The Leaky Semicolon

Compositional Semantic Dependencies for Relaxed-Memory Concurrency

Alan Jeffrey* James Riely† Mark Batty‡
Simon Cooksey‡ Ilya Kaysin** Anton Podkopaev††

*Roblox †DePaul ‡Kent

**JetBrains + Cambridge ††HSE

POPL
January 2022

The Leaky Semicolon

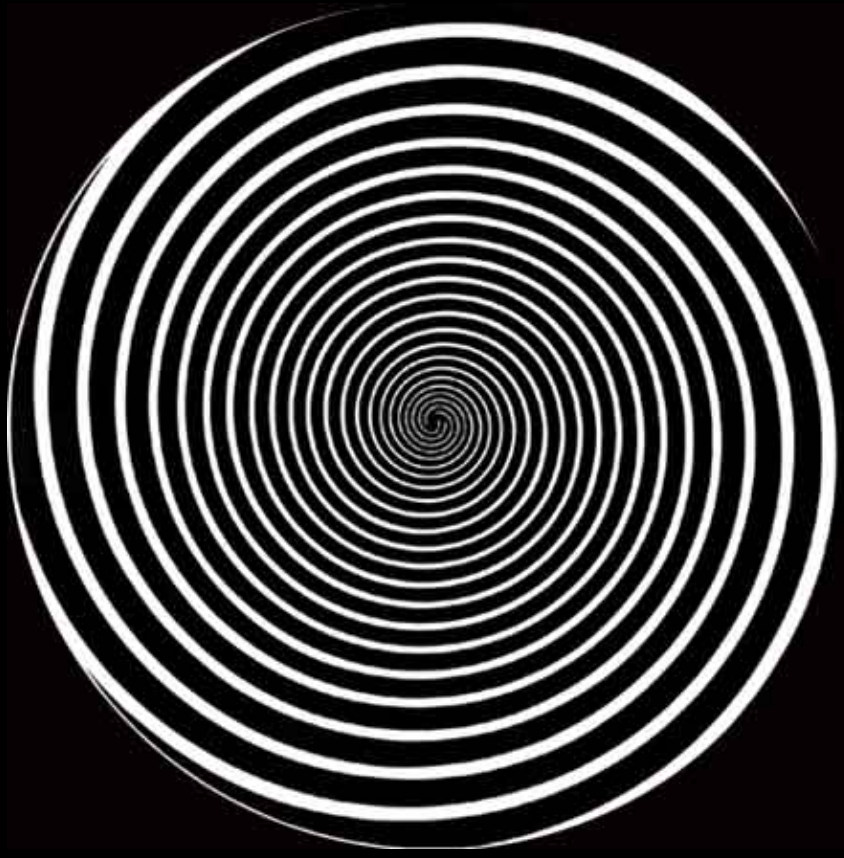
Compositional Semantic Dependencies for Relaxed-Memory Concurrency

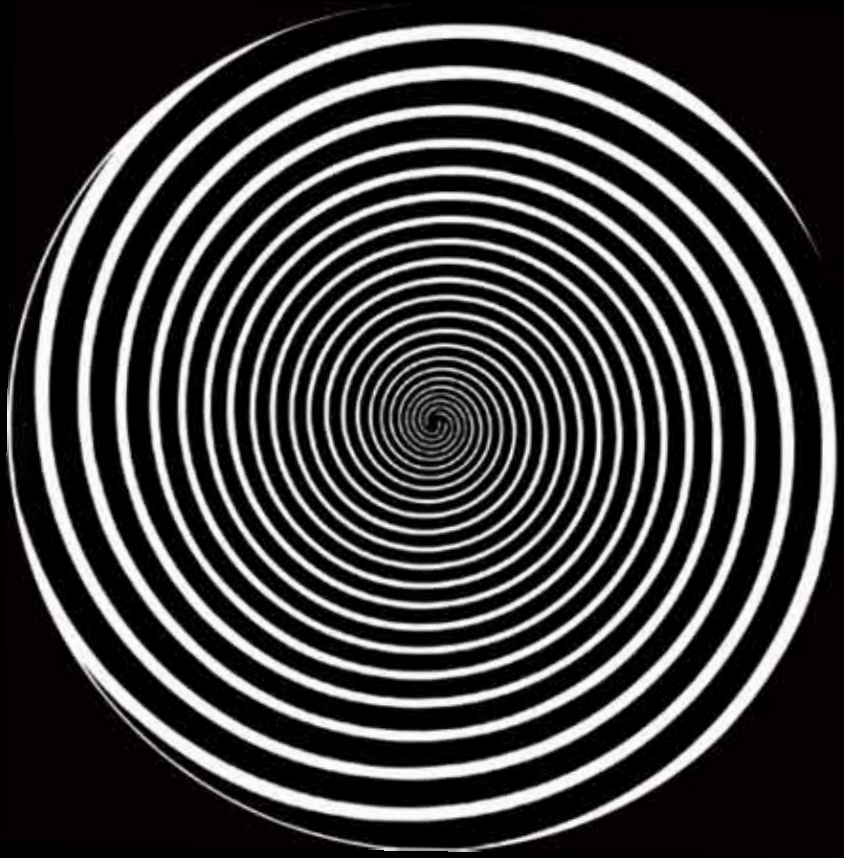
Alan Jeffrey* James Riely† Mark Batty‡
Simon Cooksey‡ Ilya Kaysin** Anton Podkopaev††

*Roblox †DePaul ‡Kent

**JetBrains + Cambridge ††HSE

POPL
January 2022





Revert to 1999 ...

Revert to 1999 ...

A model of imperative programming

Revert to 1999 ...

A model of imperative programming

Shared-memory concurrency

Revert to 1999 ...

A model of imperative programming

- Shared-memory concurrency
- Compositional for both parallelism and sequencing

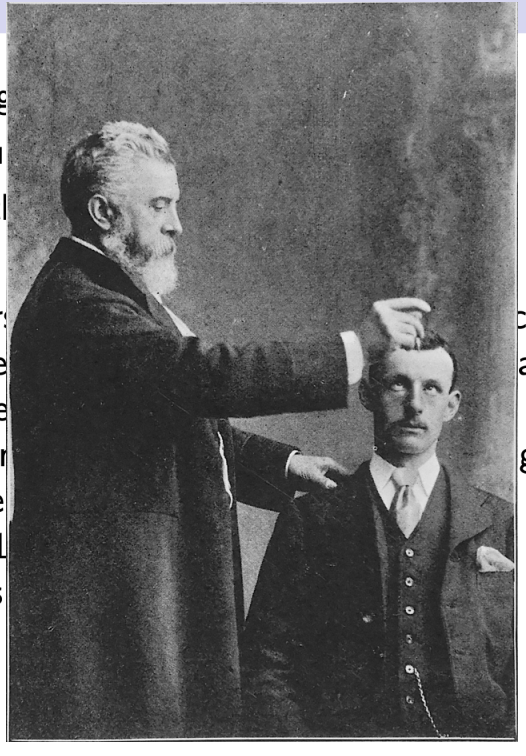
Revert to 1999 ...

A model of imperative programming

- Shared-memory concurrency
- Compositional for both parallelism and sequencing
- A few other things:
  - Features: pointers, release/acquire/SC access, fences, RMWs
  - Optimizations: reorderings, if-introduction, redundant read elimination, etc.
  - Java Causality Test Cases
  - DRF-SC, No-Thin-Air, temporal invariant reasoning
  - Efficient implementation on Arm8
  - Connection to C11, JavaScript, PTX, etc.
  - Automatic litmus test checker
  - Certified proofs

Revert to 1999 ...

- A model of imperative prog
  - Shared-memory concu
  - Compositional for botl
  - A few other things:
    - Features: pointer
    - Optimizations: re
    - Java Causality Te
    - DRF-SC, No-Thir
    - Efficient impleme
    - Connection to C1
    - Automatic litmus
    - Certified proofs



ces, RMWs

ant read elimination, etc.

g

Sequential Computation

$$x := 0;\ r := x;\ y := r + 1$$

Sequential Computation

$$x := 0;\ r := x;\ y := r + 1$$

Preconditions!

Sequential Computation: Preconditions + Hoare Logic!

$$x := 0;\ r := x;\ y := r + 1$$

$$y := r + 1 \quad \{y = 1\}$$

[Hoare 1969]

Sequential Computation: Preconditions + Hoare Logic!

$$x := 0;\ r := x;\ y := r + 1$$

$$\{r = 0\}\ \ y := r + 1\ \ \{y = 1\}$$

[Hoare 1969]

Sequential Computation: Preconditions + Hoare Logic!

$$x := 0;\ r := x;\ y := r + 1$$

$$\begin{array}{ccc} & r := x & \{r = 0\} \\ \{r = 0\} & y := r + 1 & \{y = 1\} \end{array}$$

[Hoare 1969]

Sequential Computation: Preconditions + Hoare Logic!

$$x := 0;\ r := x;\ y := r + 1$$

$$\begin{array}{lll} \{x = 0\} & r := x & \{r = 0\} \\ \{r = 0\} & y := r + 1 & \{y = 1\} \end{array}$$

[Hoare 1969]

Sequential Computation: Preconditions + Hoare Logic!

$$x := 0;\ r := x;\ y := r + 1$$

$$\begin{array}{lll} & x := 0 & \{x = 0\} \\ \{x = 0\} & r := x & \{r = 0\} \\ \{r = 0\} & y := r + 1 & \{y = 1\} \end{array}$$

[Hoare 1969]

Sequential Computation: Preconditions + Hoare Logic!

$$x := 0;\ r := x;\ y := r + 1$$

$$\begin{array}{rlr} \{\mathsf{tt}\} & x := 0 & \{x = 0\} \\ \{x = 0\} & r := x & \{r = 0\} \\ \{r = 0\} & y := r + 1 & \{y = 1\} \end{array}$$

[Hoare 1969]

Sequential Computation: Preconditions + Predicate Transformers!

$$x := 0;\ r := x;\ y := r + 1$$

$$\begin{aligned}
\mathsf{tt} &\Rightarrow wp(x := 0\ \ \ \ \ \ )(x = 0) \\
(x = 0) &\Rightarrow wp(r := x\ \ \ \ \ \ )(r = 0) \\
(r = 0) &\Rightarrow wp(y := r + 1)(y = 1)
\end{aligned}$$

[Hoare 1969] [Dijkstra 1975]

ntial Computation: Preconditions + Predicate Transformers!

