e \leq e$$

Statements

$$\begin{aligned} s ::= {} & \texttt{skip} & & \mid \texttt{x:= e} \\ & \mid \texttt{if } b \ \{s\} \ \{s\} & & \mid \texttt{while } b \ \{s\} \\ & \mid \texttt{s ; s} & & \mid \texttt{s} \parallel \texttt{s} \end{aligned}$$

Traces

- Concrete events: assignments

Traces

$$\tau_C ::= [\,] \mid \tau_C :: (x := e)$$

Traces

- Concrete events: assignments
- Symbolic events: assignments and boolean guards

Traces

$$\tau_C ::= [\,] \mid \tau_C :: (x := e)$$
$$\tau_S ::= [\,] \mid \tau_S :: (x := e) \mid \tau_S :: b$$

Traces

- Concrete events: assignments
- Symbolic events: assignments and boolean guards
- Final states

Traces

$$\tau_C ::= [\,] \mid \tau_C :: (x := e)$$
$$\tau_S ::= [\,] \mid \tau_S :: (x := e) \mid \tau_S :: b$$

$$\tau_C {\Downarrow_V} \triangleq \texttt{fold update } V\ \tau_C$$
$$\tau_S {\Downarrow_\sigma} \triangleq \texttt{fold update } \sigma\ \tau_S$$

Traces

- Concrete events: assignments
- Symbolic events: assignments and boolean guards
- Final states
- Path condition

Traces

$$\tau_C ::= [\,] \mid \tau_C :: (x := e)$$
$$\tau_S ::= [\,] \mid \tau_S :: (x := e) \mid \tau_S :: b$$

$$\tau_C {\Downarrow_V} \triangleq \texttt{fold update } V \ \tau_C$$
$$\tau_S {\Downarrow_\sigma} \triangleq \texttt{fold update } \sigma \ \tau_S$$
$$pc(\tau_S) \triangleq \texttt{fold } \wedge \ \texttt{true } \tau_S$$

(sort of)

Traces

- Concrete events: assignments
- Symbolic events: assignments and boolean guards
- Final states
- Path condition
- Abstraction

Trace Abstraction

τ_S V-abstracts τ_C if $\tau_C{\Downarrow_V} = V \circ \tau_S{\Downarrow_{id}}$

Traces

$$\tau_C ::= [\,] \mid \tau_C :: (x := e)$$
$$\tau_S ::= [\,] \mid \tau_S :: (x := e) \mid \tau_S :: b$$

$$\tau_C{\Downarrow_V} \triangleq \texttt{fold update } V \ \tau_C$$
$$\tau_S{\Downarrow_\sigma} \triangleq \texttt{fold update } \sigma \ \tau_S$$
$$pc(\tau_S) \triangleq \texttt{fold } \wedge \ \texttt{true } \tau_S$$

(sort of)

Transition Systems I

Concrete $(s, \tau_C) \Rightarrow_V^* (s', \tau'_C)$

$$(\texttt{x:} = \texttt{e}, \tau_C) \Rightarrow_V (\texttt{skip}, \tau_C :: (x := e))$$

$$(\texttt{if}\ \ b\ \{s_1\}\{s_2\}, \tau_C) \Rightarrow_V (s_1, \tau_C), \text{if } \tau_C \Downarrow_V (b) = \texttt{true}$$

$$\ldots$$

Transition Systems I

Concrete $(s, \tau_C) \Rightarrow^*_V (s', \tau'_C)$

$$(\texttt{x:} = \texttt{e}, \tau_C) \Rightarrow_V (\texttt{skip}, \tau_C :: (x := e))$$
$$(\texttt{if}\ \ b\ \{s_1\}\{s_2\}, \tau_C) \Rightarrow_V (s_1, \tau_C), \text{if } \tau_C \Downarrow_V (b) = \texttt{true}$$
$$\ldots$$

Symbolic $(s, \tau_S) \rightarrow^* (s', \tau'_S)$

$$(\texttt{x:} = \texttt{e}, \tau_S) \rightarrow (\texttt{skip}, \tau_S :: (x := e))$$
$$(\texttt{if}\ \ b\ \{s_1\}\{s_2\}, \tau_S) \rightarrow (s_1, \tau_S :: b)$$
$$(\texttt{if}\ \ b\ \{s_1\}\{s_2\}, \tau_S) \rightarrow (s_2, \tau_S :: \neg b)$$
$$\ldots$$

