Model Checking for Weakly Consistent Libraries

Michalis Kokologiannakis Azalea Raad Viktor Vafeiadis

June 24, 2019

Max Planck Institute for Software Systems (MPI-SWS)

How Do We Verify Concurrent Programs?

How Do We Verify Concurrent Programs?

Stateless Model Checking (SMC): enumerates all executions

- without explicitly *storing* the visited states

Challenges:

- State space explosion
- Weak memory

Challenge #1: State Space Explosion

Challenge #1: State Space Explosion

$[x = y = 0]$
$x := 1 \parallel x := 2 \parallel y := 42$

Executions:

Challenge #1: State Space Explosion

$[x = y = 0]$
$x := 1 \parallel x := 2 \parallel y := 42$

Executions:

SMC : 6

Challenge #1: State Space Explosion

$[x = y = 0]$
$x := 1 \parallel x := 2 \parallel y := 42$

Executions:

SMC : 6

SMC+POR$^{\text{mo}}$: 2

Challenge #1: State Space Explosion

$[x = y = 0]$		
$x := 1$	$x := 2$	$y := 42$

Executions:

SMC : 6

SMC+POR$^{\texttt{mo}}$: 2

SMC+POR$^{\texttt{porf}}$: 1

Challenge #2: Weak Memory Models

Challenge #2: Weak Memory Models

All current techniques are memory-model specific

$\Rightarrow$ with the exception of `herd`

Challenge #2: Weak Memory Models

All current techniques are memory-model specific

$\Rightarrow$ with the exception of `herd`

What memory model **properties** are sufficient for efficient SMC?

Our contribution

Our contribution

- We present sufficient properties for efficient SMC

Our contribution

- We present sufficient properties for efficient SMC
- **GenMC:** an SMC procedure
  - parametric in the choice of the memory model

Our contribution

- We present sufficient properties for efficient SMC
- **GenMC:** an SMC procedure
  - parametric in the choice of the memory model
  - sound, complete, optimal, and efficient

Our contribution

- We present sufficient properties for efficient SMC
- **GenMC:** an SMC procedure
    - parametric in the choice of the memory model (+ libraries!)
    - sound, complete, optimal, and efficient

Generic Model Checking

Goal: Enumerate all consistent execution graphs of P for *any* memory model.

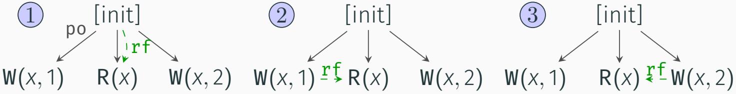
Goal: Enumerate all consistent execution graphs of P for *any* memory model.

$[x = 0]$
$x := 1 \parallel a := x \parallel x := 2$

GenMC: Generic Model Checking

Goal: Enumerate all consistent execution graphs of *P* for *any* memory model.

$[x = 0]$		
$x := 1$	$a := x$	$x := 2$



Systematically Enumerate All Graphs

Systematically Enumerate All Graphs

$$[x = y = 0]$$

$$\begin{array}{l} a := y \\ x := a \end{array} \;\Bigg\|\; \begin{array}{l} b := x \\ y := b \end{array}$$

Systematically Enumerate All Graphs

$$[x = y = 0]$$

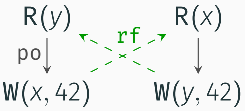
$$\begin{array}{l|l} a := y & b := x \\ x := a & y := b \end{array}$$

Can the reads of this program read 42?

Systematically Enumerate All Graphs

$$[x = y = 0]$$

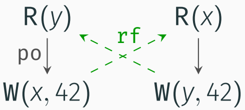
$$\begin{array}{l||l} a := y & b := x \\ x := a & y := b \end{array}$$



Can the reads of this program read 42?

Systematically Enumerate All Graphs

$[x = y = 0]$	
$a := y$	$b := x$
$x := a$	$y := b$



Can the reads of this program read 42?

The number of executions may be infinite!