$0;\ r := x;\ y := r$

$(x =$ $0)$

$(r = 0) \rightarrow$ $(y := r + 1)$ $)$

Concurrency?

e 1969] [Dijkstra 1975]

ntial Computation: Preconditions + Predicate Transformers!

$0;\ r := x;\ y := r$

$(r = 0)$ $(y := r + 1)$

Ordered Events!

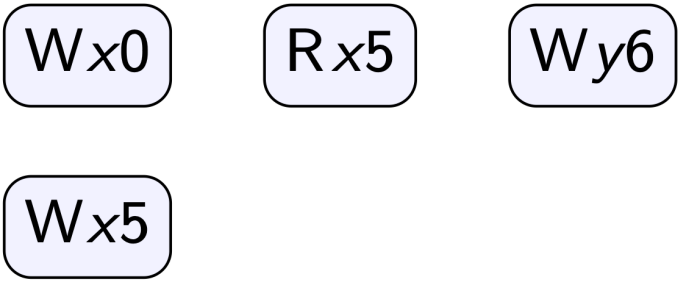
1969] [Dijkstra 1975]

Concurrent Computation

$$x := 0;\ r := x;\ y := r + 1 \ \parallel\ x := 5$$

Concurrent Computation: Pomsets! (aka labeled partial orders)

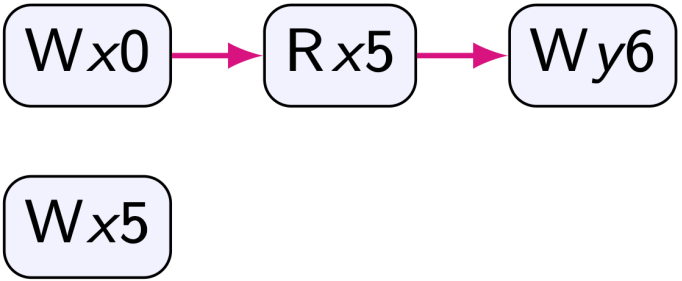
$$x := 0;\ r := x;\ y := r + 1 \ \|\ x := 5$$



[Gischer 1988] [Plotkin & Pratt 1990] [Brookes 1993]

Concurrent Computation: Pomsets! (aka labeled partial orders)

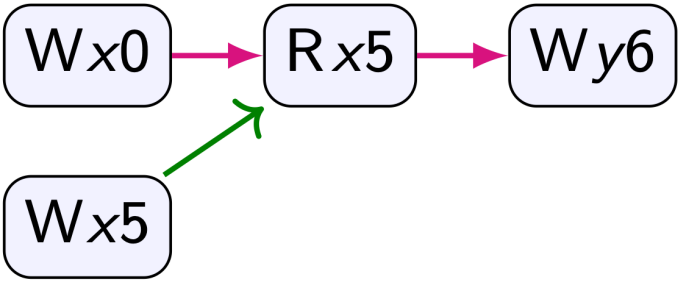
$$x := 0;\ r := x;\ y := r + 1 \ \|\ x := 5$$



[Gischer 1988] [Plotkin & Pratt 1990] [Brookes 1993]

Concurrent Computation: Pomsets! (aka labeled partial orders)

$$x := 0;\ r := x;\ y := r + 1 \ \|\ x := 5$$

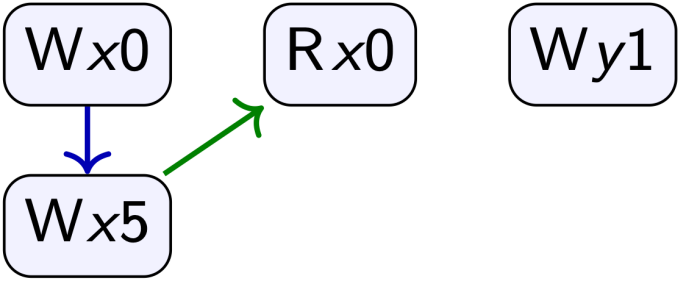


Rxv is *fulfilled* if:

$\exists \mathrm{W}xv < \mathrm{R}xv$

Concurrent Computation: Pomsets! (aka labeled partial orders)

$$x := 0;\ r := x;\ y := r + 1 \ \|\ x := 5$$



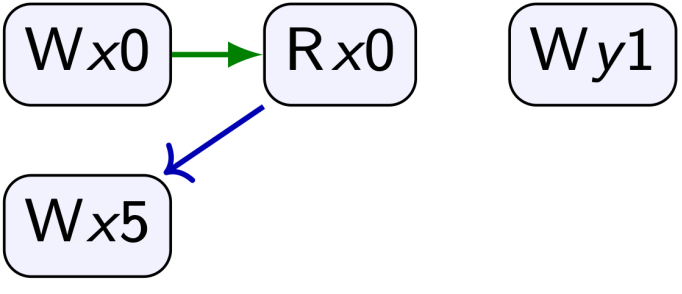
Rxv is *fulfilled* if:

$\exists \mathrm{W}xv < \mathrm{R}xv$

$\forall \mathrm{W}xu$ either $\mathrm{W}xu \leqslant \mathrm{W}xv$ or $\mathrm{R}xv < \mathrm{W}xu$

Concurrent Computation: Pomsets! (aka labeled partial orders)

$$x := 0;\ r := x;\ y := r + 1 \ \|\ x := 5$$



Rxv is *fulfilled* if:

$\exists \mathrm{W}xv < \mathrm{R}xv$

$\forall \mathrm{W}xu$ either $\mathrm{W}xu \leqslant \mathrm{W}xv$ or $\mathrm{R}xv < \mathrm{W}xu$

rrent Computation: Pomsets! (aka labeled partial orders)

$x := x; y := r + 1 \parallel$ 5

Hardware Reordering?

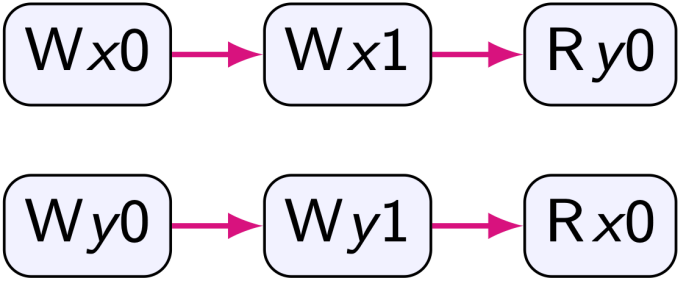
R if:

$\forall Rxv < Rxv$

$\forall Wxu$ either $Wxu \leqslant Wxv$ or $Rxv < Wxu$

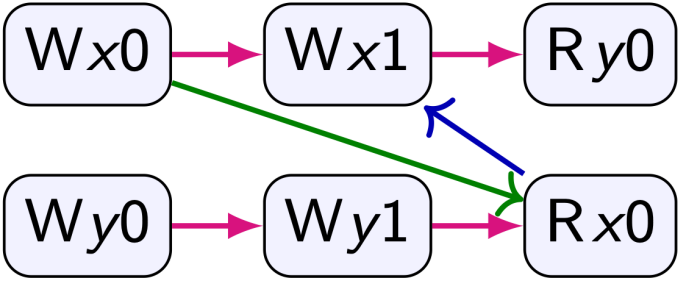
Hardware Reordering?

$$x := 0;\ x := 1;\ r := y \ \|\ y := 0;\ y := 1;\ s := x$$



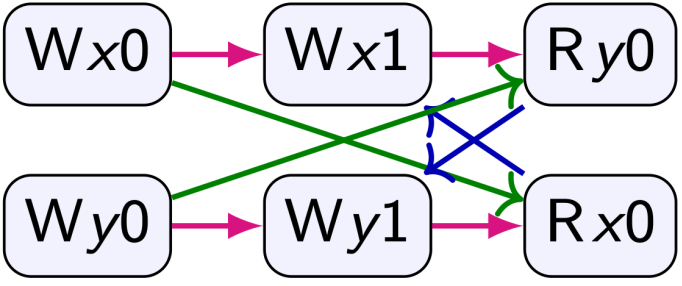
Hardware Reordering?

$$x := 0;\ x := 1;\ r := y \ \|\ y := 0;\ y := 1;\ s := x$$



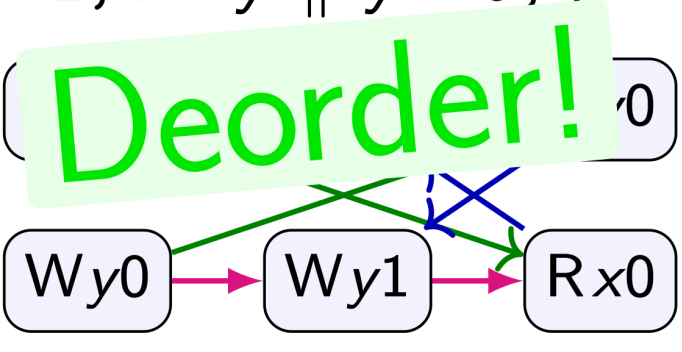
Hardware Reordering?

$$x := 0;\ x := 1;\ r := y \ \|\ y := 0;\ y := 1;\ s := x$$



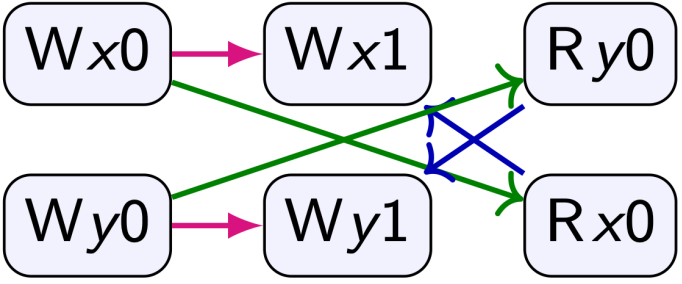
Hardware Reordering?

