

# Pomsets with Preconditions

## A Simple Model of Relaxed Memory

Radha Jagadeesan<sup>†</sup>    Alan Jeffrey<sup>\*</sup>    James Riely<sup>†</sup>

<sup>†</sup>DePaul University

<sup>\*</sup>Mozilla Research and the Servo Project

OOPSLA  
November 2020

# Memory model

What value can be read for  $x$ ?

$x := 1; x := 2$

$\parallel x := 3; x := 4;$

$s := x; x := 5$



- $x-z$ : shared locations, initially 0
- $r-s$ : registers
- $po$ : program order
- Concurrent: 1? 2?
- Sequential: 3? 4? 5?

# Memory model

What value can be read for  $x$ ?

$x := 1; x := 2$

$\parallel x := 3; x := 4;$

$s := x; x := 5$

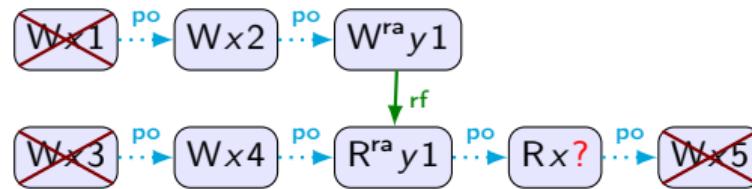


- $x-z$ : shared locations, initially 0
- $r-s$ : registers
- $po$ : program order
- Concurrent: 1✓ 2✓
- Sequential: 3✗ 4✓ 5✗

# Memory model

What value can be read for  $x$ ?

$x := 1; x := 2; y^{\text{ra}} := 1 \parallel x := 3; x := 4; r := y^{\text{ra}}; s := x; x := 5$

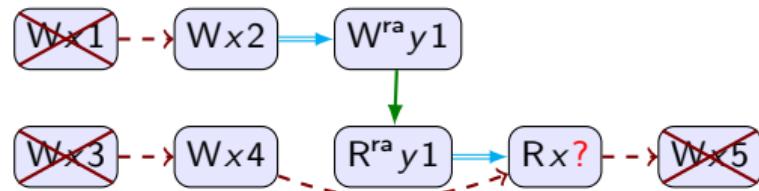


- $x-z$ : shared locations, initially 0
  - $r-s$ : registers
  - $\text{po}$ : program order
  - $\text{rf}$ : reads-from
  - $W^{\text{ra}}$ : release
  - $R^{\text{ra}}$ : acquire
- Concurrent: 1 $\times$  2✓
  - Sequential: 3 $\times$  4✓ 5 $\times$

# Memory model

What value can be read for  $x$ ?

$x := 1; x := 2; y^{\text{ra}} := 1 \parallel x := 3; x := 4; r := y^{\text{ra}}; s := x; x := 5$

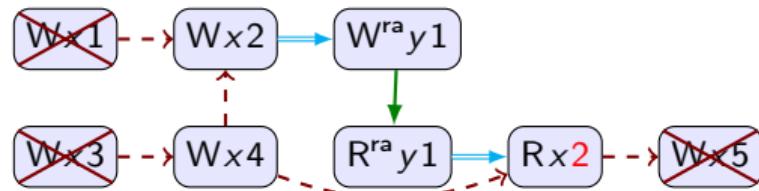


- Labelled partial order (Pomset)
  - ->  $\text{po} \cap x$
  - =>  $\text{po}$  in to release
  - =>  $\text{po}$  out of acquire
- Fulfillment
  - $(Wxv)$  before  $(R xv)$
  - -> Any other  $(Wx)$  before  $(Wxv)$  or after  $(R xv)$

# Memory model

What value can be read for  $x$ ?

$x := 1; x := 2; y^{\text{ra}} := 1 \parallel x := 3; x := 4; r := y^{\text{ra}}; s := x; x := 5$

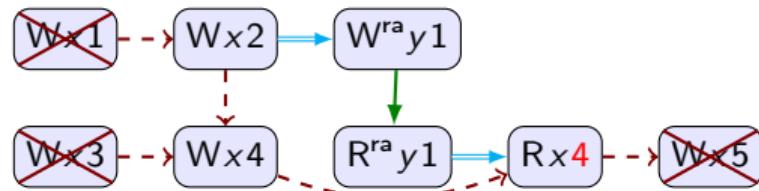


- Labelled partial order (Pomset)
  - ->  $\text{po} \cap x$
  - ==>  $\text{po}$  in to release
  - ==>  $\text{po}$  out of acquire
- Fulfillment
  - $(Wxv)$  before  $(Rxv)$
  - -> Any other  $(Wx)$  before  $(Wxv)$  or after  $(Rxv)$

# Memory model

What value can be read for  $x$ ?

$x := 1; x := 2; y^{\text{ra}} := 1 \parallel x := 3; x := 4; r := y^{\text{ra}}; s := x; x := 5$

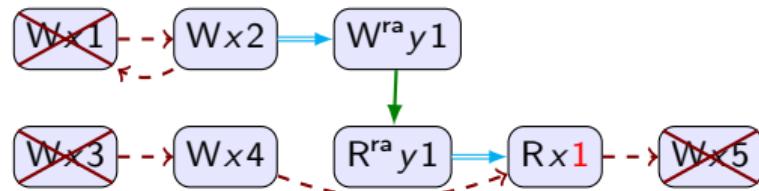


- Labelled partial order (Pomset)
  - ->  $\text{po} \cap x$
  - ==>  $\text{po}$  in to release
  - ==>  $\text{po}$  out of acquire
- Fulfillment
  - $(Wxv)$  before  $(Rxv)$
  - -> Any other  $(Wx)$  before  $(Wxv)$  or after  $(Rxv)$

# Memory model

What value can be read for  $x$ ?

$x := 1; x := 2; y^{\text{ra}} := 1 \parallel x := 3; x := 4; r := y^{\text{ra}}; s := x; x := 5$

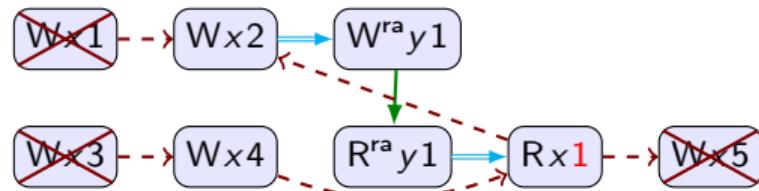


- Labelled partial order (Pomset)
  - ->  $\text{po} \cap x$
  - =>  $\text{po}$  in to release
  - =>  $\text{po}$  out of acquire
- Fulfillment
  - $(Wxv)$  before  $(Rxv)$
  - -> Any other  $(Wx)$  before  $(Wxv)$  or after  $(Rxv)$

# Memory model

What value can be read for  $x$ ?

$x := 1; x := 2; y^{\text{ra}} := 1 \parallel x := 3; x := 4; r := y^{\text{ra}}; s := x; x := 5$



- Labelled partial order (Pomset)
  - -> po  $\cap x$
  - ==> po in to release
  - ==> po out of acquire
- Fulfillment
  - (Wxv) before (Rxv)
  - -> Any other (Wx) before (Wxv) or after (Rxv)

A vibrant field of sunflowers bathed in golden sunlight. The flowers are in various stages of bloom, from tight buds to fully open blossoms with dark brown centers. The leaves are large and green, some with visible veins. In the foreground, the word "PRETTY" is printed in large, bold, white capital letters. The letters have a thin black outline and a warm, glowing yellow-to-orange gradient fill, matching the color of the sunflowers.

PRETTY

# Preconditions for dependency

Pomset = a single execution

$$s := x; y := s \parallel r := y; x := 1; z := r$$

Rx1

s=1 | Wy1

Ry1

true | Wx1

r=1 | Wz1

- Writes have preconditions

# Preconditions for dependency

Pomset = a single execution

$$s := x; y := s \parallel r := y; x := 1; z := r$$



- Writes have preconditions
- Order from  $s := (Rx1)$  substitutes [1/s]

# Preconditions for dependency

Pomset = a single execution

$$s := x; y := s \parallel r := y; x := 1; z := r$$



- Writes have preconditions
- Order from  $s := (Rx1)$  substitutes  $[1/s]$
- Order from  $r := (Ry1)$  substitutes  $[1/r]$

# Preconditions for dependency

Pomset = a single execution

$$y := x \parallel r := y; x := 1; z := r$$



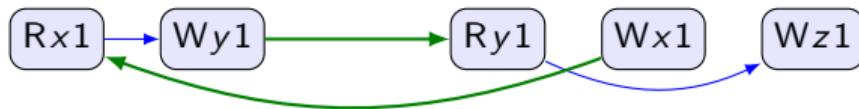
- Writes have preconditions
- Order from  $s := (Rx1)$  substitutes  $[1/s]$
- Order from  $r := (Ry1)$  substitutes  $[1/r]$
- We elide tautologies

# Out of order

What value can be written to  $z$ ?

$y := x \parallel r := y; x := 1; z := r$

(\*) ✓



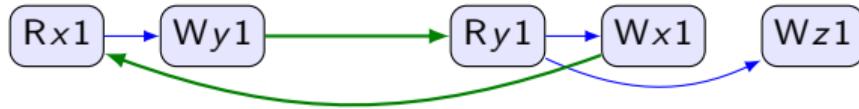
- Partial order
  - -> per-location
  - synchronization
  - reads-from
  - dependency
- May reorder  $r := y$  and  $x := 1$ 
  - in compiler
  - on ARM

# Out of order thin air (OOTA)

What value can be written to  $z$ ?

$y := x \parallel r := y; x := \textcolor{red}{r}; z := r$

(OOTA1) X



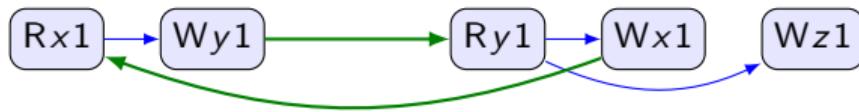
- Partial order
  - per-location
  - synchronization
  - reads-from
  - dependency
- ~~May reorder  $r := y$  and  $x := 1$~~ 
  - ~~in compiler~~
  - ~~on ARM~~

# Out of order thin air (OOTA)

What value can be written to  $z$ ?

$y := x \parallel r := y; x := \textcolor{red}{r}; z := r$

(OOTA1) X



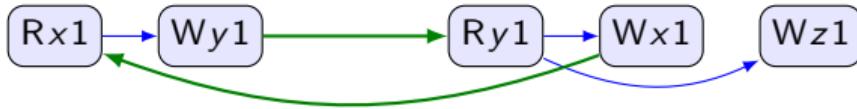
- Partial order
  - -> per-location
  - synchronization
  - reads-from
  - dependency
- ~~May reorder  $r := y$  and  $x := 1$~~ 
  - ~~in compiler~~
  - ~~on ARM~~
- Partial order prevents!

# Out of order thin air (OOTA)

What value can be written to  $z$ ?

$\text{if}(x)\{y:=1\} \parallel r:=y; \text{if}(r)\{x:=1; z:=r\}$

(OOTA2) X



- Partial order
  - -> per-location
  - synchronization
  - reads-from
  - dependency
- ~~May reorder  $r := y$  and  $x := 1$~~ 
  - ~~in compiler~~
  - ~~on ARM~~
- Partial order prevents!
- Control flow variant is DRF

A photograph of a sunflower field at sunset. The sunflowers are tall with large, bright yellow flowers and green leaves. The background is filled with the golden glow of many sunflowers. Overlaid on the image is the text "VERY PRETTY" in large, white, sans-serif capital letters. The letters are semi-transparent, allowing the sunflowers to be seen through them.

VERY  
PRETTY

Can this program write 1 to z?

$y := x \parallel r := y;$        $x := r; z := r$       (OOTA1)

Can this program write 1 to z?