Goal: Enumerate all consistent execution graphs of P for *any* memory model

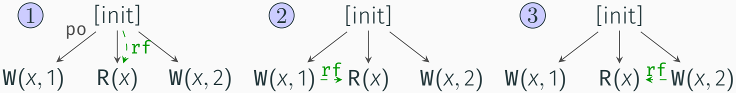
Goal: Enumerate all consistent execution graphs of P for *any* memory model , where

- po $\cup$ `rf` is acyclic

Checking Consistency At Each Step

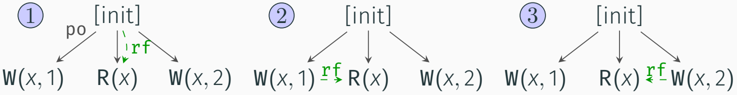
Checking Consistency At Each Step

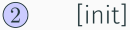
$[x = 0]$		
$x := 1$	$a := x$	$x := 2$



Checking Consistency At Each Step

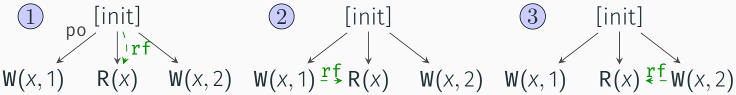
$[x = 0]$		
$x := 1$	$a := x$	$x := 2$

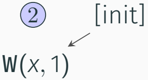




Checking Consistency At Each Step

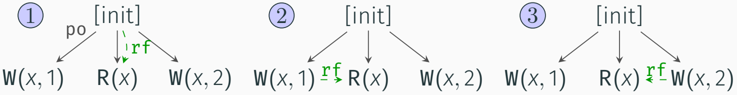
$[x = 0]$		
$x := 1$	$a := x$	$x := 2$

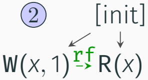




Checking Consistency At Each Step

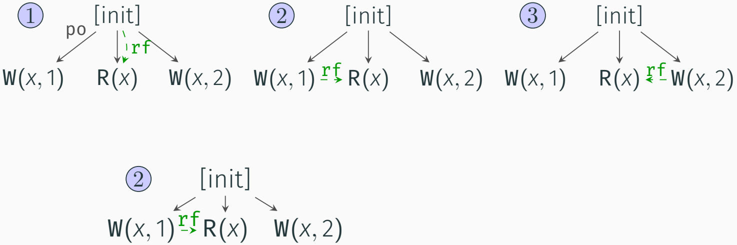
$[x = 0]$		
$x := 1$	$a := x$	$x := 2$





Checking Consistency At Each Step

$[x = 0]$
$x := 1 \parallel a := x \parallel x := 2$



Goal: Enumerate all consistent execution graphs of *P* for *any* memory model, where

- $\mathrm{po} \cup \mathtt{rf}$ is acyclic

Goal: Enumerate all consistent execution graphs of P for *any* memory model, where

- $\mathsf{po} \cup \mathtt{rf}$ is acyclic
- Each execution can be obtained from some linear extension of $\mathsf{po} \cup \mathtt{rf}$

GenMC: Generic Model Checking

Goal: Enumerate all consistent execution graphs of P for *any* memory model, where

- $\mathsf{po} \cup \mathtt{rf}$ is acyclic
- Each execution can be obtained from ~~some~~ every linear extension of $\mathsf{po} \cup \mathtt{rf}$

Goal: Enumerate all consistent execution graphs of *P* for *any* memory model, where

- $\mathsf{po} \cup \mathtt{rf}$ is acyclic
- Consistency is prefix-closed

Fixing The Construction Order

Fixing The Construction Order

$[x = 0]$		
$x := 1$	$a := x$	$x := 2$

Fixing The Construction Order

$[x = 0]$
$x := 1 \parallel a := x \parallel x := 2$

[init]

Fixing The Construction Order

$[x = 0]$
$x := 1 \;\Big\|\; a := x \;\Big\|\; x := 2$

$$\texttt{W}(x, 1) \xleftarrow{} \text{[init]}$$

Fixing The Construction Order

$[x = 0]$
$x := 1 \parallel a := x \parallel x := 2$

$$\begin{array}{c} \phantom{\mathtt{W}(x,1) \xrightarrow{\mathtt{rf}}} \text{[init]} \\ \mathtt{W}(x, 1) \xrightarrow{\mathtt{rf}} \mathtt{R}(x) \end{array}$$

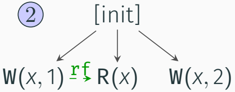
Fixing The Construction Order

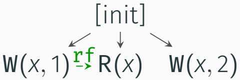
$[x = 0]$
$x := 1 \;\|\; a := x \;\|\; x := 2$

$$
\begin{array}{ccc}
 & [\text{init}] & \\
\swarrow & \downarrow & \searrow \\
\mathtt{W}(x,1) \xrightarrow{\mathtt{rf}} & \mathtt{R}(x) & \mathtt{W}(x,2)
\end{array}
$$