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



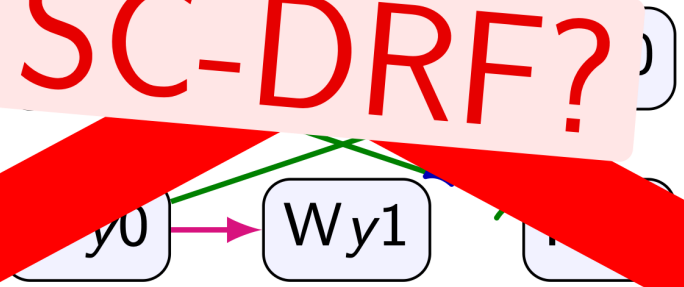
Deordering!

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

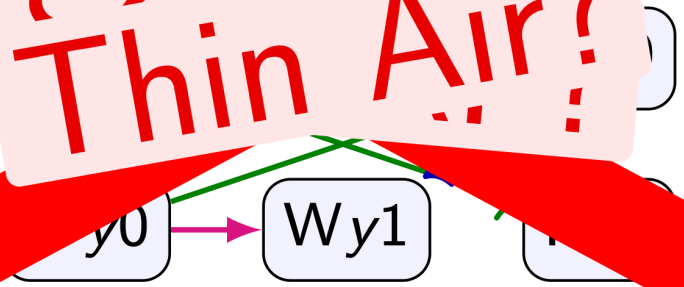


$x := 0; x := y \parallel y := 1; s := x$

SC-DRF?



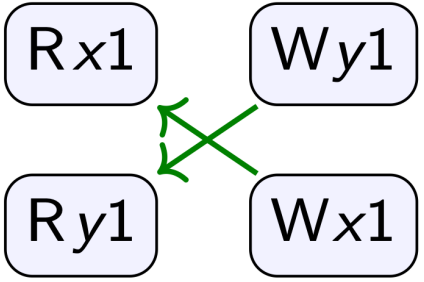
$x := 0; x :$ $:= y \parallel y :=$ $- 1; s := x$



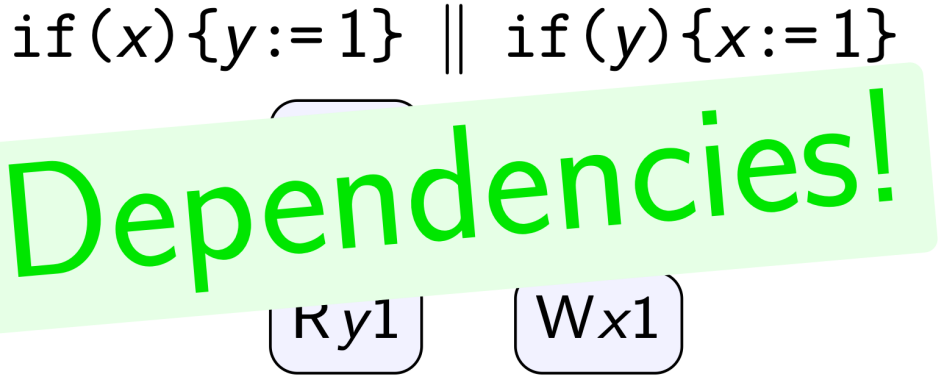
Thin Air?

SC-DRF? Thin Air?

$$\texttt{if}(x)\{y := 1\} \parallel \texttt{if}(y)\{x := 1\}$$



SC-DRF? Thin Air?



SC-DRF? Thin Air?

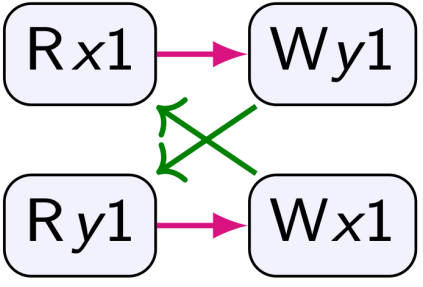
$\texttt{if}(x)\{y := 1\} \parallel \texttt{if}(y)\{x := 1\}$

Syntactic Dependencies!

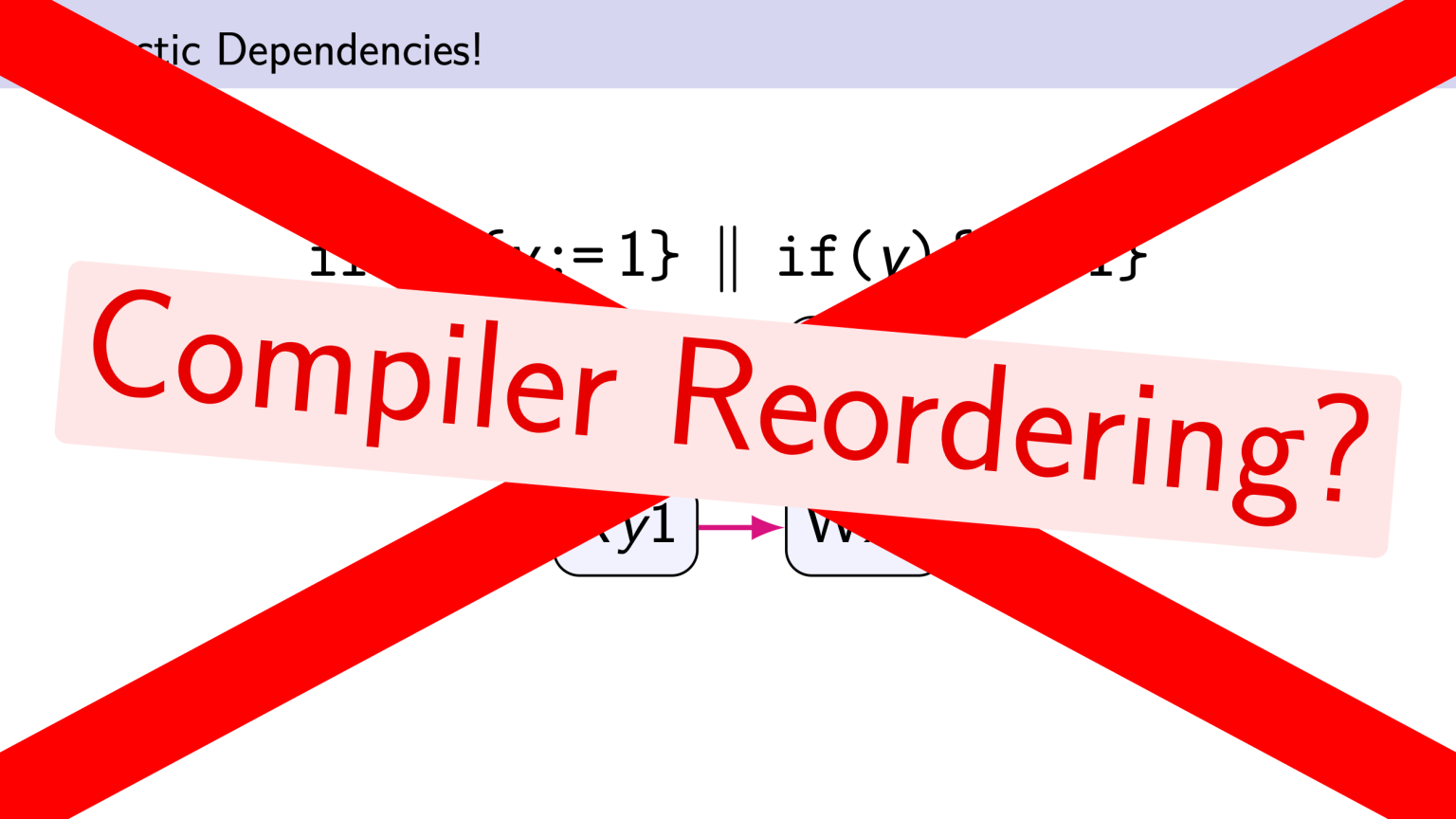
Ry1 Wx1

Syntactic Dependencies!

$$\texttt{if}(x)\{y := 1\} \parallel \texttt{if}(y)\{x := 1\}$$

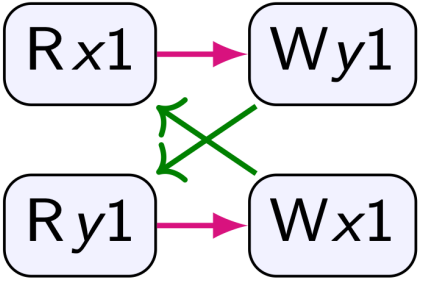


ctic Dependencies!



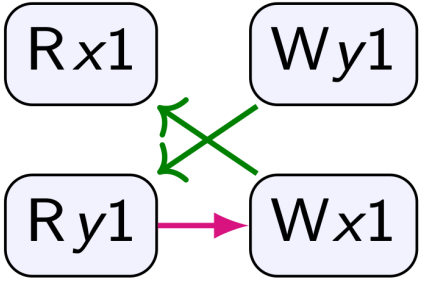
Compiler Reordering?

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



Compiler Reordering?