$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 1\}$  (\*)

Can this program write 1 to z?

$$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 1\} \quad (*)$$

x and y can only be 0 or 1

$$y := x \parallel r := y; \text{if}(r)\{x := 1; z := 1\} \text{else}\{x := 1\}$$

Can this program write 1 to z?

$$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 1\} \quad (\star)$$

x and y can only be 0 or 1

$$y := x \parallel r := y; \text{if}(r)\{\textcolor{red}{x := 1}; z := 1\} \text{else}\{\textcolor{red}{x := 1}\}$$

lift common code

$$y := x \parallel r := y; \textcolor{red}{x := 1}; \text{if}(r)\{z := 1\}$$

Can this program write 1 to z?

$$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 1\} \quad (\star)$$

x and y can only be 0 or 1

$$y := x \parallel r := y; \text{if}(r)\{x := 1; z := 1\} \text{else}\{x := 1\}$$

lift common code

$$y := x \parallel \color{red}{r := y; x := 1}; \text{if}(r)\{z := 1\}$$

commute independent statements

$$y := x \parallel \color{red}{x := 1; r := y}; \text{if}(r)\{z := 1\}$$

Can this program write 1 to z?

$$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 1\} \quad (*)$$

x and y can only be 0 or 1

$$y := x \parallel r := y; \text{if}(r)\{x := 1; z := 1\} \text{else}\{x := 1\}$$

lift common code

$$y := x \parallel r := y; x := 1; \text{if}(r)\{z := 1\}$$

commute independent statements

$$\color{red}{y := x} \parallel x := 1; r := y; \text{if}(r)\{z := 1\}$$

interleaving

$$x := 1; \color{red}{y := x}; r := y; \text{if}(r)\{z := 1\}$$

Can this program write 1 to z?

$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 2\}$

(OOTA3)

$x$  and  $y$  can only be 0 or 2

$y := x \parallel r := y; \text{if}(r)\{x := 2; z := 2\} \text{else}\{x := 2\}$

lift common code

$y := x \parallel r := y; x := 2; \text{if}(r)\{z := 2\}$

commute independent statements

$y := x \parallel x := 2; r := y; \text{if}(r)\{z := 2\}$

interleaving

$x := 2; y := x; r := y; \text{if}(r)\{z := 2\}$

Can this program write 1 to z?

$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := \textcolor{red}{2}\}$

(OOTA3)

~~x and y can only be 0 or 1~~

~~$y := x \parallel r := y; \text{if}(r)\{x := r; z := 1\} \text{else}\{x := 1\}$~~

~~lift common code~~

~~$y := x \parallel r := y; x := 1; \text{if}(r)\{z := 1\}$~~

~~commute independent statements~~

~~$y := x \parallel x := 1; r := y; \text{if}(r)\{z := 1\}$~~

~~interleaving~~

~~$x := 1; y := x; r := y; \text{if}(r)\{z := 1\}$~~

Can this program write 1 to z?

$y := x \parallel r := y; \text{if}(b)\{x := r; z := r\} \text{else}\{x := 1\} \parallel b := 1$  (OOTA4)

~~x and y can only be 0 or 1~~

~~$y := x \parallel r := y; \text{if}(r)\{x := r; z := 1\} \text{else}\{x := 1\}$~~

~~lift common code~~

~~$y := x \parallel r := y; x := 1; \text{if}(r)\{z := 1\}$~~

~~commute independent statements~~

~~$y := x \parallel x := 1; r := y; \text{if}(r)\{z := 1\}$~~

~~interleaving~~

~~$x := 1; y := x; r := y; \text{if}(r)\{z := 1\}$~~

Can this program write 1 to z?

$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 1\}$  (\*)

$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 2\}$  (OOTA3)

$y := x \parallel r := y; \text{if}(\textcolor{red}{b})\{x := r; z := r\} \text{else}\{x := 1\} \parallel b := 1$  (OOTA4)

# Can this program write 1 to z?

$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 1\}$  (\*)

$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 2\}$  (OOTA3)

$y := x \parallel r := y; \text{if}(\textcolor{red}{b})\{x := r; z := r\} \text{else}\{x := 1\} \parallel b := 1$  (OOTA4)

- Existential justification: ARM ✓ (\*) ✓ OOTA1-3 ✓ OOTA4 ✗
  - Commitment (JMM - Java Memory Model) [Manson, Pugh, Adve, 2005]
  - Speculation [Jagadeesan, Pitcher, Riely, 2010]
  - Promising [Kang, Hur, Lahav, Vafeiadis, Dreyer 2017]

# Can this program write 1 to z?

$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 1\}$  (\*)

$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 2\}$  (OOTA3)

$y := x \parallel r := y; \text{if}(b)\{x := r; z := r\} \text{else}\{x := 1\} \parallel b := 1$  (OOTA4)

- Existential justification: ARM ✓ (\*) ✓ OOTA1-3 ✓ OOTA4 ✗
  - Commitment (JMM - Java Memory Model) [Manson, Pugh, Adve, 2005]
  - Speculation [Jagadeesan, Pitcher, Riely, 2010]
  - Promising [Kang, Hur, Lahav, Vafeiadis, Dreyer 2017]
- Universal justification: ARM ✗ (\*) ✓ OOTA1-3 ✓ OOTA4 ✓
  - Event Structures [Jeffrey, Riely, 2016]

# Can this program write 1 to z?

$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 1\}$  (\*)

$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 2\}$  (OOTA3)

$y := x \parallel r := y; \text{if}(b)\{x := r; z := r\} \text{else}\{x := 1\} \parallel b := 1$  (OOTA4)

$y := x \parallel r := y; \text{if}(b)\{r := \text{new D}\} \text{else}\{s := \text{new C}\}; x := r \parallel b := 1$  (OOTA5)

- In the JMM, OOTA5 “*is type correct if it declares x, y and r of type D. However, it has a legal execution where they reference a C object.*” [Lochbihler 2013]

# Can this program write 1 to z?

$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 1\}$  (\*)

$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 2\}$  (OOTA3)

$y := x \parallel r := y; \text{if}(b)\{x := r; z := r\} \text{else}\{x := 1\} \parallel b := 1$  (OOTA4)

$y := x \parallel r := y; \text{if}(b)\{r := \text{new D}\} \text{else}\{s := \text{new C}\}; x := r \parallel b := 1$  (OOTA5)

- In the JMM, OOTA5 “*is type correct if it declares x, y and r of type D. However, it has a legal execution where they reference a C object.*” [Lochbihler 2013]
  - To prove safety, Lochbihler partitions memory by type

# Can this program write 1 to z?

$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 1\}$  (\*)

$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 2\}$  (OOTA3)

$y := x \parallel r := y; \text{if}(b)\{x := r; z := r\} \text{else}\{x := 1\} \parallel b := 1$  (OOTA4)

$y := x \parallel r := y; \text{if}(b)\{r := \text{new D}\} \text{else}\{s := \text{new C}\}; x := r \parallel b := 1$  (OOTA5)

- In the JMM, OOTA5 “*is type correct if it declares x, y and r of type D. However, it has a legal execution where they reference a C object.*” [Lochbihler 2013]
  - To prove safety, Lochbihler partitions memory by type
    - ~~Realistic memory allocation~~
    - ~~Java Security Architecture (security sensitive constants)~~

# Can this program write 1 to z?

$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 1\}$  (\*)

$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 2\}$  (OOTA3)

$y := x \parallel r := y; \text{if}(b)\{x := r; z := r\} \text{else}\{x := 1\} \parallel b := 1$  (OOTA4)

$y := x \parallel r := y; \text{if}(b)\{r := \text{new D}\} \text{else}\{s := \text{new C}\}; x := r \parallel b := 1$  (OOTA5)

- In the JMM, OOTA5 “*is type correct if it declares x, y and r of type D. However, it has a legal execution where they reference a C object.*” [Lochbihler 2013]
  - To prove safety, Lochbihler partitions memory by type
    - ~~Realistic memory allocation~~
    - ~~Java Security Architecture (security sensitive constants)~~
- *Out Of Thin Air (OOTA) is a queasy feeling*

# Can this program write 1 to z?

$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 1\}$  (\*)

$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 2\}$  (OOTA3)

$y := x \parallel r := y; \text{if}(b)\{x := r; z := r\} \text{else}\{x := 1\} \parallel b := 1$  (OOTA4)

$y := x \parallel r := y; \text{if}(b)\{r := \text{new D}\} \text{else}\{s := \text{new C}\}; x := r \parallel b := 1$  (OOTA5)

- In the JMM, OOTA5 “*is type correct if it declares x, y and r of type D. However, it has a legal execution where they reference a C object.*” [Lochbihler 2013]
  - To prove safety, Lochbihler partitions memory by type
    - ~~Realistic memory allocation~~
    - ~~Java Security Architecture (security sensitive constants)~~
- *Out Of Thin Air (OOTA) is a queasy feeling  
Compositionality of proof rules is a verifiable property*

# Can this program write 1 to z?

$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 1\}$  (\*)

$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 2\}$  (OOTA3)

$y := x \parallel r := y; \text{if}(b)\{x := r; z := r\} \text{else}\{x := 1\} \parallel b := 1$  (OOTA4)

$y := x \parallel r := y; \text{if}(b)\{r := \text{newD}\} \text{else}\{s := \text{newC}\}; x := r \parallel b := 1$  (OOTA5)

- In OOTA4, every thread satisfies:
  - $Wy_1$  must be preceded by  $Rx_1$
  - if  $Wz_1$ , then  $Wx_1$  must be preceded by  $Ry_1$
  - In PLTL:  $[\Diamond Wy_1 \Rightarrow \Diamond Rx_1] \wedge [Wz_1 \Rightarrow (\Diamond Ry_1 \wedge \Box(Wx_1 \Rightarrow \Diamond Ry_1))]$

# Can this program write 1 to z?

$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 1\}$  (\*)

$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else}\{x := 2\}$  (OOTA3)

$y := x \parallel r := y; \text{if}(b)\{x := r; z := r\} \text{else}\{x := 1\} \parallel b := 1$  (OOTA4)

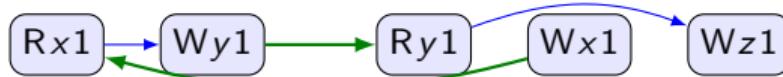
$y := x \parallel r := y; \text{if}(b)\{r := \text{newD}\} \text{else}\{s := \text{newC}\}; x := r \parallel b := 1$  (OOTA5)

- In OOTA4, every thread satisfies:
  - $Wy_1$  must be preceded by  $Rx_1$
  - if  $Wz_1$ , then  $Wx_1$  must be preceded by  $Ry_1$
  - In PLTL:  $[\Diamond Wy_1 \Rightarrow \Diamond Rx_1] \wedge [Wz_1 \Rightarrow (\Diamond Ry_1 \wedge \Box(Wx_1 \Rightarrow \Diamond Ry_1))]$
- Compositionality: every thread satisfies  $\implies$  program satisfies
  - $\exists$  justification: compositional for predicate logic OOTA1-3
  - $\forall$  justification: compositional for temporal logic OOTA4-5

## Comparing attempted executions

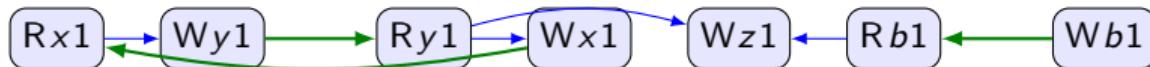
$y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else } \{x := 1\}$

(\*) ✓



$y := x \parallel r := y; \text{if}(b)\{x := r; z := r\} \text{else } \{x := 1\} \parallel b := 1$

(OOTA4) ✗



A photograph of a sunflower field with large, semi-transparent white letters overlaid. The letters spell out "STOLE PRETTY?" in a bold, sans-serif font. The background consists of numerous sunflowers with their characteristic yellow petals and dark centers, set against a backdrop of green foliage and a bright, slightly hazy sky.

STOLE  
PRETTY?

It's hard...

---

- Programmer desiderata
  - Compositional/local reasoning
  - Sequential Consistency for Data Race Free programs (SC-DRF)

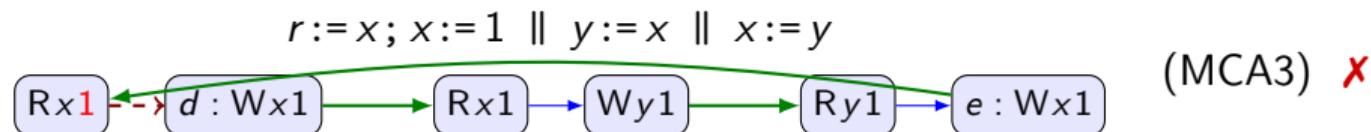
It's hard...

---

- Programmer desiderata
  - Compositional/local reasoning
  - Sequential Consistency for Data Race Free programs (SC-DRF)
- Implementer desiderata
  - Efficient on hardware
  - Compiler optimizations

It's hard...