Fixing The Construction Order

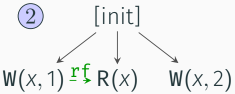
$[x = 0]$
$x := 1 \parallel a := x \parallel x := 2$

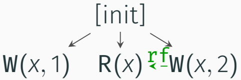




Fixing The Construction Order

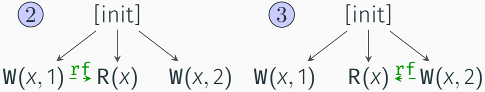
$[x = 0]$		
$x := 1$	$a := x$	$x := 2$

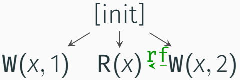




Fixing The Construction Order

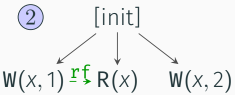
$[x = 0]$		
$x := 1$	$a := x$	$x := 2$

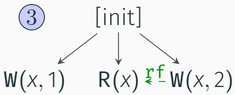


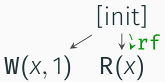


Fixing The Construction Order

$[x = 0]$		
$x := 1$	$a := x$	$x := 2$



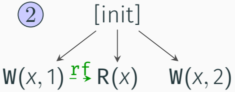


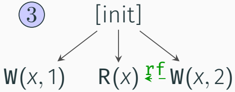


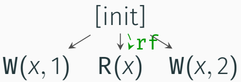
Fixing The Construction Order

$$[x = 0]$$

$$x := 1 \parallel a := x \parallel x := 2$$

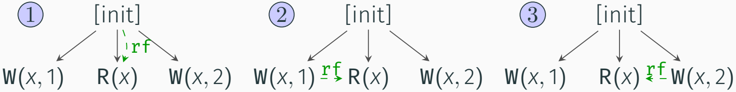


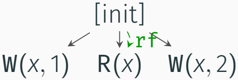




Fixing The Construction Order

$[x = 0]$		
$x := 1$	$a := x$	$x := 2$





Goal: Enumerate all consistent execution graphs of P for *any* memory model, where

- $\mathtt{po} \cup \mathtt{rf}$ is acyclic
- Consistency is prefix-closed

Goal: Enumerate all consistent execution graphs of *P* for *any* memory model, where

- $\mathtt{po} \cup \mathtt{rf}$ is acyclic
- Consistency is prefix-closed
- The memory model is extensible

Goal: Enumerate all consistent execution graphs of *P* for *any* memory model, where

- $\mathsf{po} \cup \mathtt{rf}$ is acyclic
- Consistency is prefix-closed
- The memory model is extensible

These are fulfilled by SC, TSO, PSO, RC11

GenMC: Generic Model Checking

Goal: Enumerate all consistent execution graphs of P for *any* memory model, where

- $\mathtt{po} \cup \mathtt{rf}$ is acyclic
- Consistency is prefix-closed
- The memory model is extensible

These are fulfilled by SC, TSO, PSO, RC11
but not by POWER and ARM

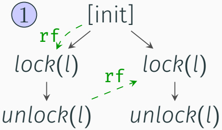
Handling Locks

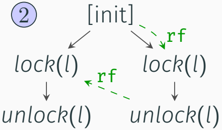
Handling Locks

$[init(l)]$	
$lock(l)$ $unlock(l)$	$lock(l)$ $unlock(l)$

Handling Locks

$[init(l)]$	
$lock(l)$	$lock(l)$
$unlock(l)$	$unlock(l)$





Results

An Interesting Example

	Nidhugg		RCMC	GenMC	
	SC	SC^{o}	$RC11^{\texttt{mo}}$	$RC11^{\texttt{mo}}$	$RC11^{\texttt{porf}}$
lamport(2)	.13	.10	.04	.03	.03
lamport(3)	7.53	4.49	5.40	6.87	1.36
lamport(4)	🕒	🕒	🕒	🕒	🕒

🕒 = tool did not finish within 2 days

All times are in seconds

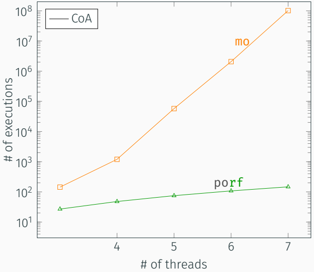
An Interesting Example

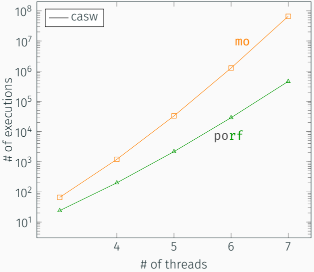
	Nidhugg		RCMC	GenMC		
	SC	SC^{o}	$RC11^{\mathtt{mo}}$	$RC11^{\mathtt{mo}}$	$RC11^{\mathtt{porf}}$	LIB
lamport(2)	.13	.10	.04	.03	.03	.09
lamport(3)	7.53	4.49	5.40	6.87	1.36	.09
lamport(4)	🕒	🕒	🕒	🕒	🕒	.09