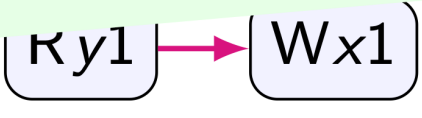
$$r := x;\ y := (r * 0) + 1 \ \parallel\ s := y;\ x := s$$



Compiler Reordering?

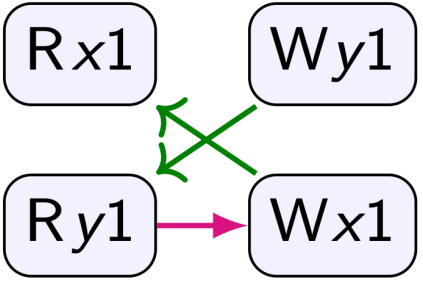
$$r := x;\ y := (r * 0) + 1 \ \parallel\ s := y;\ x := s$$

Semantic Dependencies!



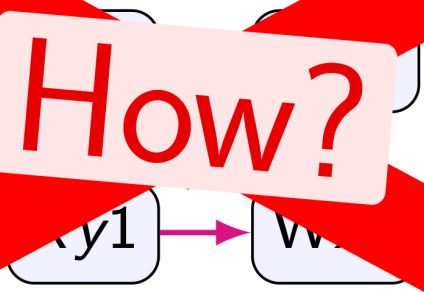
Semantic Dependencies!

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



tic Dependencies!

$r := x$ $(r * 0) + 1 \parallel s$ $:= s$



How?

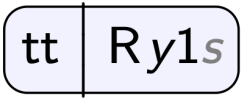
...tic Dependencies!

Pomsets

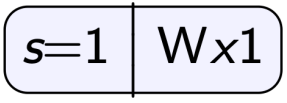
+ Preconditions!

Pomsets with Preconditions

$s := y$



$x := s$

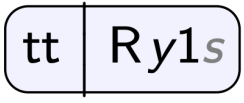


Pomsets with Preconditions

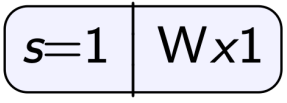
$s := y$; $x := s$

Pomsets with Preconditions and Predicate Transformers

$s := y$; $x := s$



D	$\tau^D(\psi)$
$\emptyset$	ψ
$\{\mathsf{R}y1s\}$	$(1{=}s) \Rightarrow \psi$



E	$\tau^E(\psi)$
$\emptyset$	ψ
$\{\mathsf{W}x1\}$	ψ

Pomsets with Preconditions and Predicate Transformers

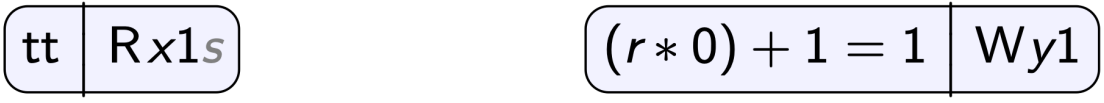
$s := y$; $x := s$

D	$\tau^D(\psi)$
$\emptyset$	ψ
$\{Ry1s\}$	$(1{=}s) \Rightarrow \psi$

E	$\tau^E(\psi)$
$\emptyset$	ψ
$\{Wx1\}$	ψ

Pomsets with Preconditions and Predicate Transformers

$r := x$; $y := (r * 0) + 1$

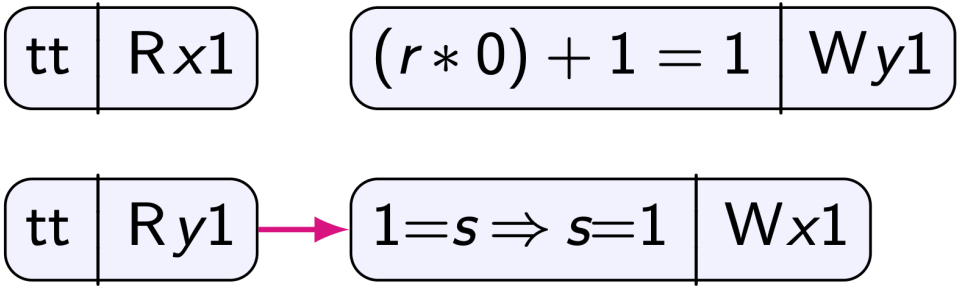


D	$\tau^D(\psi)$
$\emptyset$	ψ
$\{\mathsf{R}x1r\}$	$(1{=}r) \Rightarrow \psi$

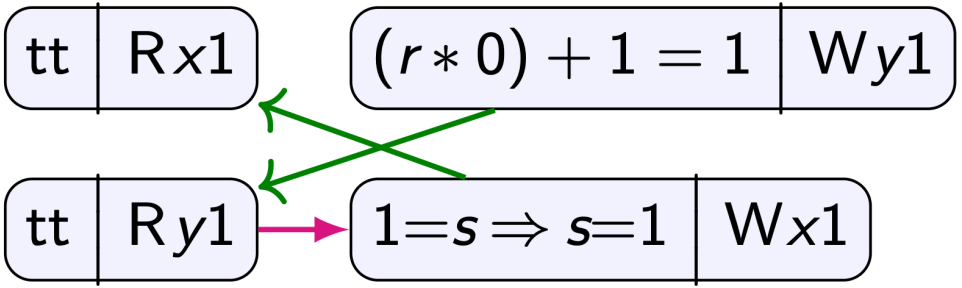
E	$\tau^E(\psi)$
$\emptyset$	ψ
$\{\mathsf{W}y1\}$	ψ

Pomsets with Preconditions and Predicate Transformers

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

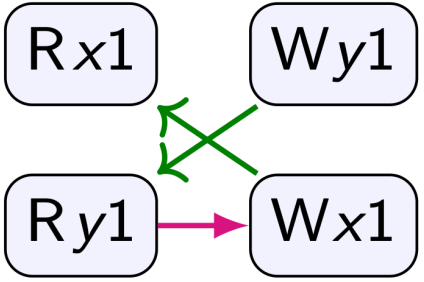


$$r := x;\ y := (r * 0) + 1 \ \parallel\ s := y;\ x := s$$



Pomsets with Preconditions and Predicate Transformers

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



Logic=Local order

Pomset=Other order

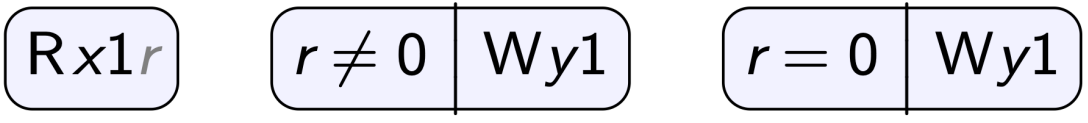
Local Order

Poms…Out

Control Independencies?

Control Independencies?

$r := x;\ \texttt{if}(r \neq 0)\{y := 1\};\ \texttt{if}(r = 0)\{y := 1\}$

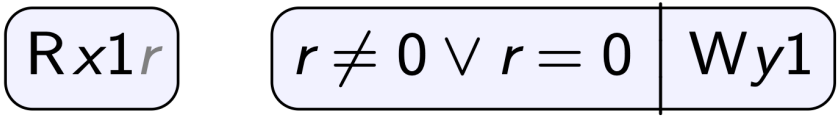


Control Independencies?

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

Merging!

$r := x;\ \texttt{if}(r \neq 0)\{y := 1\};\ \texttt{if}(r = 0)\{y := 1\}$



ng!

$r := x$; if( ){$y := 1$}; i(0){$y := 1$}

Associative?

Associative! (Left to Right)

$r := x$ $\quad$ $s := y$ $\quad$ $\texttt{if}\,(s)\,\{z := r{*}(s{-}1)\}$

$\mathsf{R}x1r$ $\quad$ $\mathsf{R}y1s$ $\quad$ $(s{\neq}0) \wedge (r{*}(s{-}1)){=}0 \mid \mathsf{W}z0$

C	$\tau^C(\psi)$
$\emptyset$	ψ
$\{\mathsf{R}x1r\}$	$(1{=}r) \Rightarrow \psi$

D	$\tau^D(\psi)$
$\emptyset$	ψ
$\{\mathsf{R}y1s\}$	$(1{=}s) \Rightarrow \psi$

E	$\tau^E(\psi)$
$\emptyset$	ψ
$\{\mathsf{W}y1\}$	ψ

Associative! (Left to Right)

$(r := x \quad ; \quad s := y)$ $\qquad$ $\texttt{if}\,(s)\,\{z := r * (s-1)\}$

Rx1r		Ry1s		$(s \neq 0) \wedge (r*(s-1))=0$	Wz0

C	$\tau^C(\psi)$
$\emptyset$	ψ
{Rx1r}	$(1=r) \Rightarrow \psi$

D	$\tau^D(\psi)$
$\emptyset$	ψ
{Ry1s}	$(1=s) \Rightarrow \psi$

E	$\tau^E(\psi)$
$\emptyset$	ψ
{Wy1}	ψ

Associative! (Left to Right)

$(r := x \quad ; \quad s := y)$

$\mathsf{R}x1r$ $\quad$ $\mathsf{R}y1s$

$\texttt{if}\,(s)\,\{z := r*(s-1)\}$

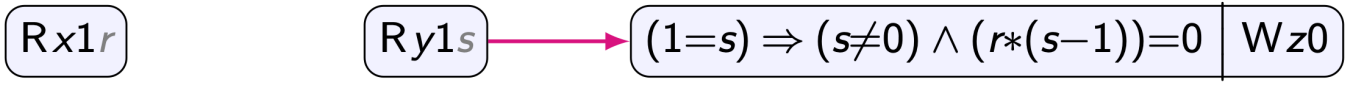
$(s \neq 0) \wedge (r*(s-1))=0$ | $\mathsf{W}z0$