- Programmer desiderata
  - Compositional/local reasoning
  - Sequential Consistency for Data Race Free programs (SC-DRF)
- Implementer desiderata
  - Efficient on hardware MCA hardware
  - Compiler optimizations
- Multi-copy atomicity (MCA)
  - When write published to one processor, published to all
  - ARM✓ x86-64✓ RISC-V✓ POWER✗
  - Prevents anomalies:



We've been trying for 20 years...

- Operational with alternate executions: ARM✓ OOTA✗
  - Java 1.1 fails CSE [Pugh 1999]
  - Commitment/Speculation/Promises [2005, 2010, 2017]
- Strong models: ARM✗ OOTA✓
  - SC [Singh, Narayanasamy, Marino, Millstein, Musuvathi 2012]
  - RC11 [Lahav, Vafeiadis 2016]
  - Event Structures [2016]

We've been trying for 20 years...

- Operational with alternate executions: ARM✓ OOTA✗
  - Java 1.1 fails CSE [Pugh 1999]
  - Commitment/Speculation/Promises [2005, 2010, 2017]
- Strong models: ARM✗ OOTA✓
  - SC [Singh, Narayanasamy, Marino, Millstein, Musuvathi 2012]
  - RC11 [Lahav, Vafeiadis 2016]
  - Event Structures [2016]
- Operational with compiler rewrites
  - [Ferreira, Feng, Shao 2010], [Pichon-Pharabod, Sewell 2016]
- Symbolic execution with multiple orders and acyclicly requirements
  - Hardware [Alglave 2010], [Alglave, Maranget, Tautschnig 2014]
  - C++ [Batty, Owens, Sarkar, Sewell, Weber 2011]
- Event Structures
  - WeakestMO [Chakraborty, Vafeiadis 2019]
  - Modularity [Paviotti, Cooksey, Paradis, Wright, Owens, Batty 2020]

# PRINCIPLES

A vibrant field of sunflowers bathed in golden sunlight. The flowers are in various stages of bloom, from tight buds to fully open blossoms with dark brown centers. The leaves are large and green, some with visible veins. In the foreground, the word "PRINCIPLES" is printed in large, bold, white, sans-serif capital letters. The letters have a slight shadow or glow effect, making them stand out against the bright background. The overall composition is a mix of nature's organic beauty and a clean, modern graphic design.

Image in the Public domain

# Compositionality

A photograph of a sunflower field at sunset. The sunflowers are bright yellow with dark centers. The background is filled with more sunflowers and some green foliage. In the foreground, there is a large, out-of-focus sunflower head on the left and a sharp sunflower head on the right. Overlaid on the image is the word "Compositionality" in a large, bold, white sans-serif font.

# Compositionality + Construction

A photograph of a sunflower field at sunset. The sunflowers are tall with large, bright yellow flowers and green leaves. The background is filled with more sunflowers, slightly out of focus. In the foreground, a large, out-of-focus sunflower head is visible on the left. Overlaid on the image is the title "Compositionality + Construction" in a large, bold, white sans-serif font. The text is positioned in the upper half of the image, with "Compositionality" on top and "+ Construction" below it.



**Compositionality**  
+ Construction  
+ Reasoning  
+ Local races  
+ Safety properties

A photograph of a sunflower field in full bloom. The flowers are bright yellow with dark brown centers. Large, bold, white letters spell out "LOGIC" across the center of the image. The letter "O" has a gold-colored circular glow behind it, and the letter "G" has a similar glow. The background is filled with more sunflowers and green foliage.

LOGIC



**LOGIC**  
+ PRECONDITIONS

The background of the image is a vibrant field of sunflowers at sunset, with their bright yellow petals and green leaves filling the frame. A large, semi-transparent white watermark is overlaid across the center. The word "LOGIC" is written in a bold, sans-serif font, with each letter having a thin black outline. The "O" is a circle containing a smaller sunflower head, and the "I" is a vertical rectangle with a diagonal line through it.

LOGIC

+ PRECONDITIONS  
+ TEMPORAL SAFETY



ONE  
ORDER

A photograph of a sunflower field with large, bold, white text overlaid. The text reads "ONE" on the first line and "ORDER" on the second line. The letters are slightly staggered. The background consists of numerous sunflowers with their characteristic yellow petals and brown centers, set against a backdrop of green foliage and a clear blue sky. The lighting suggests a bright, sunny day.

A vibrant field of sunflowers in full bloom, their bright yellow petals and dark brown centers contrasting against a backdrop of green leaves and stems. A large, semi-transparent white text overlay reads "4 SLIDES". The number "4" is positioned on the left, and "SLIDES" is on the right, both in a bold, sans-serif font. The letters have a thin gold outline, and the interior of the letters is white.

4 SLIDES

## Semantic Domain

- A *pomset with preconditions* is a tuple  $(E, \leq, \lambda)$  where
  - $E$  is a set of events
  - $\leq \subseteq (E \times E)$  is a partial order
  - $\lambda : E \rightarrow (\Phi \times \mathcal{A})$  is a *labeling* from which we derive functions
    - $\Phi : E \rightarrow \Phi$  (*formulae*)
    - $\mathcal{A} : E \rightarrow \mathcal{A}$  (*actions*)

## Semantic Domain

- A *pomset with preconditions* is a tuple  $(E, \leq, \lambda)$  where
  - $E$  is a set of events
  - $\leq \subseteq (E \times E)$  is a partial order
  - $\lambda : E \rightarrow (\Phi \times \mathcal{A})$  is a *labeling* from which we derive functions
    - $\Phi : E \rightarrow \Phi$  (*formulae*)
    - $\mathcal{A} : E \rightarrow \mathcal{A}$  (*actions*)
  - $\bigwedge_e \Phi(e)$  is satisfiable (*consistency*)
  - if  $d \leq e$  then  $\Phi(e)$  implies  $\Phi(d)$  (*causal strengthening*)

# Semantic Domain

- A *pomset with preconditions* is a tuple  $(E, \leq, \lambda)$  where
  - $E$  is a set of events
  - $\leq \subseteq (E \times E)$  is a partial order
  - $\lambda : E \rightarrow (\Phi \times \mathcal{A})$  is a *labeling* from which we derive functions
    - $\Phi : E \rightarrow \Phi$  (*formulae*)
    - $\mathcal{A} : E \rightarrow \mathcal{A}$  (*actions*)
  - $\bigwedge_e \Phi(e)$  is satisfiable (*consistency*)
  - if  $d \leq e$  then  $\Phi(e)$  implies  $\Phi(d)$  (*causal strengthening*)
- We say  $\mathcal{A}(d) = (Wxv)$  fulfills  $\mathcal{A}(e) = (Rxv)$  if
  - $d < e$
  - if  $\mathcal{A}(c) = (Wx..)$  then either  $c \leq d$  or  $e \leq c$

# Semantic Domain

- A *pomset with preconditions* is a tuple  $(E, \leq, \lambda)$  where
  - $E$  is a set of events
  - $\leq \subseteq (E \times E)$  is a partial order
  - $\lambda : E \rightarrow (\Phi \times \mathcal{A})$  is a *labeling* from which we derive functions
    - $\Phi : E \rightarrow \Phi$  (*formulae*)
    - $\mathcal{A} : E \rightarrow \mathcal{A}$  (*actions*)
  - $\bigwedge_e \Phi(e)$  is satisfiable (*consistency*)
  - if  $d \leq e$  then  $\Phi(e)$  implies  $\Phi(d)$  (*causal strengthening*)
- We say  $\mathcal{A}(d) = (Wxv)$  *fulfills*  $\mathcal{A}(e) = (Rxv)$  if
  - $d < e$
  - if  $\mathcal{A}(c) = (Wx..)$  then either  $c \leq d$  or  $e \leq c$
- A pomset is *x-closed* if
  - every  $\mathcal{A}(e) = (Rx..)$  is fulfilled
  - every  $\Phi(e)$  is independent of  $x$ :  $(\forall v. \Phi(e) \models \Phi(e)[v/x] \models \Phi(e))$

## Semantic Operations (1/2)

- Let  $P \in (\nu x. \mathcal{P})$  when  $P \in \mathcal{P}$  and  $P$  is  $x$ -closed

## Semantic Operations (1/2)

- Let  $P \in (\nu x. \mathcal{P})$  when  $P \in \mathcal{P}$  and  $P$  is  $x$ -closed
- Let  $P \in (\phi \triangleright \mathcal{P})$  when  $P \in \mathcal{P}$  and  $(\forall e \in E) \Phi(e)$  implies  $\phi$

## Semantic Operations (1/2)

- Let  $P \in (\nu x. \mathcal{P})$  when  $P \in \mathcal{P}$  and  $P$  is  $x$ -closed
- Let  $P \in (\phi \triangleright \mathcal{P})$  when  $P \in \mathcal{P}$  and  $(\forall e \in E) \Phi(e)$  implies  $\phi$
- Let  $P' \in (\mathcal{P}[M/x])$  when  $(\exists P \in \mathcal{P})$   
 $E' = E, \leq' = \leq, \mathcal{A}' = \mathcal{A}$ , and  $(\forall e \in E') \Phi'(e) = \Phi(e)[M/x]$

## Semantic Operations (1/2)

- Let  $P \in (\nu x. \mathcal{P})$  when  $P \in \mathcal{P}$  and  $P$  is  $x$ -closed
- Let  $P \in (\phi \triangleright \mathcal{P})$  when  $P \in \mathcal{P}$  and  $(\forall e \in E) \Phi(e)$  implies  $\phi$
- Let  $P' \in (\mathcal{P}[M/x])$  when  $(\exists P \in \mathcal{P})$   
 $E' = E$ ,  $\leq' = \leq$ ,  $\mathcal{A}' = \mathcal{A}$ , and  $(\forall e \in E') \Phi'(e) = \Phi(e)[M/x]$
- Let  $P' \in (\mathcal{P}^1 \parallel \mathcal{P}^2)$  when  $(\exists P^1 \in \mathcal{P}^1) (\exists P^2 \in \mathcal{P}^2)$   
 $E' = E^1 \cup E^2$ ,  $\leq' \supseteq \leq^1 \cup \leq^2$ , and  $(\forall e \in E')$  either

$e \notin E^2$ ,  $\mathcal{A}'(e) = \mathcal{A}^1(e)$  and  $\Phi'(e)$  implies  $\Phi^1(e)$ ,  
 $e \notin E^1$ ,  $\mathcal{A}'(e) = \mathcal{A}^2(e)$  and  $\Phi'(e)$  implies  $\Phi^2(e)$ , or  
 $\mathcal{A}'(e) = \mathcal{A}^1(e) = \mathcal{A}^2(e)$  and  $\Phi'(e)$  implies  $\Phi^1(e) \vee \Phi^2(e)$

## Semantic Operations (2/2)

- Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$ 
  - (p1)  $E' = E \cup \{d\}$
  - (p2)  $\leq' \supseteq \leq$
  - (p3a)  $\mathcal{A}'(e) = \mathcal{A}(e)$
  - (p3b)  $\mathcal{A}'(d) = a$
  - (p4a)  $\Phi'(d)$  implies  $\phi \wedge (d \notin E \vee \Phi(d))$
  - (p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$
  - (p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$
  - (p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$
  - (p5b) if  $d$  and  $e$  are conflicting actions then  $d \leq' e$
  - (p5c) if  $d$  is an acquire or  $e$  is a release then  $d \leq' e$
  - (p5d) if  $d$  is an SC write and  $e$  is an SC read then  $d \leq' e$
  - (p5e) if  $d$  reads, and  $e$  is an acquiring fence, then  $d \leq' e$
  - (p5f) if  $d$  is a releasing fence, and  $e$  writes, then  $d \leq' e$

## Semantic Operations (2/2)

- Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$ 
  - (p1)  $E' = E \cup \{d\}$
  - (p2)  $\leq' \supseteq \leq$
  - (p3a)  $\mathcal{A}'(e) = \mathcal{A}(e)$
  - (p3b)  $\mathcal{A}'(d) = a$
  - (p4a)  $\Phi'(d)$  implies  $\phi \wedge (d \notin E \vee \Phi(d))$
  - (p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$
  - (p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$
  - (p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$
  - (p5b) if  $d$  and  $e$  are conflicting actions then  $d \leq' e$
  - (p5c) if  $d$  is an acquire or  $e$  is a release then  $d \leq' e$
  - (p5d) if  $d$  is an SC write and  $e$  is an SC read then  $d \leq' e$
  - (p5e) if  $d$  reads, and  $e$  is an acquiring fence, then  $d \leq' e$
  - (p5f) if  $d$  is a releasing fence, and  $e$  writes, then  $d \leq' e$