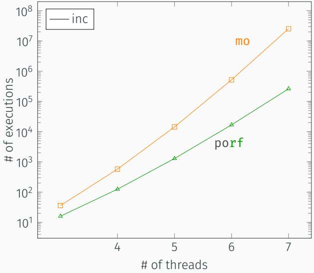
🕒 = tool did not finish within 2 days

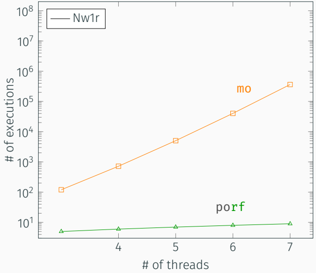
All times are in seconds

Equivalence Partitionings

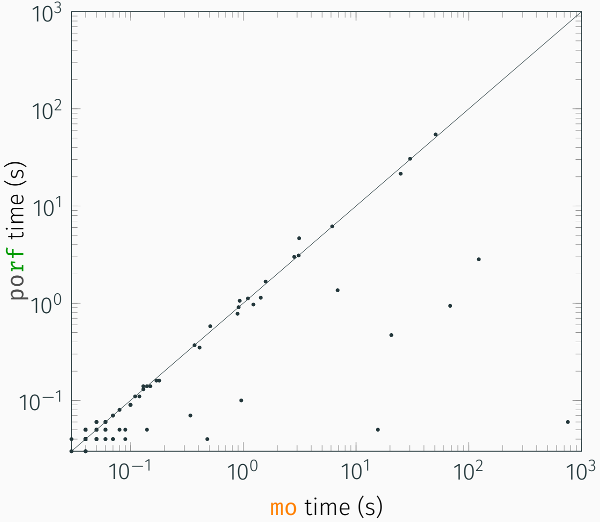








Equivalence Partitionings (202 benchmarks)



More in the paper

- Detailed description of the algorithm
- Formalization of memory model assumptions
- More benchmarks and evaluation

Conclusions

Summary

- Sound, complete, and optimal SMC procedure for memory models that are:
  - $\mathtt{po} \cup \mathtt{rf}$-acyclic
  - prefix-closed
  - extensible
- GenMC can be exponentially faster than existing tools
- GenMC is available at `github.com/MPI-SWS/genmc`

Conclusions

Summary

- Sound, complete, and optimal SMC procedure for memory models that are:
  - $\mathtt{po} \cup \mathtt{rf}$-acyclic
  - prefix-closed
  - extensible
- GenMC can be exponentially faster than existing tools
- GenMC is available at `github.com/MPI-SWS/genmc`

Future work

- Can we relax the memory-model assumptions?

Thank You!

Backup Slides

Why Extensibility Is Necessary

$[x = y = 0]$
$a := x \parallel b := y \parallel x := 42$

Why Extensibility Is Necessary

$[x = y = 0]$
$a := x \parallel b := y \parallel x := 42$

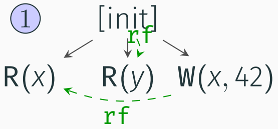
Under a memory model that dictates the following:

"If a read of y reads 0, then there cannot be a read of x that also reads 0"

Why Extensibility Is Necessary

$$[x = y = 0]$$

$$a := x \parallel b := y \parallel x := 42$$



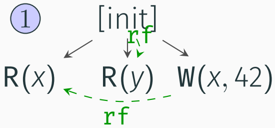
Under a memory model that dictates the following:

“If a read of y reads 0, then there cannot be a read of x that also reads 0”

Why Extensibility Is Necessary

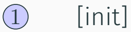
$$[x = y = 0]$$

$$a := x \parallel b := y \parallel x := 42$$



Under a memory model that dictates the following:

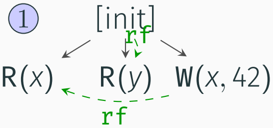
"If a read of y reads 0, then there cannot be a read of x that also reads 0"



Why Extensibility Is Necessary

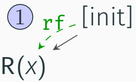
$$[x = y = 0]$$

$$a := x \parallel b := y \parallel x := 42$$



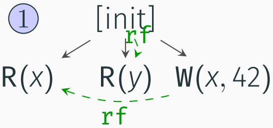
Under a memory model that dictates the following:

“If a read of y reads 0, then there cannot be a read of x that also reads 0”



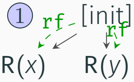
Why Extensibility Is Necessary

$[x = y = 0]$
$a := x \parallel b := y \parallel x := 42$



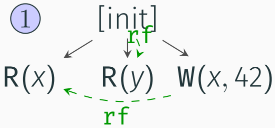
Under a memory model that dictates the following:

“If a read of y reads 0, then there cannot be a read of x that also reads 0”



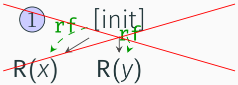
Why Extensibility Is Necessary

$[x = y = 0]$		
$a := x$	$b := y$	$x := 42$



Under a memory model that dictates the following:

"If a read of y reads 0, then there cannot be a read of x that also reads 0"



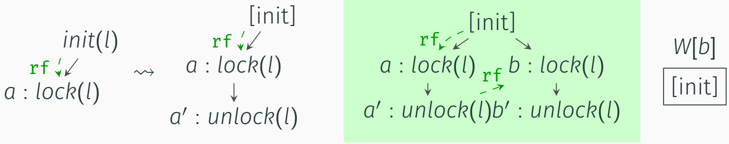
Handling Locks

[$init(l)$]	
$a : lock(l)$	$b : lock(l)$
$a' : unlock(l)$	$b' : unlock(l)$

$$
\begin{tikzcd}
init(l) \arrow[d, "\mathtt{rf}"', dashed] \arrow[d] \\
a : lock(l)
\end{tikzcd}
\rightsquigarrow
\begin{tikzcd}
[init] \arrow[d, "\mathtt{rf}"', dashed] \arrow[d] \\
a : lock(l) \arrow[d] \\
a' : unlock(l)
\end{tikzcd}
$$

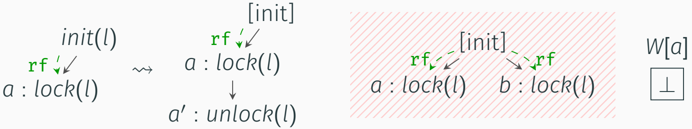
Handling Locks

$[init(l)]$	
$a : lock(l)$	$b : lock(l)$
$a' : unlock(l)$	$b' : unlock(l)$



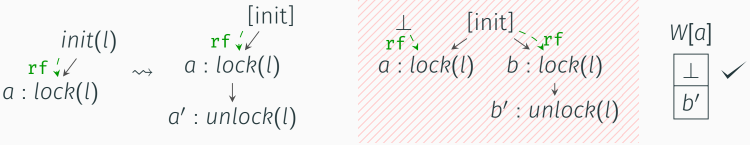
Handling Locks

$[init(l)]$	
$a : lock(l)$ $a' : unlock(l)$	$b : lock(l)$ $b' : unlock(l)$



Handling Locks

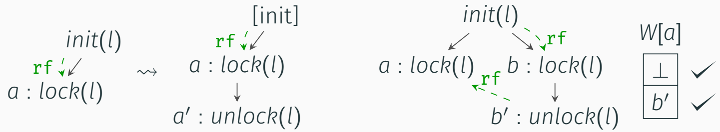
$[init(l)]$	
$a : lock(l)$	$b : lock(l)$
$a' : unlock(l)$	$b' : unlock(l)$



Handling Locks

$$[\mathit{init}(l)]$$

$a : \mathit{lock}(l)$	$\Vert$	$b : \mathit{lock}(l)$
$a' : \mathit{unlock}(l)$	$\Vert$	$b' : \mathit{unlock}(l)$



Linux-Kernel Benchmarks

	Nidhugg SC	Nidhugg TSO	Nidhugg PSO	RCMC RC11	RCMC WRC11	GENMC MO	GENMC WB
mcs_spinlock(2)	.12	.09	.10	.05	.05	.05	.05
mcs_spinlock(3)	2.98	6.84	12.54	.84	.67	.89	.78
mcs_spinlock(4)	0.68h	1.51h	3.32h	0.16h	0.15h	0.42h	0.26h
qspinlock(2)	.17	.11	.11	.04	.04	.04	.04
qspinlock(3)	10.93	18.20	23.43	2.13	2.08	1.10	1.12
seqlock(2)	.10	.09	.10	.04	.04	.04	.04
seqlock(3)	1.64	3.07	11.00	.49	.51	.37	.37