D	$\tau^D(\psi)$
$\emptyset$	ψ
$\{\mathsf{R}x1r\}$	$(1=r) \Rightarrow \psi$
$\{\mathsf{R}y1s\}$	$(1=s) \Rightarrow \psi$
$\{\mathsf{R}x1r, \mathsf{R}y1s\}$	$(1=r) \Rightarrow (1=s) \Rightarrow \psi$

E	$\tau^E(\psi)$
$\emptyset$	ψ
$\{\mathsf{W}y1\}$	ψ

Associative! (Left to Right)

$$(r := x \quad ; \quad s := y) \quad ; \quad \texttt{if}\,(s)\,\{z := r * (s-1)\}$$

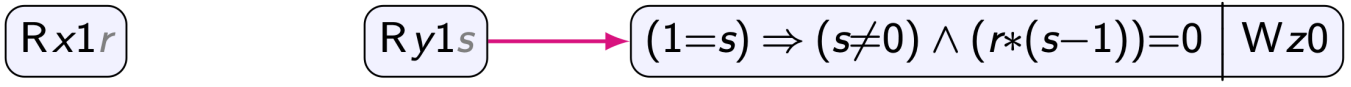


D	$\tau^D(\psi)$
$\emptyset$	ψ
$\{\mathrm{R}x1r\}$	$(1{=}r) \Rightarrow \psi$
$\{\mathrm{R}y1s\}$	$(1{=}s) \Rightarrow \psi$
$\{\mathrm{R}x1r, \mathrm{R}y1s\}$	$(1{=}r) \Rightarrow (1{=}s) \Rightarrow \psi$

E	$\tau^E(\psi)$
$\emptyset$	ψ
$\{\mathrm{W}y1\}$	ψ

Associative! (Left to Right)

$$(r := x \quad ; \quad s := y) \quad ; \quad \texttt{if}\,(s)\,\{z := r * (s-1)\}$$



D	$\tau^D(\psi)$
$\emptyset, \{\mathsf{W}y1\}$	ψ
$\{\mathsf{R}x1r\}, \{\mathsf{R}x1r, \mathsf{W}y1\}$	$(1=r) \Rightarrow \psi$
$\{\mathsf{R}y1s\}, \{\mathsf{R}y1s, \mathsf{W}y1\}$	$(1=s) \Rightarrow \psi$
$\{\mathsf{R}x1r, \mathsf{R}y1s\}, \{\mathsf{R}x1r, \mathsf{R}y1s, \mathsf{W}y1\}$	$(1=r) \Rightarrow (1=s) \Rightarrow \psi$

Associative! (Right to Left)

$r := x$ $\quad$ $s := y$ $\quad$ $\texttt{if}\,(s)\{z := r*(s-1)\}$

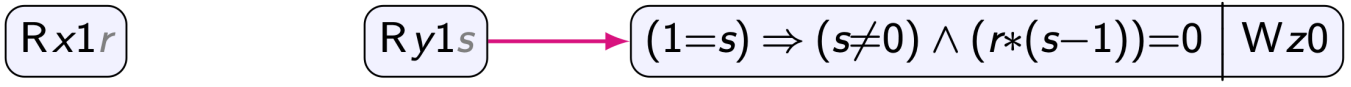
C	$\tau^C(\psi)$
$\emptyset$	ψ
$\{\mathrm{R}x1r\}$	$(1{=}r) \Rightarrow \psi$

D	$\tau^D(\psi)$
$\emptyset$	ψ
$\{\mathrm{R}y1s\}$	$(1{=}s) \Rightarrow \psi$

E	$\tau^E(\psi)$
$\emptyset$	ψ
$\{\mathrm{W}y1\}$	ψ

Associative! (Right to Left)

$r := x \quad (s := y \quad ; \quad \texttt{if}\,(s)\{z := r*(s-1)\})$



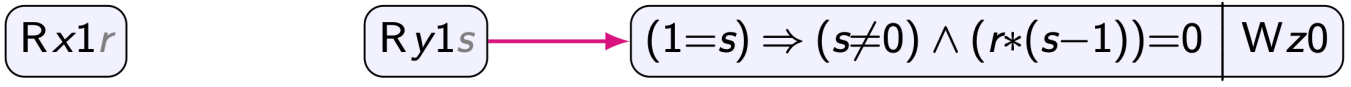
C	$\tau^C(\psi)$
$\emptyset$	ψ
$\{\mathrm{R}x1r\}$	$(1{=}r) \Rightarrow \psi$

D	$\tau^D(\psi)$
$\emptyset$	ψ
$\{\mathrm{R}y1s\}$	$(1{=}s) \Rightarrow \psi$

E	$\tau^E(\psi)$
$\emptyset$	ψ
$\{\mathrm{W}y1\}$	ψ

Associative! (Right to Left)

$$r := x \qquad (s := y \quad ; \quad \texttt{if}\,(s)\{z := r * (s-1)\})$$

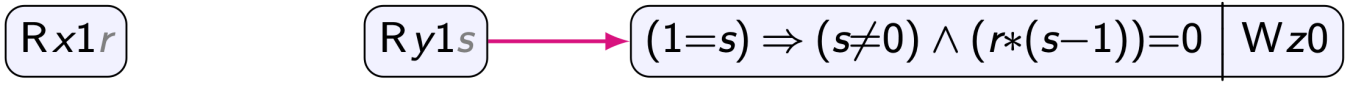


C	$\tau^C(\psi)$
$\emptyset$	ψ
$\{\mathrm{R}x1r\}$	$(1{=}r) \Rightarrow \psi$

D	$\tau^D(\psi)$
$\emptyset, \{\mathrm{W}y1\}$	ψ
$\{\mathrm{R}y1s\}, \{\mathrm{R}y1s, \mathrm{W}y1\}$	$(1{=}s) \Rightarrow \psi$

Associative! (Right to Left)

$$r := x \quad ; \quad (s := y \quad ; \quad \texttt{if}\,(s)\{z := r*(s-1)\})$$

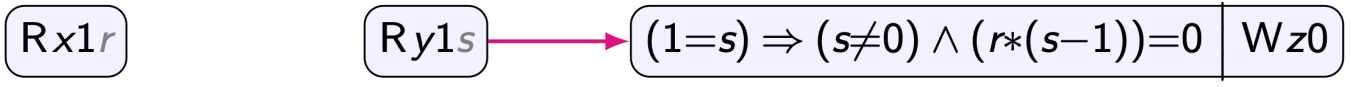


C	$\tau^C(\psi)$
$\emptyset$	ψ
$\{\mathsf{R}x1r\}$	$(1{=}r) \Rightarrow \psi$

D	$\tau^D(\psi)$
$\emptyset, \{\mathsf{W}y1\}$	ψ
$\{\mathsf{R}y1s\}, \{\mathsf{R}y1s, \mathsf{W}y1\}$	$(1{=}s) \Rightarrow \psi$

Associative! (Right to Left)

$$r := x \quad ; \quad (s := y \quad ; \quad \texttt{if}\,(s)\{z := r*(s-1)\})$$



D	$\tau^D(\psi)$
$\emptyset, \{\mathsf{W}y1\}$	ψ
$\{\mathsf{R}x1r\}, \{\mathsf{R}x1r, \mathsf{W}y1\}$	$(1{=}r) \Rightarrow \psi$
$\{\mathsf{R}y1s\}, \{\mathsf{R}y1s, \mathsf{W}y1\}$	$(1{=}s) \Rightarrow \psi$
$\{\mathsf{R}x1r, \mathsf{R}y1s\}, \{\mathsf{R}x1r, \mathsf{R}y1s, \mathsf{W}y1\}$	$(1{=}r) \Rightarrow (1{=}s) \Rightarrow \psi$

Associative! (Right to Left)

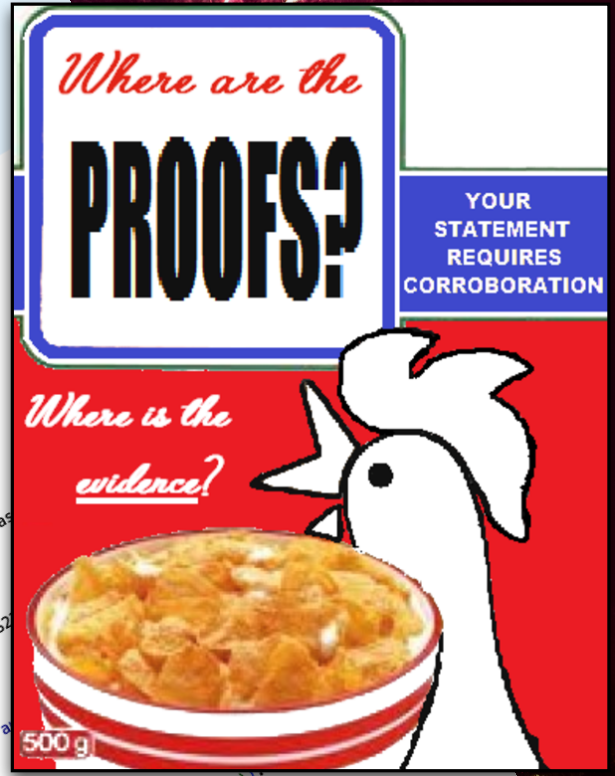
$r := x \quad ; \quad (s := y \quad ; \quad \texttt{if}(s)\{z := x(\ldots 1)\})$