## Language (See paper for RMWs and address calculation)

$$\begin{aligned}\llbracket \text{skip} \rrbracket &\triangleq \{\checkmark\} \\ \llbracket r := M; C \rrbracket &\triangleq \llbracket C \rrbracket[M/r] \\ \llbracket r := x^{\mu}; C \rrbracket &\triangleq \bigcup_v (R^{\mu} x v) \Rightarrow \llbracket C \rrbracket[x/r] \\ \llbracket x^{\mu} := M; C \rrbracket &\triangleq \bigcup_v (M = v \mid W^{\mu} x v) \Rightarrow \llbracket C \rrbracket[M/x] \\ \llbracket F^{\kappa}; C \rrbracket &\triangleq (F^{\kappa}) \Rightarrow \llbracket C \rrbracket \\ \llbracket \text{if}(M)\{C\} \text{ else }\{D\} \rrbracket &\triangleq (M \triangleright \llbracket C \rrbracket) \parallel (\neg M \triangleright \llbracket D \rrbracket) \\ \llbracket C \parallel D \rrbracket &\triangleq \llbracket C \rrbracket \parallel \llbracket D \rrbracket \\ \llbracket \text{var } x; C \rrbracket &\triangleq \nu x . \llbracket C \rrbracket\end{aligned}$$

|       |              |              |                           |          |              |              |           |
|-------|--------------|--------------|---------------------------|----------|--------------|--------------|-----------|
| $\mu$ | $\text{::=}$ | $\text{rlx}$ | (Relaxed)                 | $\kappa$ | $\text{::=}$ | $\text{rel}$ | (Release) |
|       | $ $          | $\text{ra}$  | (Release/Acquire)         |          | $ $          | $\text{acq}$ | (Acquire) |
|       | $ $          | $\text{sc}$  | (Sequentially Consistent) |          | $ $          | $\text{sc}$  | (SC)      |

## Language (See paper for RMWs and address calculation)

$$\begin{aligned}\llbracket \text{skip} \rrbracket &\triangleq \{\checkmark\} \\ \llbracket r := M; C \rrbracket &\triangleq \llbracket C \rrbracket[M/r] \\ \llbracket r := x^\mu; C \rrbracket &\triangleq \bigcup_v (R^\mu x v) \Rightarrow \llbracket C \rrbracket[x/r] \\ \llbracket x^\mu := M; C \rrbracket &\triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow \llbracket C \rrbracket[M/x] \\ \llbracket F^\kappa; C \rrbracket &\triangleq (F^\kappa) \Rightarrow \llbracket C \rrbracket \\ \llbracket \text{if}(M)\{C\} \text{ else }\{D\} \rrbracket &\triangleq (M \triangleright \llbracket C \rrbracket) \parallel (\neg M \triangleright \llbracket D \rrbracket) \\ \llbracket C \parallel D \rrbracket &\triangleq \llbracket C \rrbracket \parallel \llbracket D \rrbracket \\ \llbracket \text{var } x; C \rrbracket &\triangleq \nu x . \llbracket C \rrbracket\end{aligned}$$

|           |                 |                           |              |              |           |
|-----------|-----------------|---------------------------|--------------|--------------|-----------|
| $\mu ::=$ | $r\downarrow x$ | (Relaxed)                 | $\kappa ::=$ | $\text{rel}$ | (Release) |
|           | $\text{ra}$     | (Release/Acquire)         |              | $\text{acq}$ | (Acquire) |
|           | $\text{sc}$     | (Sequentially Consistent) |              | $\text{sc}$  | (SC)      |

## Language (See paper for RMWs and address calculation)

$$\begin{aligned}\llbracket \text{skip} \rrbracket &\triangleq \{\checkmark\} \\ \llbracket r := M; C \rrbracket &\triangleq \llbracket C \rrbracket[M/r] \\ \llbracket r := x^\mu; C \rrbracket &\triangleq \bigcup_v (R^\mu x v) \Rightarrow \llbracket C \rrbracket[x/r] \\ \llbracket x^\mu := M; C \rrbracket &\triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow \llbracket C \rrbracket[M/x] \\ \llbracket F^\kappa; C \rrbracket &\triangleq (F^\kappa) \Rightarrow \llbracket C \rrbracket \\ \llbracket \text{if}(M)\{C\} \text{ else }\{D\} \rrbracket &\triangleq (M \triangleright \llbracket C \rrbracket) \parallel (\neg M \triangleright \llbracket D \rrbracket) \\ \llbracket C \parallel D \rrbracket &\triangleq \llbracket C \rrbracket \parallel \llbracket D \rrbracket \\ \llbracket \text{var } x; C \rrbracket &\triangleq \nu x . \llbracket C \rrbracket\end{aligned}$$

|           |                 |                           |              |              |           |
|-----------|-----------------|---------------------------|--------------|--------------|-----------|
| $\mu ::=$ | $r\downarrow x$ | (Relaxed)                 | $\kappa ::=$ | $\text{rel}$ | (Release) |
|           | $\text{ra}$     | (Release/Acquire)         |              | $\text{acq}$ | (Acquire) |
|           | $\text{sc}$     | (Sequentially Consistent) |              | $\text{sc}$  | (SC)      |

## Language (See paper for RMWs and address calculation)

$$\begin{aligned}\llbracket \text{skip} \rrbracket &\triangleq \{\checkmark\} \\ \llbracket r := M; C \rrbracket &\triangleq \llbracket C \rrbracket[M/r] \\ \llbracket r := x^\mu; C \rrbracket &\triangleq \bigcup_v (R^\mu x v) \Rightarrow \llbracket C \rrbracket[x/r] \\ \llbracket x^\mu := M; C \rrbracket &\triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow \llbracket C \rrbracket[M/x] \\ \llbracket F^\kappa; C \rrbracket &\triangleq (F^\kappa) \Rightarrow \llbracket C \rrbracket \\ \llbracket \text{if}(M)\{C\} \text{ else }\{D\} \rrbracket &\triangleq (M \triangleright \llbracket C \rrbracket) \parallel (\neg M \triangleright \llbracket D \rrbracket) \\ \llbracket C \parallel D \rrbracket &\triangleq \llbracket C \rrbracket \parallel \llbracket D \rrbracket \\ \llbracket \text{var } x; C \rrbracket &\triangleq \nu x . \llbracket C \rrbracket\end{aligned}$$

|           |                 |                           |              |              |           |
|-----------|-----------------|---------------------------|--------------|--------------|-----------|
| $\mu ::=$ | $r\downarrow x$ | (Relaxed)                 | $\kappa ::=$ | $\text{rel}$ | (Release) |
|           | $\text{ra}$     | (Release/Acquire)         |              | $\text{acq}$ | (Acquire) |
|           | $\text{sc}$     | (Sequentially Consistent) |              | $\text{sc}$  | (SC)      |

## Language (See paper for RMWs and address calculation)

$$\begin{aligned}\llbracket \text{skip} \rrbracket &\triangleq \{\checkmark\} \\ \llbracket r := M; C \rrbracket &\triangleq \llbracket C \rrbracket[M/r] \\ \llbracket r := x^\mu; C \rrbracket &\triangleq \bigcup_v (R^\mu x v) \Rightarrow \llbracket C \rrbracket[x/r] \\ \llbracket x^\mu := M; C \rrbracket &\triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow \llbracket C \rrbracket[M/x] \\ \llbracket F^\kappa; C \rrbracket &\triangleq (F^\kappa) \Rightarrow \llbracket C \rrbracket \\ \llbracket \text{if}(M)\{C\} \text{ else }\{D\} \rrbracket &\triangleq (\textcolor{red}{M} \triangleright \llbracket C \rrbracket) \parallel (\neg \textcolor{red}{M} \triangleright \llbracket D \rrbracket) \\ \llbracket C \parallel D \rrbracket &\triangleq \llbracket C \rrbracket \parallel \llbracket D \rrbracket \\ \llbracket \text{var } x; C \rrbracket &\triangleq \nu x . \llbracket C \rrbracket\end{aligned}$$

|           |                 |                           |              |                 |           |
|-----------|-----------------|---------------------------|--------------|-----------------|-----------|
| $\mu ::=$ | $r\downarrow x$ | (Relaxed)                 | $\kappa ::=$ | $\text{rel}$    | (Release) |
|           | $  \text{ ra}$  | (Release/Acquire)         |              | $  \text{ acq}$ | (Acquire) |
|           | $  \text{ sc}$  | (Sequentially Consistent) |              | $  \text{ sc}$  | (SC)      |

## Language (See paper for RMWs and address calculation)

$$\begin{aligned}\llbracket \text{skip} \rrbracket &\triangleq \{\checkmark\} \\ \llbracket r := M; C \rrbracket &\triangleq \llbracket C \rrbracket[M/r] \\ \llbracket r := x^\mu; C \rrbracket &\triangleq \bigcup_v (R^\mu x v) \Rightarrow \llbracket C \rrbracket[x/r] \\ \llbracket x^\mu := M; C \rrbracket &\triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow \llbracket C \rrbracket[M/x] \\ \llbracket F^\kappa; C \rrbracket &\triangleq (F^\kappa) \Rightarrow \llbracket C \rrbracket \\ \llbracket \text{if}(M)\{C\} \text{ else }\{D\} \rrbracket &\triangleq (M \triangleright \llbracket C \rrbracket) \parallel (\neg M \triangleright \llbracket D \rrbracket) \\ \llbracket C \parallel D \rrbracket &\triangleq \llbracket C \rrbracket \parallel \llbracket D \rrbracket \\ \llbracket \text{var } x; C \rrbracket &\triangleq \nu x . \llbracket C \rrbracket\end{aligned}$$

|           |                 |                           |              |              |           |
|-----------|-----------------|---------------------------|--------------|--------------|-----------|
| $\mu ::=$ | $r\downarrow x$ | (Relaxed)                 | $\kappa ::=$ | $\text{rel}$ | (Release) |
|           | $ra$            | (Release/Acquire)         |              | $acq$        | (Acquire) |
|           | $sc$            | (Sequentially Consistent) |              | $sc$         | (SC)      |

## Language (See paper for RMWs and address calculation)

$$\begin{aligned}\llbracket \text{skip} \rrbracket &\triangleq \{\checkmark\} \\ \llbracket r := M; C \rrbracket &\triangleq \llbracket C \rrbracket[M/r] \\ \llbracket r := x^\mu; C \rrbracket &\triangleq \bigcup_v (R^\mu x v) \Rightarrow \llbracket C \rrbracket[x/r] \\ \llbracket x^\mu := M; C \rrbracket &\triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow \llbracket C \rrbracket[M/x] \\ \llbracket F^\kappa; C \rrbracket &\triangleq (F^\kappa) \Rightarrow \llbracket C \rrbracket \\ \llbracket \text{if}(M)\{C\} \text{ else }\{D\} \rrbracket &\triangleq (M \triangleright \llbracket C \rrbracket) \parallel (\neg M \triangleright \llbracket D \rrbracket) \\ \llbracket C \parallel D \rrbracket &\triangleq \llbracket C \rrbracket \parallel \llbracket D \rrbracket \\ \llbracket \text{var } x; C \rrbracket &\triangleq \nu x . \llbracket C \rrbracket\end{aligned}$$

|           |                 |                           |              |              |           |
|-----------|-----------------|---------------------------|--------------|--------------|-----------|
| $\mu ::=$ | $r\downarrow x$ | (Relaxed)                 | $\kappa ::=$ | $\text{rel}$ | (Release) |
|           | $\text{ra}$     | (Release/Acquire)         |              | $\text{acq}$ | (Acquire) |
|           | $\text{sc}$     | (Sequentially Consistent) |              | $\text{sc}$  | (SC)      |

A vibrant field of sunflowers in full bloom, their bright yellow petals and dark brown centers contrasting against a backdrop of green leaves and stems. A large, semi-transparent white text overlay reads "4 SLIDES" in a bold, sans-serif font. The number "4" is positioned on the left, and "SLIDES" is on the right, with a thin vertical line separating the two words.