Transition Systems II

- Capture both parallel and sequential composition
- Fewer rules $\rightarrow$ shorter proofs

Contexts

$$C ::= \square \mid C; \mathtt{s} \mid C \parallel \mathtt{s} \mid \mathtt{s} \parallel C$$

C $\mathtt{s}$ "puts s in the hole"

Transition Systems II

- Capture both parallel and sequential composition
- Fewer rules $\to$ shorter proofs

Contexts

$$C ::= \square \mid C; \mathtt{s} \mid C \| \mathtt{s} \mid \mathtt{s} \| C$$

C $\mathtt{s}$ "puts s in the hole"

Transitions

$$\frac{(\mathtt{s}, t) \to (\mathtt{s}', t')}{(C\ \mathtt{s}, t) \to (C\ \mathtt{s}', t')}$$

Reflexive-transitive closure

Correct- and Completeness

$$(x := e, \tau_S) \rightarrow (skip, \tau_S :: (x := e)) \iff (x := e, \tau_C) \Rightarrow_V (skip, \tau_C :: (x := e))$$

Figure: If τ_S V-abstractsτ_C and $V \models pc(\tau_S)$, then rhs. also V-abstracts.

- Applies in arbitrary context
- Extends to reflexive-transitive closure by induction

Correct- and Completeness

$$(x := e, \tau_S) \rightarrow (skip, \tau_S :: (x := e)) \iff (x := e, \tau_C) \Rightarrow_V (skip, \tau_C :: (x := e))$$
$$(if\ \ b\ \{s_1\}\{s_2\}, \tau_S) \rightarrow (s_1, \tau_S :: b) \iff (if\ \ b\ \{s_1\}\{s_2\}, \tau_C) \Rightarrow_V (s_1, \tau_C)$$

Figure: If $\tau_S V$-abstractsτ_C and $V \models pc(\tau_S)$, then rhs. also V-abstracts.

- Applies in arbitrary context
- Extends to reflexive-transitive closure by induction

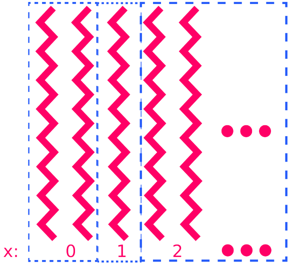
Correct- and Completeness

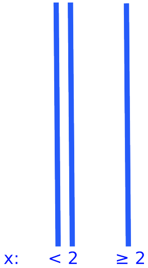
$$(x := e, \tau_S) \rightarrow (skip, \tau_S :: (x := e)) \iff (x := e, \tau_C) \Rightarrow_V (skip, \tau_C :: (x := e))$$
$$(if\ \ b\ \{s_1\}\{s_2\}, \tau_S) \rightarrow (s_1, \tau_S :: b) \iff (if\ \ b\ \{s_1\}\{s_2\}, \tau_C) \Rightarrow_V (s_1, \tau_C)$$
$$(if\ \ b\ \{s_1\}\{s_2\}, \tau_S) \rightarrow (s_1, \tau_S :: \neg b) \iff (if\ \ b\ \{s_1\}\{s_2\}, \tau_C) \Rightarrow_V (s_2, \tau_C)$$
$$\ldots$$

Figure: If τ_S V-abstractsτ_C and $V \models pc(\tau_S)$, then rhs. also V-abstracts.

- Applies in arbitrary context
- Extends to reflexive-transitive closure by induction

Correctness and Completeness – Intuition





Trace Equivalence

Symbolic Trace Equivalence

$\tau \sim \tau'$ if

- τ' is a permutation of τ
- $\tau{\Downarrow_\sigma} = \tau'{\Downarrow_\sigma}$ for all σ
- $V \models pc(\tau)$ iff $V \models pc(\tau')$ for all V

Trace Equivalence

Symbolic Trace Equivalence

$\tau \sim \tau'$ if

- τ' is a permutation of τ
- $\tau{\Downarrow_\sigma} = \tau'{\Downarrow_\sigma}$ for all σ
- $V \models pc(\tau)$ iff $V \models pc(\tau')$ for all V

- Overapproximates possible transitions
- Equivalent for concrete traces ($\simeq$)