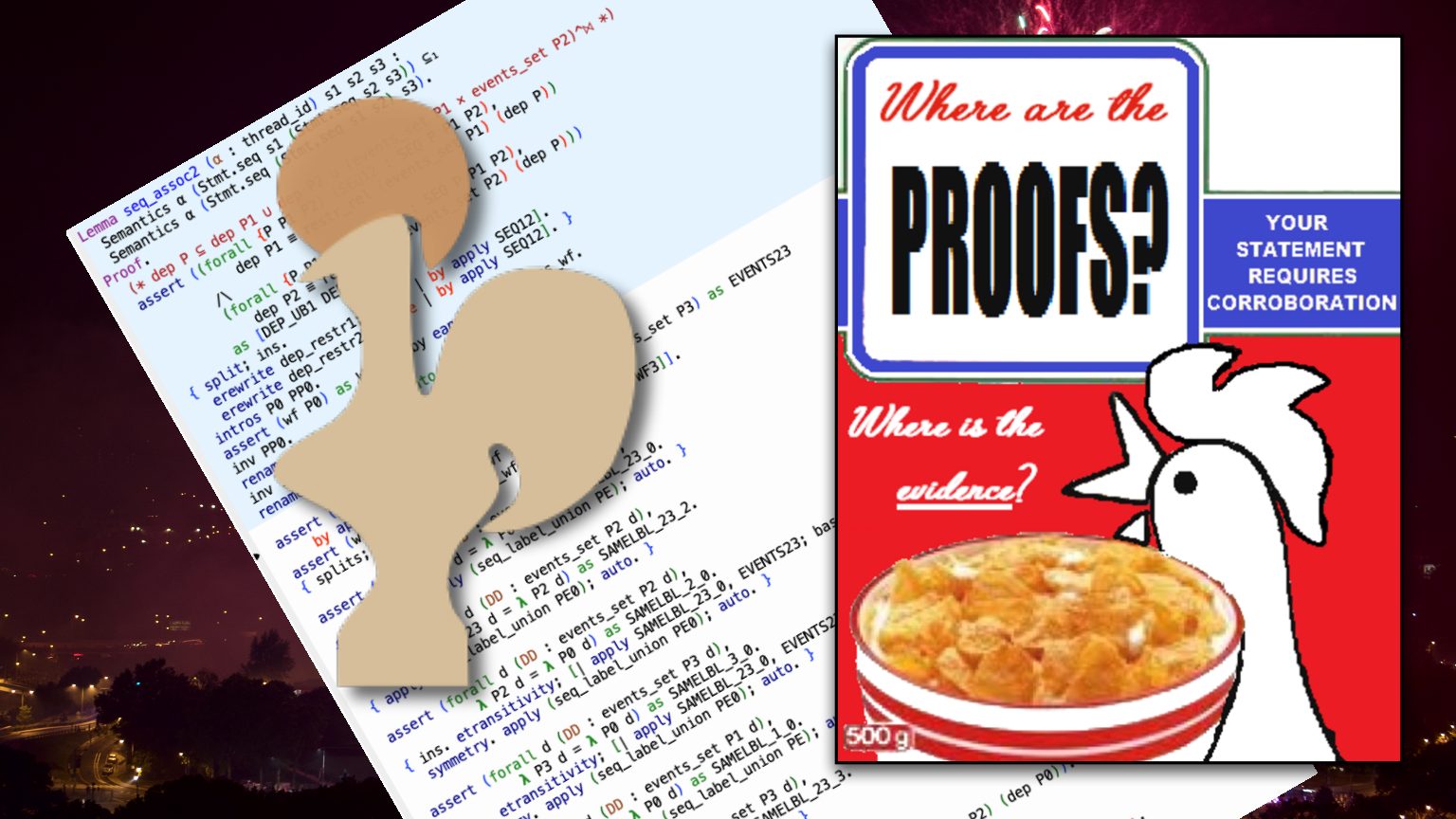
R… Wz0

They're the same!

	$\tau^D(\psi)$
$\emptyset, \{\mathsf{W}y1\}$	ψ
$\{\mathsf{R}x1r\}, \{\mathsf{R}x1r, \mathsf{W}y1\}$	$(1{=}r) \Rightarrow \psi$
$\{\mathsf{R}y1s\}, \{\mathsf{R}y1s, \mathsf{W}y1\}$	$(1{=}s) \Rightarrow \psi$
$\{\mathsf{R}x1r, \mathsf{R}y1s\}, \{\mathsf{R}x1r, \mathsf{R}y1s, \mathsf{W}y1\}$	$(1{=}r) \Rightarrow (1{=}s) \Rightarrow \psi$


Where are the
PROOFS?
YOUR STATEMENT REQUIRES CORROBORATION
Where is the evidence?
500 g


Where are the
PROOFS?
YOUR STATEMENT REQUIRES CORROBORATION
Where is the evidence?
500 g


Where are the
PROOFS?
YOUR STATEMENT REQUIRES CORROBORATION
Where is the evidence?
500 g


Where are the
YOUR STATEMENT REQUIRES CORROBORATION
500 g
Real Hardware?

Logic=Local order
Pomset=Other order


Logic=Local order
Pomset=Other order
MCA!

Multicopy Atomic Architectures

$$S_1 ; S_2$$

Let order include (in addition to dependency):

- Coherence ($Wx \rightarrow Wx$)
- Release/Acquire Access/Fences ($Wx \rightarrow W^{rel}y$, $R^{acq}x \rightarrow Wy$)
- SC Access/Fences ($W^{sc}x \rightarrow R^{sc}y$)

Multicopy Atomic Architectures

$$S_1 ; S_2$$

Let order include (in addition to dependency):

- Coherence ($Wx \rightarrow Wx$)
- Release/Acquire Access/Fences ($Wx \rightarrow W^{rel}y$, $R^{acq}x \rightarrow Wy$)
- SC Access/Fences ($W^{sc}x \rightarrow R^{sc}y$)

Arm8 optimal for: Relaxed✓ Release✓ Acquiring✗

Multicopy Atomic Architectures

$$S_1 ; S_2$$

Let order include (in addition to dependency):

- Coherence ($Wx \rightarrow Wx$)
- Release/Acquire Access/Fences ($Wx \rightarrow W^{rel}y$, $R^{acq}x \rightarrow Wy$)
- SC Access/Fences ($W^{sc}x \rightarrow R^{sc}y$)

Arm8 optimal for: Relaxed✓ Release✓ Acquiring✗

Arm8 some work: Relaxed✓ Release✓ Acquiring✓

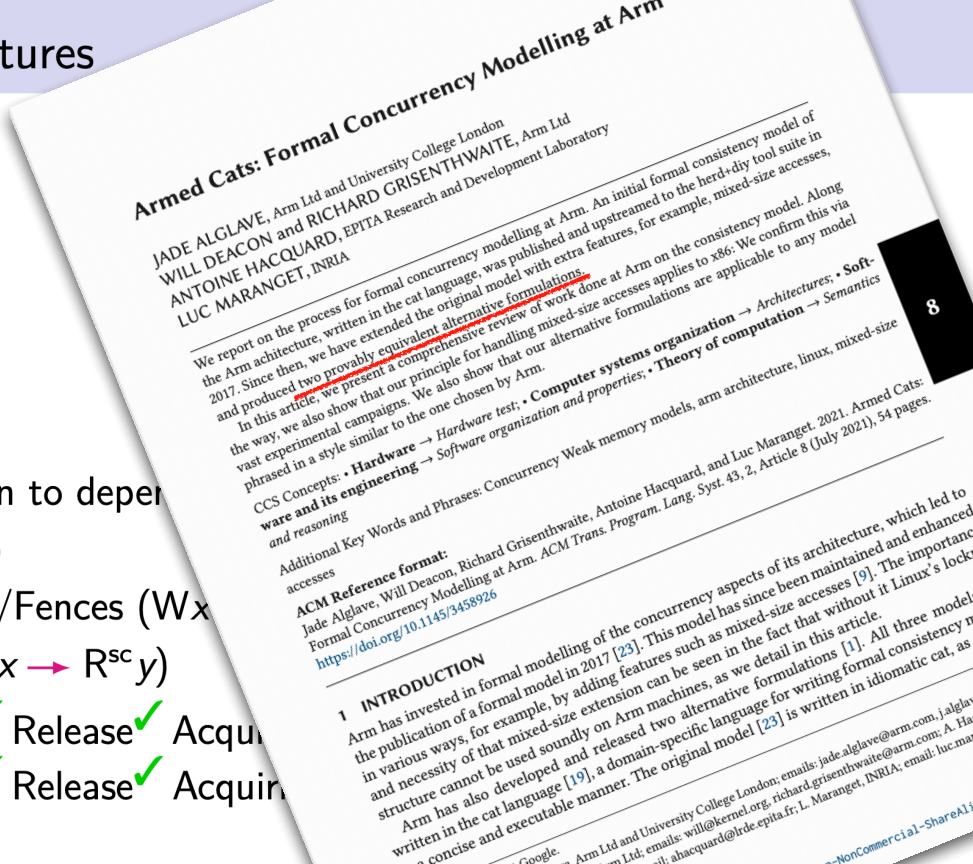
Multicopy Atomic Architectures

Let order include (in addition to depe

- Coherence ($Wx \rightarrow Wx$)
- Release/Acquire Access/Fences (Wx
- SC Access/Fences ($W^{sc}x \rightarrow R^{sc}y$)

Arm8 optimal for: Relaxed✓ Release✓ Acqu

Arm8 some work: Relaxed✓ Release✓ Acquir



Armed Cats: Formal Concurrency Modelling at Arm

JADE ALGLAVE, Arm Ltd and University College London
WILL DEACON and RICHARD GRISENTHWAITE, Arm Ltd
ANTOINE HACQUARD, EPITA Research and Development Laboratory
LUC MARANGET, INRIA

We report on the process for formal concurrency modelling at Arm. An initial formal consistency model of the Arm achitecture, written in the cat language, was published and upstreamed to the herd+diy tool suite in 2017. Since then, we have extended the original model with extra features, for example, mixed-size accesses, and produced two provably equivalent alternative formulations.
In this article, we present a comprehensive review of work done at Arm on the consistency model. Along the way, we also show that our principle for handling mixed-size accesses applies to x86: We confirm this via vast experimental campaigns. We also show that our alternative formulations are applicable to any model phrased in a style similar to the one chosen by Arm.