4 SLIDES

# Sequential Semantics: Preconditions

$z := r$

$r=0 \mid Wz0$

Precondition records dependence on  $r$

$$[r := x^\mu ; C] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [C][x/r]$$

$$[x^\mu := M ; C] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [C][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

(p1)  $E' = E \cup \{d\}$

(p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$

(p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$

(p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

# Sequential Semantics: Preconditions

$z := r$

$r=1 \mid Wz1$

One pomset for each value

$$[r := x^\mu ; C] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [C][x/r]$$

$$[x^\mu := M ; C] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [C][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

(p1)  $E' = E \cup \{d\}$

(p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$

(p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$

(p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

# Sequential Semantics: Preconditions

$z := r$

$r=2 \mid Wz2$

One pomset for each value

$$[r := x^\mu ; C] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [C][x/r]$$

$$[x^\mu := M ; C] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [C][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

(p1)  $E' = E \cup \{d\}$

(p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$

(p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$

(p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

# Sequential Semantics: Preconditions

$x := r; z := r$

$r=0 \mid Wx0$      $r=0 \mid Wz0$

Two writes = two events per pomset

$$[r := x^\mu; C] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [C][x/r]$$

$$[x^\mu := M; C] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [C][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

(p1)  $E' = E \cup \{d\}$

(p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$

(p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$

(p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

# Sequential Semantics: Preconditions

$x := r; z := r$

$r=1 \mid Wx1$      $r=1 \mid Wz1$

Two writes = two events per pomset

$$[r := x^\mu ; C] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [C][x/r]$$

$$[x^\mu := M ; C] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [C][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

- (p1)  $E' = E \cup \{d\}$
- (p4b) if  $d \neq (R..)$       then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$
- (p4c) if  $d = (Rvx)$       then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$
- (p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

# Sequential Semantics: Preconditions

$x := r; z := r$

$r=1 \mid Wx1$      $r=2 \mid Wz2$

Preconditions must be consistent  
Pomset = a single execution

$$[r := x^\mu; C] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [C][x/r]$$

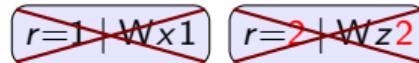
$$[x^\mu := M; C] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [C][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

- (p1)  $E' = E \cup \{d\}$
- (p4b) if  $d \neq (R..)$       then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$
- (p4c) if  $d = (Rvx)$       then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$
- (p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

# Sequential Semantics: Preconditions

$x := r; z := r$



Preconditions must be consistent  
Pomset = a single execution

$$[r := x^\mu ; C] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [C][x/r]$$

$$[x^\mu := M ; C] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [C][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

- (p1)  $E' = E \cup \{d\}$
- (p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$
- (p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$
- (p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

# Sequential Semantics: Preconditions

 $x := r; z := r$  $r=1 \mid Wx1$  $r=1 \mid Wz1$  $x := 1$  $1=1 \mid Wx1$ 

A separate program

$$[r := x^\mu; C] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [C][x/r]$$

$$[x^\mu := M; C] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [C][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

(p1)  $E' = E \cup \{d\}$

(p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$

(p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$

(p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

# Sequential Semantics: Preconditions

$$\begin{array}{ll} \text{if}(r)\{x:=r; z:=r\} & \text{if}(\neg r)\{x:=1\} \\ \boxed{r \neq 0 \wedge r=1 \mid Wx1} \quad \boxed{r \neq 0 \wedge r=1 \mid Wz1} & \boxed{r=0 \wedge 1=1 \mid Wx1} \end{array}$$

Adding control

$$[r := x^\mu ; C] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [C][x/r]$$

$$[x^\mu := M ; C] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [C][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

- (p1)  $E' = E \cup \{d\}$
- (p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$
- (p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$
- (p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

# Sequential Semantics: Preconditions

$\text{if}(r)\{x := r; z := r\}$

$r = 1 \mid Wx1$

$\text{if}(\neg r)\{x := 1\}$

$r = 0 \mid Wx1$

Simplifying

$$[r := x^\mu ; C] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [C][x/r]$$

$$[x^\mu := M ; C] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [C][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

(p1)  $E' = E \cup \{d\}$

(p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$

(p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$

(p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

# Sequential Semantics: Preconditions

$\text{if}(r)\{x := r; z := r\}$

$r=1 \mid Wx1$

$\text{if}(\neg r)\{x := 1\}$

$r=0 \mid Wx1$

Combining both sides

$\text{if}(r)\{x := r; z := r\} \text{ else } \{x := 1\}$

$r=1 \vee r=0 \mid Wx1$

We can coalesce the writes to  $x$

$$[r := x^\mu; C] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [C][x/r]$$

$$[x^\mu := M; C] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [C][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

(p1)  $E' = E \cup \{d\}$

(p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$

(p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$

(p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

# Sequential Semantics: Preconditions

$\text{if}(r)\{x := r; z := r\}$

$r=1 \mid Wx1$

$r=1 \mid Wz1$

$\text{if}(\neg r)\{x := 2\}$

$r=0 \mid Wx2$

Combining both sides

$\text{if}(r)\{x := r; z := r\} \text{ else } \{x := 2\}$

$r=1 \mid Wx1$

$r=1 \mid Wz1$

$r=0 \mid Wx2$

We *cannot* coalesce the writes to  $x$

$$[\![r := x^\mu ; C]\!] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [\![C]\!][x/r]$$

$$[\![x^\mu := M ; C]\!] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [\![C]\!][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

(p1)  $E' = E \cup \{d\}$

(p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$

(p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$

(p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

# Sequential Semantics: Preconditions

$\text{if}(r)\{x := r; z := r\}$

$r=1 \mid Wx1$

$r=1 \mid Wz1$

$\text{if}(\neg r)\{x := 2\}$

$r=0 \mid Wx2$

Combining both sides

$\text{if}(r)\{x := r; z := r\} \text{ else } \{x := 2\}$

~~$r=1 \mid Wx1$~~

~~$r=1 \mid Wz1$~~

~~$r=0 \mid Wx2$~~

Preconditions must be consistent

$$[\![r := x^\mu ; C]\!] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [\![C]\!][x/r]$$

$$[\![x^\mu := M ; C]\!] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [\![C]\!][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

(p1)  $E' = E \cup \{d\}$

(p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$

(p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$

(p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

# Sequential Semantics: Preconditions

$\text{if}(r)\{x:=r; z:=r\} \text{ else } \{x:=1\}$

$r=1 \vee r=0 \mid Wx1$

$r=1 \mid Wz1$

Let's start fresh here

$$[r := x^\mu ; C] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [C][x/r]$$

$$[x^\mu := M ; C] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [C][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

(p1)  $E' = E \cup \{d\}$

(p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$

(p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$

(p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

# Sequential Semantics: Preconditions

$r := y; \text{if}(r)\{x := r; z := r\} \text{else } \{x := 1\}$

$y = 1 \vee y = 0 \mid Wx1$

$y = 1 \mid Wz1$

To prepend,  $r := y$ , we first substitute  $[y/r]$

$$[r := x^\mu; C] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [C][x/r]$$

$$[x^\mu := M; C] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [C][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

(p1)  $E' = E \cup \{d\}$

(p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$

(p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$

(p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

# Sequential Semantics: Preconditions

$r := y; \text{if}(r)\{x := r; z := r\} \text{else } \{x := 1\}$

Ry1

$(y=1 \vee y=0) \wedge (1=1 \vee 1=0) \mid Wx1$

$(y=1) \wedge (1=1) \mid Wz1$

We then add the action, and its constraint:  $\phi' \models \phi[v/y]$

$$[\![r := x^\mu ; C]\!] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [\![C]\!][x/r]$$

$$[\![x^\mu := M ; C]\!] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [\![C]\!][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

(p1)  $E' = E \cup \{d\}$

(p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$

(p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$

(p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

# Sequential Semantics: Preconditions

$r := y; \text{if}(r)\{x := r; z := r\} \text{else } \{x := 1\}$

Ry2

$(y=1 \vee y=0) \wedge (2=1 \vee 2=0) \mid Wx1$

$(y=1) \wedge (2=1) \mid Wz1$

Incompatible reads are disallowed

$$[\![r := x^\mu ; C]\!] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [\![C]\!][x/r]$$

$$[\![x^\mu := M ; C]\!] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [\![C]\!][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

(p1)  $E' = E \cup \{d\}$

(p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$

(p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$

(p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

# Sequential Semantics: Preconditions

$r := y; \text{if}(r)\{x := r; z := r\} \text{else } \{x := 1\}$

~~Ry2~~

~~$(y=1 \vee y=0) \wedge (2=1 \vee 2=0) \mid Wx1$~~

~~$(y=1) \wedge (2=1) \mid Wz1$~~

Incompatible reads are disallowed

$$[\![r := x^\mu ; C]\!] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [\![C]\!][x/r]$$

$$[\![x^\mu := M ; C]\!] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [\![C]\!][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

(p1)  $E' = E \cup \{d\}$

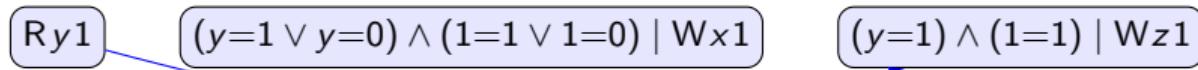
(p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$

(p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$

(p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

# Sequential Semantics: Preconditions

$r := y; \text{if}(r)\{x := r; z := r\} \text{else } \{x := 1\}$



Order may be introduced

$$[\![r := x^\mu ; C]\!] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [\![C]\!][x/r]$$

$$[\![x^\mu := M ; C]\!] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [\![C]\!][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

(p1)  $E' = E \cup \{d\}$

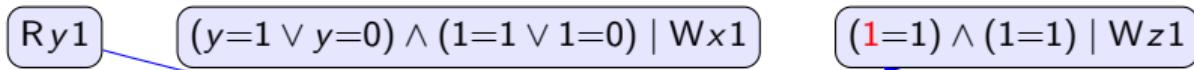
(p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$

(p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$

(p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

# Sequential Semantics: Preconditions

$r := y; \text{if}(r)\{x := r; z := r\} \text{else } \{x := 1\}$



Order enables substitution  $[v/y]$

$$[\![r := x^\mu ; C]\!] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [\![C]\!][x/r]$$

$$[\![x^\mu := M ; C]\!] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [\![C]\!][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

(p1)  $E' = E \cup \{d\}$

(p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$

(p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$

(p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

# Sequential Semantics: Preconditions

$y := 0; r := y; \text{if}(r)\{x := r; z := r\} \text{else } \{x := 1\}$

$0=0 \mid W y 0$

$R y 1$

$(0=1 \vee 0=0) \wedge (1=1 \vee 1=0) \mid W x 1$

$(1=1) \wedge (1=1) \mid W z 1$

Prepending  $y := 0$  substitutes  $[0/y]$   
Order imposed by sub? Write  $\times$  Read  $\checkmark$

$$[r := x^\mu; C] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [C][x/r]$$

$$[x^\mu := M; C] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [C][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

(p1)  $E' = E \cup \{d\}$

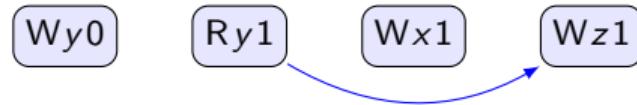
(p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$

(p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$

(p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

# Sequential Semantics: Preconditions

$y := 0; r := y; \text{if}(r)\{x := r; z := r\} \text{else } \{x := 1\}$



Simplifying tautologies

$$[r := x^\mu; C] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [C][x/r]$$

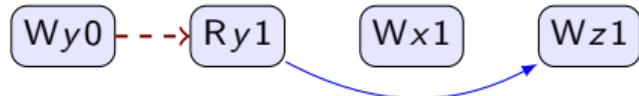
$$[x^\mu := M; C] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [C][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

- (p1)  $E' = E \cup \{d\}$
- (p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$
- (p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$
- (p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

# Concurrent Semantics: Fulfillment

$y := 0; r := y; \text{if}(r)\{x := r; z := r\} \text{else } \{x := 1\}$



Must preserve order on *conflicting* accesses (WW, WR)

$$[r := x^\mu; C] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [C][x/r]$$

$$[x^\mu := M; C] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [C][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