POR Transition System

Symbolic POR Transition Rule

$$\frac{(s_0, \tau_0) \rightarrow (s, \tau) \qquad \tau_0 \sim \tau_0'}{(s_0, \tau_0') \rightarrow^{POR} (s, \tau)}$$

POR Transition System

Symbolic POR Transition Rule

$$\frac{(s_0, \tau_0) \rightarrow (s, \tau) \qquad \tau_0 \sim \tau_0'}{(s_0, \tau_0') \rightarrow^{POR} (s, \tau)}$$

- Allows picking a different trace to extend
- Equivalent for concrete traces

POR-Bisimulation

Assume $\tau_0 \sim \tau_0'$, then

Correctness

If $(s, \tau_0) \rightarrow (s', \tau)$ there exists $(s, \tau_0') \rightarrow^{POR} (s', \tau')$ with $\tau \sim \tau'$

POR-Bisimulation

Assume $\tau_0 \sim \tau_0'$, then

Correctness

If $(s, \tau_0) \rightarrow (s', \tau)$ there exists $(s, \tau_0') \rightarrow^{POR} (s', \tau')$ with $\tau \sim \tau'$

Completeness

If $(s, \tau_0) \rightarrow^{POR} (s', \tau)$ there exists $(s, \tau_0') \rightarrow (s', \tau')$ with $\tau \sim \tau'$

Important Properties

Lemma: Trace Step

For equivalent traces $\tau \sim \tau'$, if $(s, \tau) \rightarrow (s_1, \tau_1)$ then there exists τ_2 such that $(s, \tau') \rightarrow (s_1, \tau_2)$ and $\tau_1 \sim \tau_2$

Intuitively: step relation respects trace equivalence

Important Properties

Lemma: Trace Step

For equivalent traces $\tau \sim \tau'$, if $(s, \tau) \rightarrow (s_1, \tau_1)$ then there exists τ_2 such that $(s, \tau') \rightarrow (s_1, \tau_2)$ and $\tau_1 \sim \tau_2$

Intuitively: step relation respects trace equivalence

Lemma: Abstraction Congruence

Given $\tau_S \sim \tau'_S$ and $\tau_C \simeq \tau'_C$, if τ_S V-abstracts τ_C, then τ'_S V-abstracts τ'_C

Intuitively: V-abstraction respects trace equivalence

Putting it Together

For initial τ_S that V-abstracts τ_C:

Total Correctness

If $(s, \tau_S) \rightarrow^{POR} (s', \tau'_S)$ and $V \models pc(\tau'_S)$, there exists $(s, \tau_C) \Rightarrow_V (s', \tau'_C)$ such that τ'_S V-abstracts τ'_C

Putting it Together

For initial τ_S that V-abstracts τ_C:

Total Correctness

If $(s, \tau_S) \rightarrow^{POR} (s', \tau'_S)$ and $V \models pc(\tau'_S)$, there exists $(s, \tau_C) \Rightarrow_V (s', \tau'_C)$ such that τ'_S V-abstracts τ'_C

Total Completeness

If $(s, \tau_C) \Rightarrow_V (s', \tau'_C)$ there exists $(s, \tau_S) \rightarrow^{POR} (s', \tau'_S)$ such that τ'_S V-abstracts τ'_C

Recap and Future Work

Recap

We have:

- Defined concrete and symbolic semantics,
- Related them by correct- and completeness,
- Defined a notion of trace equivalence,
- Used it to define POR-semantics,
- Composed the two for Correct and Complete Symbolic POR, and
- Mechanized the development in Coq

Recap and Future Work

Recap

We have:

- Defined concrete and symbolic semantics,
- Related them by correct- and completeness,
- Defined a notion of trace equivalence,
- Used it to define POR-semantics,
- Composed the two for Correct and Complete Symbolic POR, and
- Mechanized the development in Coq

Future Work

- Extract verified symbolic execution engine
- Other kinds of (composable) reduction techniques
- More complex languages

Recap and Future Work

Recap

We have:

- Defined concrete and symbolic semantics,
- Related them by correct- and completeness,
- Defined a notion of trace equivalence,
- Used it to define POR-semantics,
- Composed the two for Correct and Complete Symbolic POR, and
- Mechanized the development in Coq

Future Work

- Extract verified symbolic execution engine
- Other kinds of (composable) reduction techniques
- More complex languages

Thank you, questions?