CCS Concepts: • **Hardware** → *Hardware test*; • **Computer systems organization** → *Architectures*; • **Software and its engineering** → *Software organization and properties*; • **Theory of computation** → *Semantics and reasoning*

Additional Key Words and Phrases: Concurrency Weak memory models, arm architecture, linux, mixed-size accesses

ACM Reference format:
Jade Alglave, Will Deacon, Richard Grisenthwaite, Antoine Hacquard, and Luc Maranget. 2021. Armed Cats: Formal Concurrency Modelling at Arm. *ACM Trans. Program. Lang. Syst.* 43, 2, Article 8 (July 2021), 54 pages.
https://doi.org/10.1145/3458926

8

1 INTRODUCTION

Arm has invested in formal modelling of the concurrency aspects of its architecture, which led to the publication of a formal model in 2017 [23]. This model has since been maintained and enhanced in various ways, for example, by adding features such as mixed-size accesses [9]. The importance and necessity of that mixed-size extension can be seen in the fact that without it Linux's lock structure cannot be used soundly on Arm machines, as we detail in this article.
Arm has also developed and released two alternative formulations [1]. All three model written in the cat language [19], a domain-specific language for writing formal consistency m concise and executable manner. The original model [23] is written in idiomatic cat, as

Google.
Arm Ltd and University College London; emails: jade.alglave@arm.com, j.alglav
Ltd; emails: will@kernel.org, richard.grisenthwaite@arm.com; A. Ha
ahacquard@lrde.epita.fr; L. Maranget, INRIA; email: luc.mar

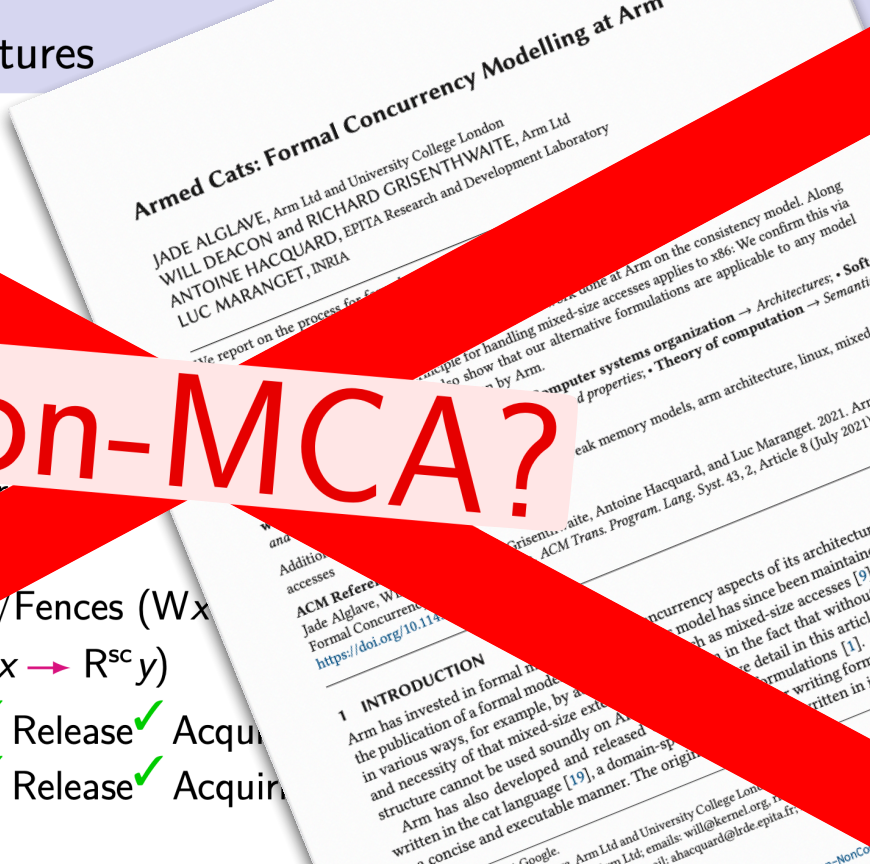
...copy Atomic Architectures

Let order include (in addition

- Coherence ($Wx \rightarrow$
- Release/Ac... .../Fences (Wx
- SC A... ...ces ($W^{sc}x \rightarrow R^{sc}y$)

A... ...al for: Relaxed✓ Release✓ Acqui...

...some work: Relaxed✓ Release✓ Acquir...



Non-MCA?

...copy Atomic Architectures

Let order inc...

Coherenc...

Release/A...

SC A... ces ($W^{sc}x \rightarrow R^{sc}y$)

A... for: Relaxed✓ Release✓ Acqu...

...some work: Relaxed✓ Release✓ Acquir...

Armed Cats: Formal Concurrency Modelling at Arm

...Arm Ltd and University College London
...CHARD GRISENTHWAITE, Arm Ltd
...arch and Development Laboratory

...on the consistency model. Along
...applies to x86: We confirm this via
...ations are applicable to any model

...rganization → *Architectures*; • **Soft-**
...heory of computation → *Semantics*

...dels, arm architecture, linux, mixed-size

...uard, and Luc Maranget. 2021. Armed Cats:
...ng. Syst. 43, 2, Article 8 (July 2021), 54 pages.

8

...ncurrency aspects of its architecture, which led to
...model has since been maintained and enhanced
...h as mixed-size accesses [9]. The importanc
...in the fact that without it Linux's lock
...e detail in this article.
...rmulations [1]. All three model
...writing formal consistency m
...ritten in idiomatic cat, as

...@arm.com, j.alglav
...arm.com; A. Ha
...ail: luc.mar

Formal Co... https://doi.org/10.11...

1 INTRODUCTION

Arm has invested in formal m...
the publication of a formal mode...
in various ways, for example, by a...
and necessity of that mixed-size exte...
structure cannot be used soundly on A...
Arm has also developed and released ...
written in the cat language [19], a domain-sp...
concise and executable manner. The origi...

...Google.
...Arm Ltd and University College Lond...
...Ltd; emails: will@kernel.org, r...
...ail: ahacquard@lrde.epita.fr...

Dependencies for C11!

Dependencies for RC11

Replace No-TAR axiom

$$sb \cup rf \text{ is acyclic}$$

Dependencies for RC11

Replace No-TAR axiom

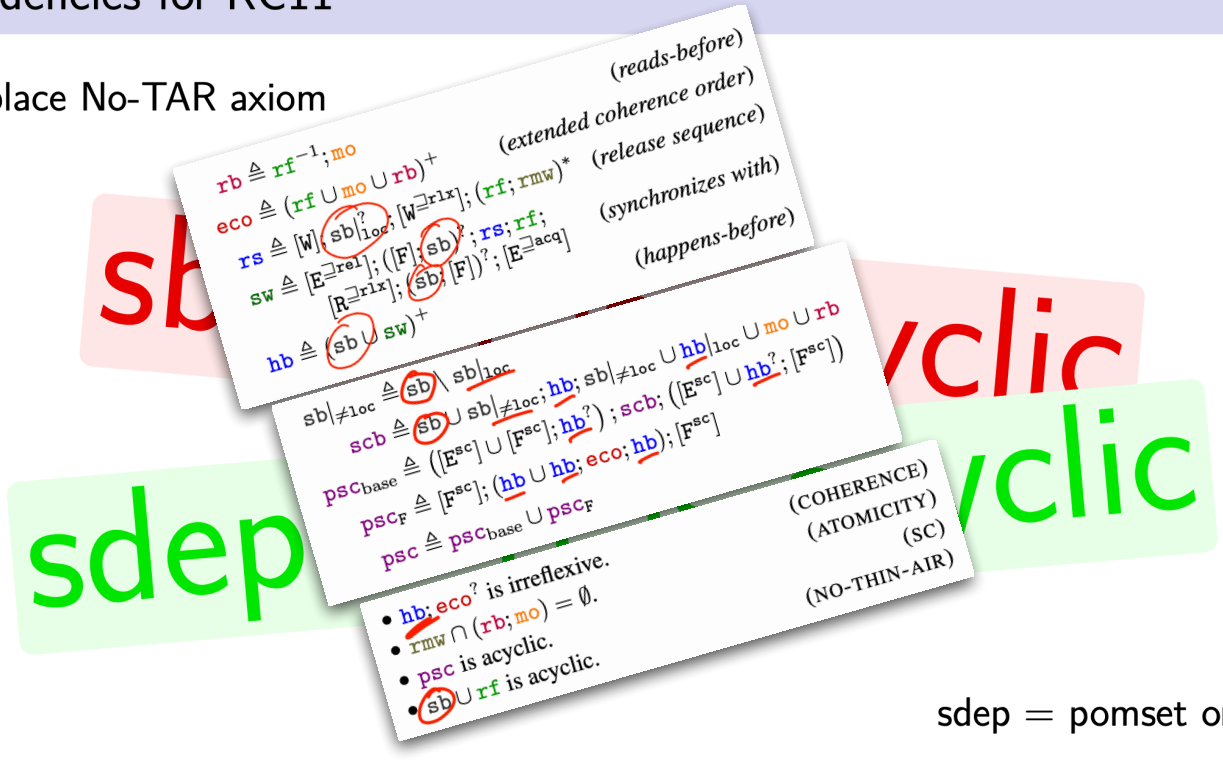
$$\mathsf{sb} \cup \mathsf{rf} \text{ is acyclic}$$

$$\mathsf{sdep} \cup \mathsf{rf} \text{ is acyclic}$$

sdep = pomset order

Dependencies for RC11

Replace No-TAR axiom



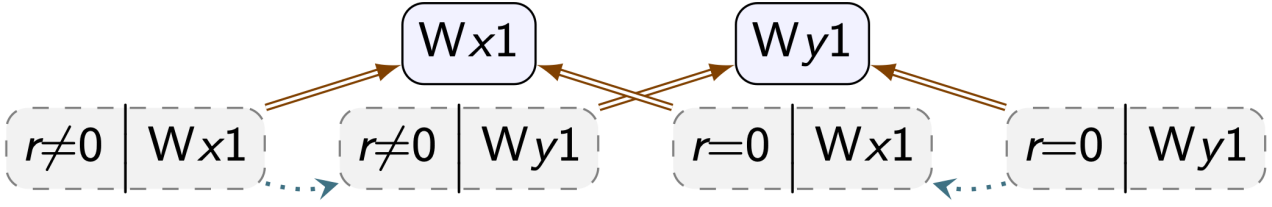
sdep = pomset order

Merging and Synchronized-Before?