(p1)  $E' = E \cup \{d\}$

(p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$

(p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$

(p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

(p5b) if  $d$  and  $e$  are conflicting actions then  $d \leq' e$

# Concurrent Semantics: Fulfillment

$x := 0; y := 0; r := y; \text{if}(r)\{x := r; z := r\} \text{else } \{x := 1\}$



Initializing  $x$

$$[r := x^\mu; C] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [C][x/r]$$

$$[x^\mu := M; C] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [C][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

- (p1)  $E' = E \cup \{d\}$
- (p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$
- (p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$
- (p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$
- (p5b) if  $d$  and  $e$  are conflicting actions then  $d \leq' e$

# Concurrent Semantics: Fulfillment

$x := 0; y := 0; (y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else } \{x := 1\})$



Introducing  $y := x$

$$[r := x^\mu; C] \triangleq \bigcup_v (R^\mu x v) \Rightarrow [C][x/r]$$

$$[x^\mu := M; C] \triangleq \bigcup_v (M = v \mid W^\mu x v) \Rightarrow [C][M/x]$$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

- (p1)  $E' = E \cup \{d\}$
- (p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$
- (p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$
- (p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$
- (p5b) if  $d$  and  $e$  are conflicting actions then  $d \leq' e$

# Concurrent Semantics: Fulfillment

$x := 0; y := 0; (y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else } \{x := 1\})$       (\*) ✓



Fulfillment requirements

We say  $\mathcal{A}(d) = (Wxv)$  fulfills  $\mathcal{A}(e) = (Rxv)$  if

- ▶  $d < e$
- ▶ if  $\mathcal{A}(c) = (Wx..)$  then either  $c \leq d$  or  $e \leq c$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

- (p1)  $E' = E \cup \{d\}$
- (p4b) if  $d \neq (R..)$       then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$
- (p4c) if  $d = (Rvx)$       then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$
- (p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$
- (p5b) if  $d$  and  $e$  are conflicting actions then  $d \leq' e$

# Concurrent Semantics: Fulfillment

$x := 0; y := 0; (y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else } \{x := 1\})$        $(*)$  ✓



Allowed!

We say  $\mathcal{A}(d) = (Wxv)$  fulfills  $\mathcal{A}(e) = (Rxv)$  if

- ▶  $d < e$
- ▶ if  $\mathcal{A}(c) = (Wx..)$  then either  $c \leq d$  or  $e \leq c$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

- (p1)  $E' = E \cup \{d\}$
- (p4b) if  $d \neq (R..)$       then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$
- (p4c) if  $d = (Rvx)$       then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$
- (p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$
- (p5b) if  $d$  and  $e$  are conflicting actions then  $d \leq' e$

# Concurrent Semantics: Fulfillment

$x := 0; y := 0; (y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else } \{x := 1\})$       (\*) ✓



Allowed!  
Partial order

We say  $\mathcal{A}(d) = (Wxv)$  fulfills  $\mathcal{A}(e) = (Rxv)$  if

- ▶  $d < e$
- ▶ if  $\mathcal{A}(c) = (Wx..)$  then either  $c \leq d$  or  $e \leq c$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

- (p1)  $E' = E \cup \{d\}$
- (p4b) if  $d \neq (R..)$       then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$
- (p4c) if  $d = (Rvx)$       then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$
- (p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$
- (p5b) if  $d$  and  $e$  are conflicting actions then  $d \leq' e$

# Concurrent Semantics: Fulfillment

$x := 0; y := 0; (y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else } \{x := 1\})$       (\*) ✓



Allowed!  
Partial order  
All reads fulfilled

We say  $\mathcal{A}(d) = (Wxv)$  fulfills  $\mathcal{A}(e) = (Rxv)$  if

- ▶  $d < e$
- ▶ if  $\mathcal{A}(c) = (Wx..)$  then either  $c \leq d$  or  $e \leq c$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

- (p1)  $E' = E \cup \{d\}$
- (p4b) if  $d \neq (R..)$       then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$
- (p4c) if  $d = (Rvx)$       then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$
- (p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$
- (p5b) if  $d$  and  $e$  are conflicting actions then  $d \leq' e$

# Concurrent Semantics: Fulfillment

$x := 0; y := 0; (y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else } \{x := 1\})$       (\*) ✓



Allowed!  
Partial order  
All reads fulfilled  
All preconditions tautologies

We say  $\mathcal{A}(d) = (Wxv)$  fulfills  $\mathcal{A}(e) = (Rxv)$  if

- ▶  $d < e$
- ▶ if  $\mathcal{A}(c) = (Wx..)$  then either  $c \leq d$  or  $e \leq c$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

- (p1)  $E' = E \cup \{d\}$
- (p4b) if  $d \neq (R..)$       then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$
- (p4c) if  $d = (Rvx)$       then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$
- (p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$
- (p5b) if  $d$  and  $e$  are conflicting actions then  $d \leq' e$

# Concurrent Semantics: Fulfillment

$x := 0; y := 0; (y := x \parallel r := y; \text{if}(r)\{x := r; z := r\} \text{else } \{x := 2\})$  (OOTA3) X



Disallow!

Cycle

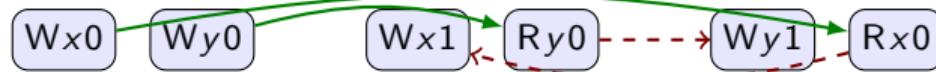
We say  $\mathcal{A}(d) = (Wxv)$  fulfills  $\mathcal{A}(e) = (Rxv)$  if

- ▶  $d < e$
- ▶ if  $\mathcal{A}(c) = (Wx..)$  then either  $c \leq d$  or  $e \leq c$

Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

- (p1)  $E' = E \cup \{d\}$
- (p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$
- (p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$
- (p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$
- (p5b) if  $d$  and  $e$  are conflicting actions then  $d \leq' e$

# Buffering

$$x := 0; y := 0; (x := 1; r := y \parallel y := 1; r := x)$$


(SB) ✓

$$r := y; x := 1 \parallel r := x; y := 1$$


(LB) ✓

# Synchronization and Fences

- Let  $P' \in (\phi \mid a) \Rightarrow \mathcal{P}$  when  $(\exists P \in \mathcal{P}) (\forall e \in E)$

(p1)  $E' = E \cup \{d\}$

(p2)  $\leq' \supseteq \leq$

(p3a)  $\mathcal{A}'(e) = \mathcal{A}(e)$

(p3b)  $\mathcal{A}'(d) = a$

(p4a)  $\Phi'(d)$  implies  $\phi \wedge (d \notin E \vee \Phi(d))$

(p4b) if  $d \neq (R..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$

(p4c) if  $d = (Rvx)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)[v/x]$

(p5a) if  $d = (R..)$ ,  $e = (W..)$  then  $e = d$  or  $\Phi'(e)$  implies  $\Phi(e)$  or  $d \leq' e$

(p5b) if  $d$  and  $e$  are conflicting actions then  $d \leq' e$

(p5c) if  $d$  is an acquire or  $e$  is a release then  $d \leq' e$

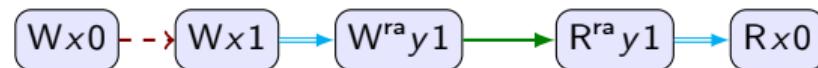
(p5d) if  $d$  is an SC write and  $e$  is an SC read then  $d \leq' e$

(p5e) if  $d$  reads, and  $e$  is an acquiring fence, then  $d \leq' e$

(p5f) if  $d$  is a releasing fence, and  $e$  writes, then  $d \leq' e$

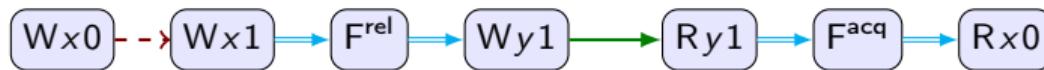
# Publication

$x := 0; x := 1; y^{\text{ra}} := 1 \parallel r := y^{\text{ra}}; s := x$



(PUB1) X

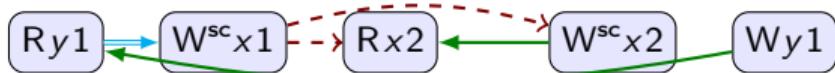
$x := 0; x := 1; F^{\text{rel}}; y := 1 \parallel r := y; F^{\text{acq}}; s := x$



(PUB2) X

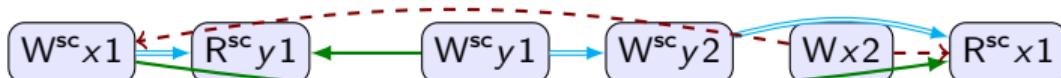
# SC Access/Fences

$r := y; x^{sc} := 1; s := x \parallel x^{sc} := 2; y := 1$



(SC1) ✓

$x^{sc} := 1; r := y^{sc} \parallel y^{sc} := 1; y^{sc} := 2; x := 2; s := x^{sc}$



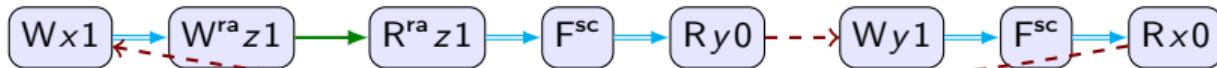
(SC2) ✓

$x := 1 \parallel r := x; F^{sc}; r := y \parallel y := 1; F^{sc}; r := x$



(SC3) ✗

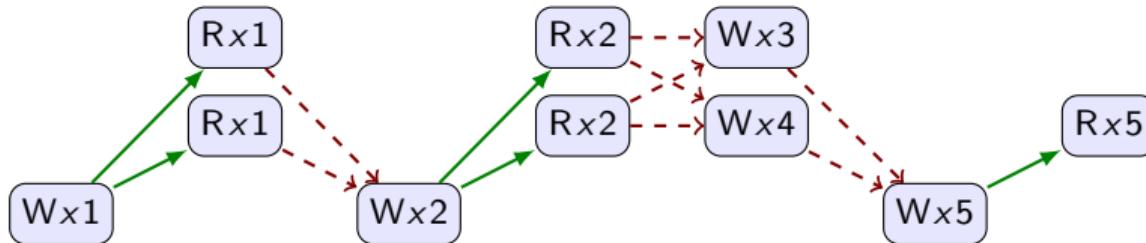
$x := 1; z^{ra} := 1; \parallel r^{ra} := z; F^{sc}; r := y \parallel y := 1; F^{sc}; r := x$



(SC4) ✗

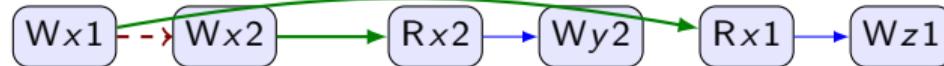
# Coherence

$x := 1 \parallel x := 2 \parallel x := 3 \parallel x := 4 \parallel x := 5 \parallel r := x; r := x; r := x; r := x; r := x$



(CO1) ✓

$x := 1; x := 2 \parallel y := x; z := x$



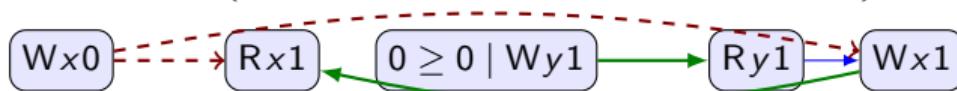
(CO2) ✓

$r := x; x := 1 \parallel s := x; x := 2$

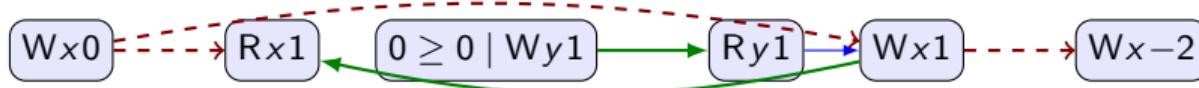


(TC16) ✗

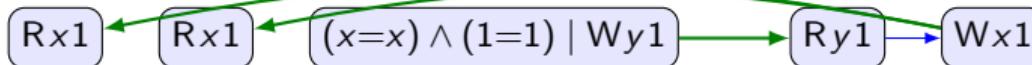
# Internal Reads

$$x := 0; (r := x; \text{if}(r \geq 0)\{y := 1\} \parallel x := y)$$


(TC1) ✓

$$x := 0; (r := x; \text{if}(r \geq 0)\{y := 1\} \parallel x := y \parallel x := -2)$$


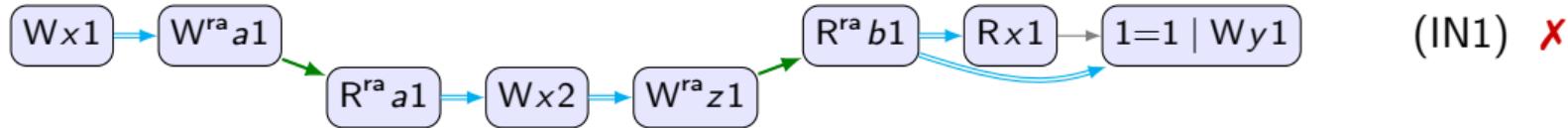
(TC9) ✓

$$r := x; s := x; \text{if}(r = s)\{y := 1\} \parallel x := y$$


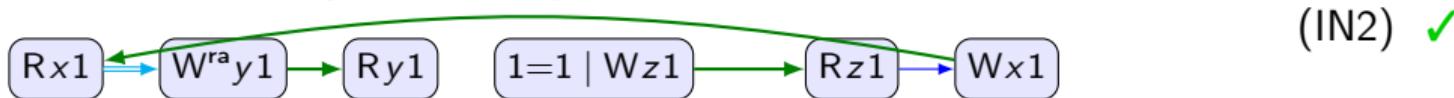
(TC2) ✓

# Internal Reads+Synchronization

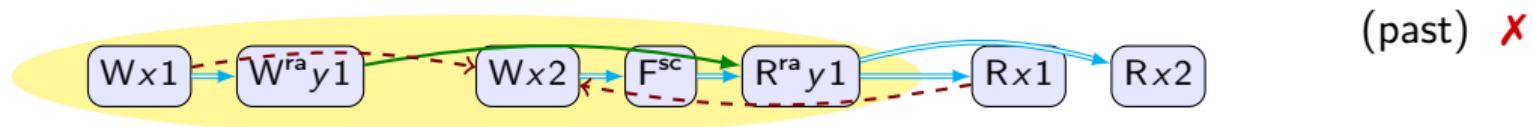
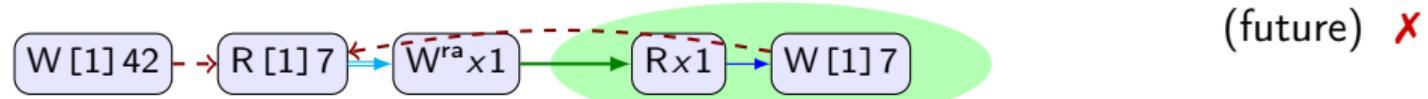
$x := 1; a^{\text{ra}} := 1; \text{if}(z^{\text{ra}})\{y := x\} \parallel \text{if}(a^{\text{ra}})\{x := 2; z^{\text{ra}} := 1\}$



$r := x; y^{\text{ra}} := 1; s := y; z := s \parallel x := z$



# Local data race freedom

$$(x := 1; y^{\text{ra}} := 1) \parallel (x := 2; F^{\text{sc}}; \text{if}(y^{\text{ra}})\{r := x; s := x\})$$

$$(r := 1; [r] := 42; s := [r]; x^{\text{ra}} := r) \parallel (r := x; [r] := 7)$$


# Valid Transformations

$$\begin{aligned}\llbracket r := x; s := y; C \rrbracket &= \llbracket s := y; r := x; C \rrbracket && \text{if } r \neq s \\ \llbracket x := M; y := N; C \rrbracket &= \llbracket y := N; x := M; C \rrbracket && \text{if } x \neq y \\ \llbracket x := M; s := y; C \rrbracket &= \llbracket s := y; x := M; C \rrbracket && \text{if } x \neq y \text{ and } s \notin \text{id}(M) \\ \llbracket x^\mu := M; s := y; C \rrbracket &\supseteq \llbracket s := y; x^\mu := M; C \rrbracket && \text{if } x \neq y \text{ and } s \notin \text{id}(M) \\ \llbracket x := M; s := y^\mu; C \rrbracket &\supseteq \llbracket s := y^\mu; x := M; C \rrbracket && \text{if } x \neq y \text{ and } s \notin \text{id}(M) \\ \llbracket F^\mu; F^\mu; C \rrbracket &\supseteq \llbracket F^\mu; s := r; C \rrbracket \\ \llbracket r := x^\mu; s := x^\mu; C \rrbracket &\supseteq \llbracket r := x^\mu; s := r; C \rrbracket \\ \llbracket r_1 := x; s := y; r_2 := x; C \rrbracket &\supseteq \llbracket r_1 := x; r_2 := r_1; s := y; C \rrbracket && \text{if } r_2 \neq s\end{aligned}$$

- Reordering: W  $\leftrightarrow$  W, R  $\leftrightarrow$  R, W  $\leftrightarrow$  R
- Roach motel: Relaxed  $\leftarrow$  Acquire, Relaxed  $\rightarrow$  Release
- Elimination: Redundant fence, Redundant read, Common Subexpression

# Valid Transformations

$$\llbracket C \parallel \text{var } x; D \rrbracket = \llbracket \text{var } x; (C \parallel D) \rrbracket \quad \text{if } x \notin \text{id}(C)$$

$$\llbracket \text{if}(M)\{C\} \text{ else }\{C\} \rrbracket \supseteq \llbracket C \rrbracket$$

$$\llbracket \text{if}(M)\{C\} \text{ else }\{C\} \rrbracket \subseteq^* \llbracket C \rrbracket$$

$$\llbracket \text{if}(M)\{C\} \text{ else }\{D\} \rrbracket = \llbracket C \rrbracket \quad \text{if } M \text{ is a tautology}$$

$$\llbracket x := M; x := N; C \rrbracket \supseteq^* \llbracket x := N; C \rrbracket$$

$$\llbracket x^\mu := M; r := x; C \rrbracket \supseteq^* \llbracket x^\mu := M; r := M; C \rrbracket$$

$$\llbracket r := x; C \rrbracket \supseteq^* \llbracket C \rrbracket \quad \text{if } r \notin \text{id}(C)$$

- Scope extrusion
- Code lifting, Case analysis\* (\*See paper)
- Elimination: Dead code, Dead store\*, Store forwarding\*, Irrelevant read\*

# Other Transformations

- Valid✓ Proof✗
  - Redundant write after read elimination
- Sound observationally✓ Valid✗
  - Access mode strengthening ( $\text{rlx} \rightarrow \text{ra} \rightarrow \text{sc}$ )
  - Commuting/Eliminating some synchronizations
  - Implementing synchronizations using fences
  - Implementing locks using synchronizations
- Sound observationally? Valid✗
  - Access mode weakening, eg ( $\text{sc} \rightarrow \text{ra} \rightarrow \text{rlx}$ )
  - Lock elision
- Sound observationally✗
  - Thread inlining
  - Relevant read introduction
  - Write introduction

## Other results and limitations

- Results
  - Compositional proof rule for PLTL
  - Efficient ARM implementation
  - Local data race freedom

## Other results and limitations

- Results
  - Compositional proof rule for PLTL
  - Efficient ARM implementation
  - Local data race freedom
- Limitations
  - No: loops or functions, sequential composition
  - No: mixed size access, separate address dependencies
  - Limited: optimizations, logic for OOTA

# Other results and limitations

- Results
  - Compositional proof rule for PLTL
  - Efficient ARM implementation
  - Local data race freedom
- Limitations
  - No: loops or functions, sequential composition
  - No: mixed size access, separate address dependencies
  - Limited: optimizations, logic for OOTA
  - MCA: ARM✓ x86-64✓ RISC-V✓ POWER✗

# Other results and limitations

- Results
  - Compositional proof rule for PLTL
  - Efficient ARM implementation
  - Local data race freedom
- Limitations
  - No: loops or functions, sequential composition
  - No: mixed size access, separate address dependencies
  - Limited: optimizations, logic for OOTA
  - MCA: ARM✓ x86-64✓ RISC-V✓ POWERX
    - Add per-location partial order:  $\sqsubseteq$
    - Require *observation*: if  $d/e$  conflict and  $d \leq e$  then  $d \sqsubseteq e$
    - Prefixing (p5b): if  $e$  conflicts then  $d \sqsubseteq' e$
    - Fulfillment (f4): if  $c$  conflicts then  $c \sqsubseteq d$  or  $e \sqsubseteq c$

# PRINCIPLES

A vibrant field of sunflowers bathed in golden sunlight. The flowers are in various stages of bloom, from tight buds to fully open blossoms with dark brown centers. The leaves are large and green, some with visible veins. In the foreground, the word "PRINCIPLES" is printed in large, bold, white, sans-serif capital letters. The letters have a slight shadow or glow effect, making them stand out against the bright background. The overall composition is a mix of nature's organic beauty and a clean, modern graphic design.

# Compositionality

A photograph of a sunflower field at sunset. The sunflowers are bright yellow with dark centers. The background is filled with more sunflowers and some green foliage. In the foreground, there is a large, slightly out-of-focus sunflower on the left. Overlaid on the right side of the image is the word "Compositionality" in a large, bold, white sans-serif font.

# Compositionality + Construction

A photograph of a sunflower field at sunset. The sunflowers are tall with large, bright yellow flowers and green leaves. The background is filled with more sunflowers, slightly blurred. In the foreground, a large, out-of-focus sunflower head is visible on the left. Overlaid on the image is the title "Compositionality + Construction" in a large, bold, white sans-serif font. The text is positioned in the upper half of the image, with "Compositionality" on top and "+ Construction" below it.

**Compositionality**  
+ Construction  
+ Reasoning  
+ Local races  
+ Safety properties

A photograph of a sunflower field in full bloom. The flowers are bright yellow with dark brown centers. Large, bold, white letters spell out "LOGIC" across the center of the image. The letter "O" has a gold-colored circular glow behind it, and the letter "G" has a similar glow. The background is filled with more sunflowers and green foliage.

LOGIC



**LOGIC**  
+ PRECONDITIONS

The background of the image is a vibrant field of sunflowers at sunset, with their bright yellow petals and green leaves filling the frame. Overlaid on this natural scene is the word "LOGIC" in large, white, sans-serif capital letters. The letter "O" is unique, featuring a circular cutout in its center that reveals a smaller sunflower head, symbolizing the connection between logic and nature.

LOGIC

+ PRECONDITIONS  
+ TEMPORAL SAFETY



ONE  
ORDER

A photograph of a sunflower field with large, bold, white text overlaid. The text reads "ONE" on the first line and "ORDER" on the second line. The letters are slightly staggered. The background consists of numerous sunflowers with their characteristic yellow petals and brown centers, set against a backdrop of green foliage and a clear blue sky. The lighting suggests a bright, sunny day.

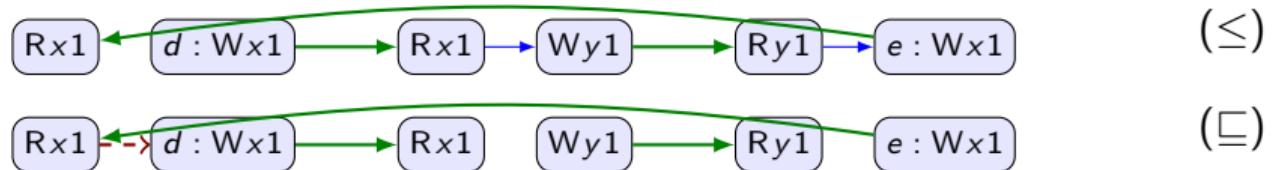
A vibrant field of sunflowers bathed in golden sunlight. In the center, the word "THANKS" is written in large, bold, white capital letters. The letters have a thin black outline and a warm, glowing yellow-to-white gradient fill, matching the color of the sunflowers. The background is filled with numerous sunflowers at various stages of bloom, from tight buds to fully open flowers with dark brown centers. The leaves are a healthy green, and the overall scene is one of a lush, sunny summer day.