$$\texttt{if}\,(r)\,\{x := 1;\ y := 1\}\,\texttt{else}\,\{y := 1;\ x := 1\}$$

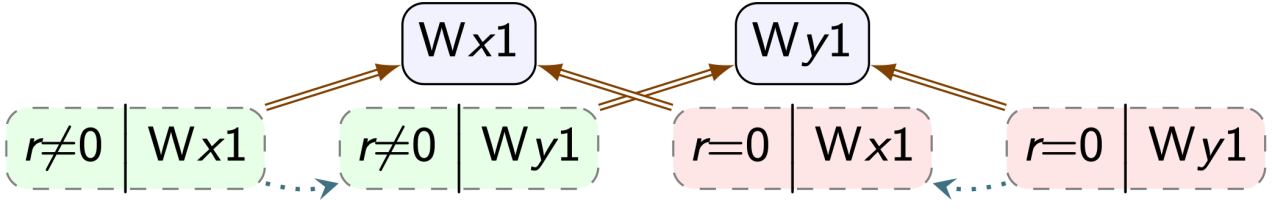
Merging and Synchronized-Before?

$$\texttt{if}\,(r)\,\{x := 1;\ y := 1\}\,\texttt{else}\,\{y := 1;\ x := 1\}$$



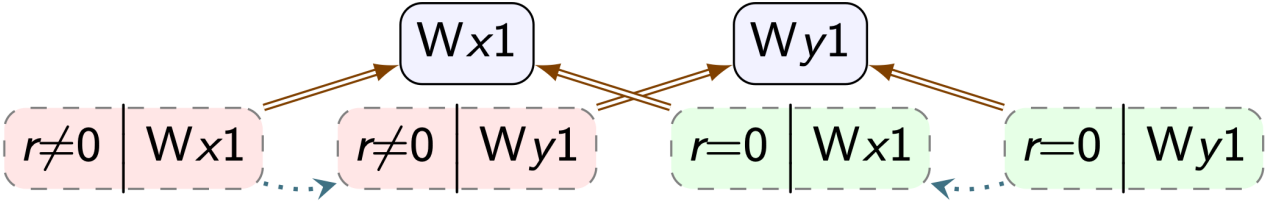
Merging and Synchronized-Before?

$\texttt{if}\,(r)\,\{x := 1;\ y := 1\}\,\texttt{else}\,\{y := 1;\ x := 1\}$

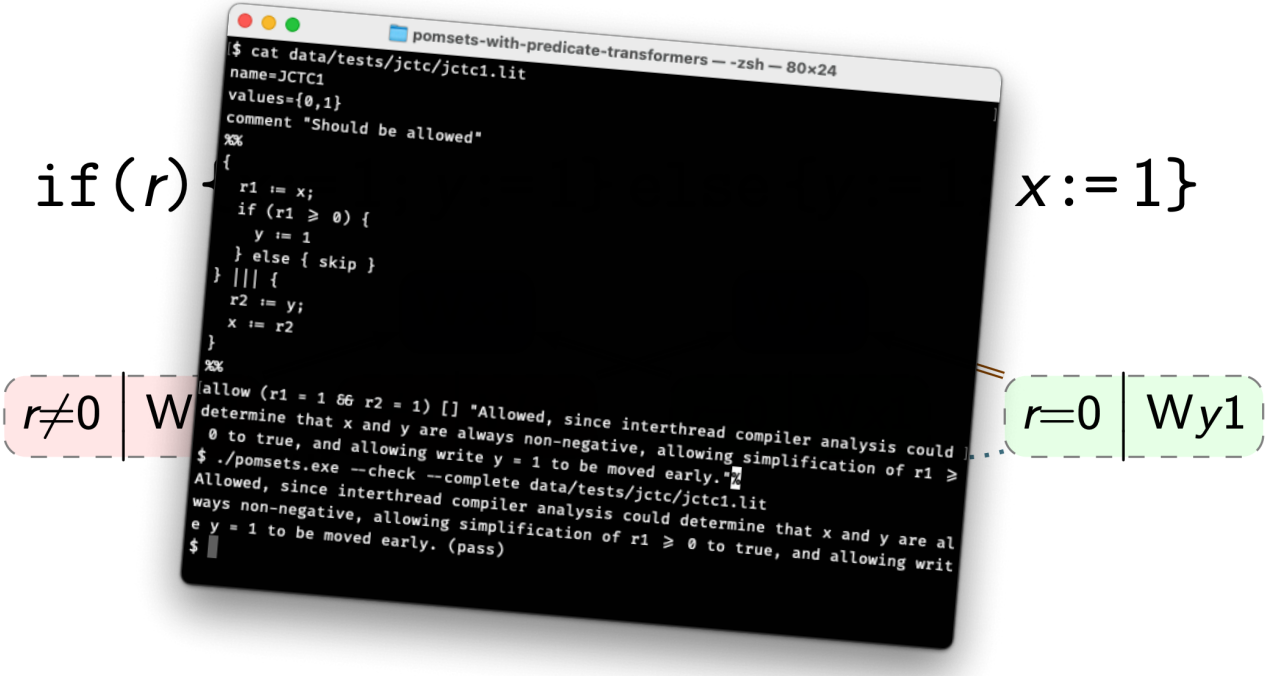


Merging and Synchronized-Before?

$$\texttt{if}\,(r)\,\{x := 1;\ y := 1\}\,\texttt{else}\,\{y := 1;\ x := 1\}$$



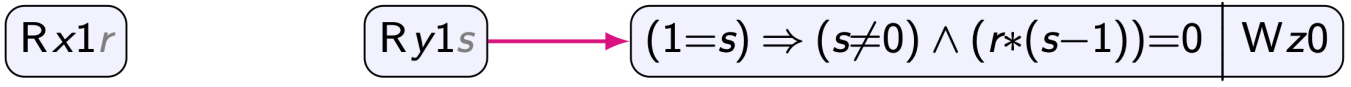
Merging and Synchronized-Before?



Merging and Synchronized-Before?

Associative! (Right to Left)

$$r := x \quad ; \quad (s := y \quad ; \quad \texttt{if}\,(s)\{z := r*(s-1)\})$$



D	$\tau^D(\psi)$
$\emptyset, \{\mathsf{W}y1\}$	ψ
$\{\mathsf{R}x1r\}, \{\mathsf{R}x1r, \mathsf{W}y1\}$	$(1{=}r) \Rightarrow \psi$
$\{\mathsf{R}y1s\}, \{\mathsf{R}y1s, \mathsf{W}y1\}$	$(1{=}s) \Rightarrow \psi$
$\{\mathsf{R}x1r, \mathsf{R}y1s\}, \{\mathsf{R}x1r, \mathsf{R}y1s, \mathsf{W}y1\}$	$(1{=}r) \Rightarrow (1{=}s) \Rightarrow \psi$

Sequential composition for an optimized concurrent language

Dependency = Pomset order = Order seen by other threads

Sequential composition for an optimized concurrent language

Dependency = Pomset order = Order seen by other threads

Very compositional for MCA

- Solution for Out-Of-Thin-Air models (C, JavaScript)

Sequential composition for an optimized concurrent language

- Dependency = Pomset order = Order seen by other threads
- Very compositional for MCA
  - Solution for Out-Of-Thin-Air models (C, JavaScript)
- Denotational semantics
  - Tractable notion of refinement for peephole optimization
  - Understandable by human beings

Sequential composition for an optimized concurrent language

- Dependency = Pomset order = Order seen by other threads
- Very compositional for MCA
  - Solution for Out-Of-Thin-Air models (C, JavaScript)
- Denotational semantics
  - Tractable notion of refinement for peephole optimization
  - Understandable by human beings
- Main complexity
  - Calculating dependencies
  - Merging and unmerging
  - In conditionals, single execution uses both sides

Sequential composition for an optimized concurrent language

- Dependency = Pomset order = Order seen by other threads
- Very compositional for MCA
  - Solution for Out-Of-Thin-Air models (C, JavaScript)
- Denotational semantics
  - Tractable notion of refinement for peephole optimization
  - Understandable by human beings
- Main complexity
  - Calculating dependencies
  - Merging and unmerging
  - In conditionals, single execution uses both sides
- More in the paper
  - RMWs, address calculation, Indirect dependencies, If-introduction, etc

Sequential composition for an optimized concurrent language

Dependency = Pomset order = Order seen by other threads

Concurrency is easy!

Tractable notion of ...

Understandable by human beings

Main complexity

- Calculating dependencies
- Merging and unmerging
- In conditionals, single execution uses both sides

More in the paper

- RMWs, address calculation, Indirect dependencies, If-introduction, etc

Sequential composition for an optimized concurrent language

Dependency = Pomset order = Order seen by other threads

Concurrency is easy!

Tractable notion of

Understandable by human

Sequentiality is hard!

tionals, single execution uses both sides

More in the paper

RMWs, address calculation, Indirect dependencies, If-introduction, etc


Concurrency is easy!
Sequentiality is hard!


Concurrency is easy!
Sequentiality is hard!