THANKS

# Non-MCA Architectures?

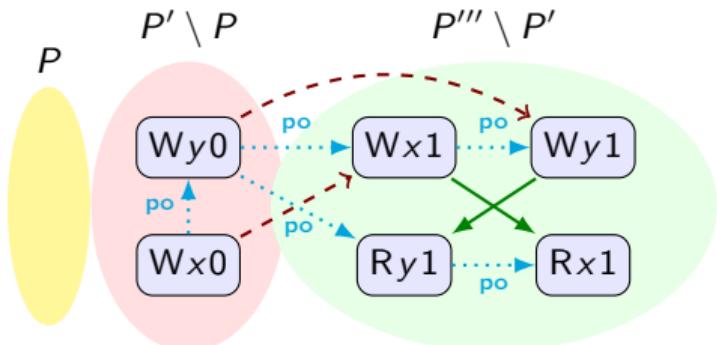
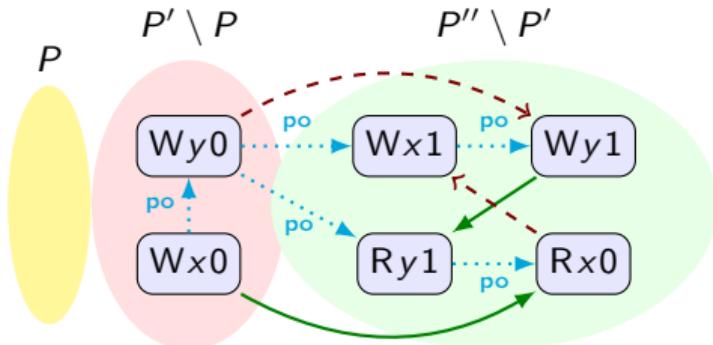
- Two partial orders:
  - ≤ causal order, as before
  - ⊑ per-location order
- Require:  $d \sqsubseteq e$  when  $d \leq e$  and they conflict (same location)
- When prefixing  $d$ :
  - (p5b) if  $d$  and  $e$  conflict then  $d \sqsubseteq' e$ ,
  - (f4) for every conflicting write  $c$ , either  $c \sqsubseteq d$  or  $e \sqsubseteq c$ .
- Example:

$r := x; x := 1 \parallel y := x \parallel x := y$



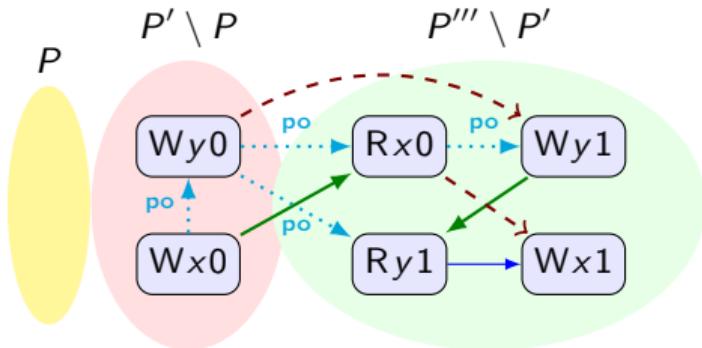
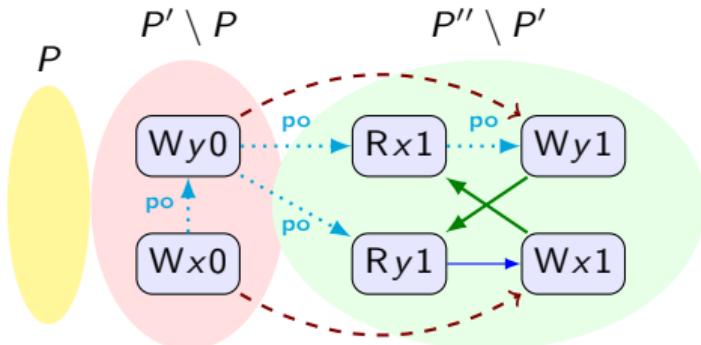
# LDRF: Reads from the Past

$x := 0; y := 0; (x := 1; y := 1 \parallel \text{if}(y)\{r := x\})$



# LDRF: Reads from the Future

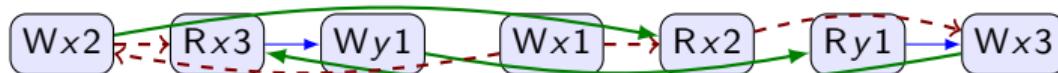
$x := 0; y := 0; (r := x; y := 1 \parallel s := y; x := s)$



## More OOTA

$$(y := x + 1 \parallel x := y)$$

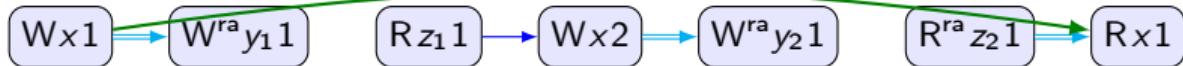

(OOTA6) X

$$x := 2; \text{if}(x \neq 2)\{y := 1\} \parallel x := 1; r := x; \text{if}(y)\{x := 3\}$$


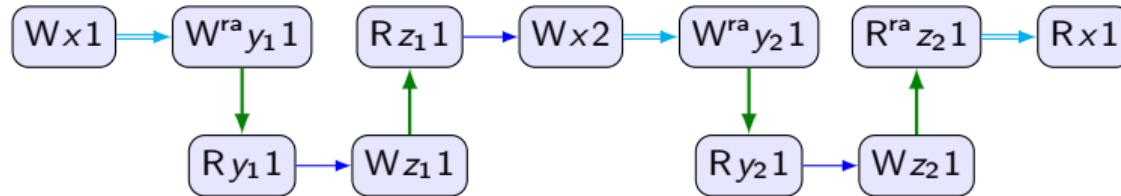
(OOTA7) X

# Blockers

$\text{var } x; (x := 1; y_1^{\text{ra}} := 1 \parallel \text{if}(z_1)\{x := 2\}; y_2^{\text{ra}} := 1 \parallel r := z_2^{\text{ra}}; s := x)$



Context  $z_1 := y_1 \parallel z_2 := y_2 \parallel [-]$

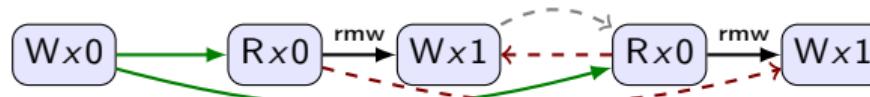


# RMWs

$\xrightarrow{\text{rmw}} \subseteq \leq:$

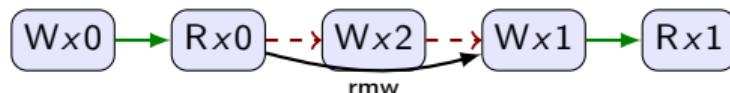
- If  $(Rx2) \xrightarrow{\text{rmw}} (Wx3)$  and  $(Wx1) \leq (Wx3)$  then  $(Wx1) \leq (Rx2)$
- If  $(Rx2) \xrightarrow{\text{rmw}} (Wx3)$  and  $(Rx2) \leq (Wx3)$  then  $(Wx3) \leq (Wx3)$
- $\xrightarrow{\text{rmw}}$  does not coalesce in  $\parallel$  or prefixing

$$x := 0; (FADD^{\text{rlx}, \text{rlx}}(x, 1) \parallel FADD^{\text{rlx}, \text{rlx}}(x, 1))$$



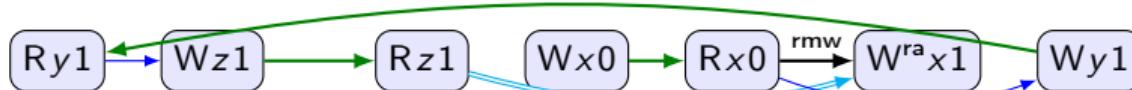
(RMW0) X

$$x := 0; s := FADD^{\text{rlx}, \text{rlx}}(x, 1) \parallel x := 2; s := x$$



(RMW1) X

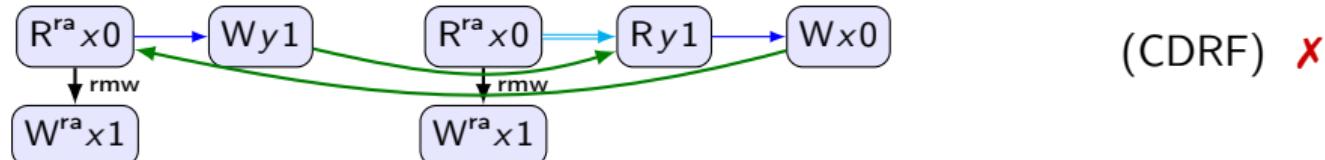
$$r := y; z := r \parallel r := z; x := 0; s := FADD^{\text{rlx}, \text{ra}}(x, 1); y := s + 1$$



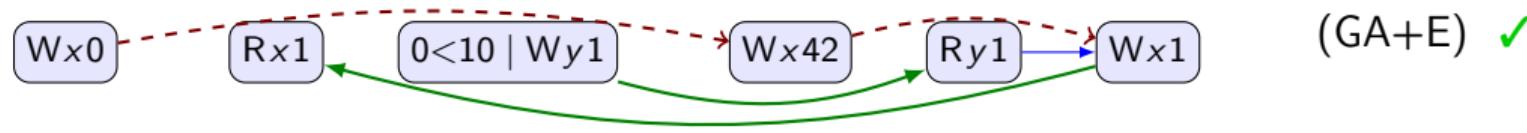
(RMW2) ✓

# PS2.0 Examples

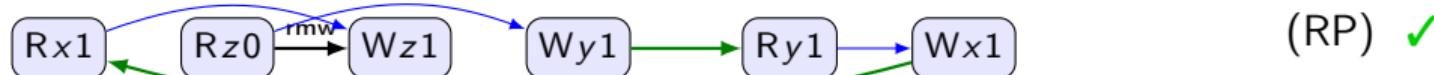
$r := \text{FADD}^{\text{ra}, \text{ra}}(x, 1); \text{if}(r=0)\{y := 1\} \parallel r := \text{FADD}^{\text{ra}, \text{ra}}(x, 1); \text{if}(r=0)\{\text{if}(y)\{x := 0\}\}$



$x := 0; (r := \text{CAS}^{\text{rlx}, \text{rlx}}(x, 0, 1); \text{if}(r < 10)\{y := 1\} \parallel x := 42; x := y)$

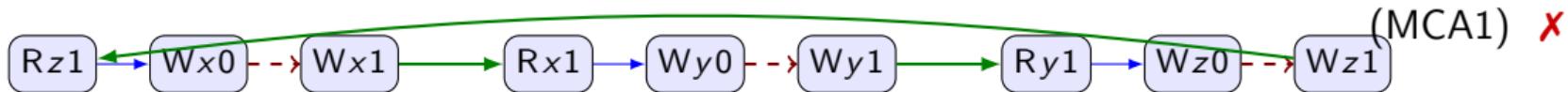


$r := x; s := \text{FADD}^{\text{rlx}, \text{rlx}}(z, r); y := s + 1 \parallel x := y$



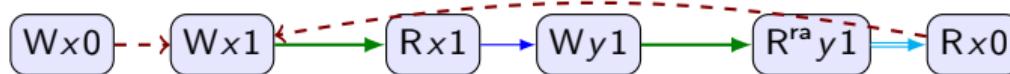
# MCA Examples

$\text{if}(z)\{x:=0\}; x:=1 \parallel \text{if}(x)\{y:=0\}; y:=1 \parallel \text{if}(y)\{z:=0\}; z:=1$



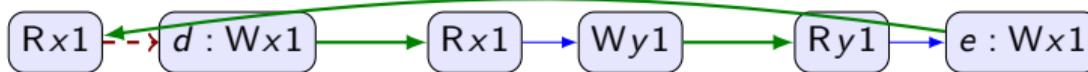
(MCA1) X

$x := 0; x := 1 \parallel y := x \parallel r := y^{\text{ra}}; s := x$



(MCA2) X

$r := x; x := 1 \parallel y := x \parallel x := y$



(MCA3) X

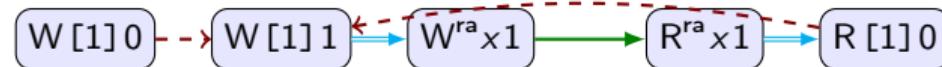
# Address Calculation Examples

$r := y; s := [r]; x := s \parallel r := x; s := [r]; y := s$



(ADDR1) X

$(r := 1; [r] := 0; [r] := 1; x^{ra} := r) \parallel (r := x^{ra}; s := [r])$



(ADDR2) X