# A DENOTATIONAL APPROACH TO RELEASE/ACQUIRE CONCURRENCY

ESOP, ETAPS 2024

**Yotam Dvir**, Ohad Kammar, Ori Lahav

PARC HOTEL (LUXEMBOURG)

# GOAL

**Release/Acquire**

**For weak, shared-memory model**

**Using Brookes-style** [1996]**, totally-ordered traces**

**Design a standard, monad-based denotational semantics à la Moggi** [1991]

# WHY RELEASE/ACQUIRE?

**RA is an important fragment of C11, enables decentralized architectures (POWER)**

**First adaptation of *Brookes's traces* to a relaxed-memory software model**

**Intricate causal semantics, not overwhelmingly detailed**

**acyclic** $(\mathtt{po} \cup \mathtt{rf})^+|_{\mathrm{loc}} \cup \mathtt{mo} \cup \mathtt{rb}$

**Threads can disagree about the order of writes**
**(non-multi-copy-atomic)**

**Supports flag-based synchronization**
**(e.g. for signaling a data structure is ready)**

**Supports important transformations**
**(e.g. thread sequencing, write-read-reorder)**

**Supports read-modify-write atomicity**
**(e.g. atomic compare-and-swap)**

# WHY MONAD-BASED?

**Standard**

**Program effects** added **modularly**

The **core language** remains exactly **the same**

**Higher-order** programming **built-in**

**Rich toolkit** of definitions, theorems, and techniques

**Structural transformations**

$$\text{if } K_{\text{pure}} \text{ then } M; P_1 \text{ else } M; P_2 \cong M; \text{if } K_{\text{pure}} \text{ then } P_1 \text{ else } P_2$$

**Logical relations**

"related inputs go to related outputs"

**Substitution lemma**

syntax substitution ~ semantic context

etc
etc
etc

# DENOTATIONAL SEMANTICS

$$[\![ - ]\!] : \mathrm{Term} \to \mathrm{Deno}$$

compose from subterms' denotations

**For example:**

Monadic bind

$$[\![ \mathbf{let}\, x = M_1 \,\mathbf{in}\, M_2 ]\!] \triangleq [\![ M_1 ]\!] \boxed{\gg\!=} \lambda x\,.\, [\![ M_2 ]\!]$$

$$[\![ M_1 \parallel M_2 ]\!] \triangleq [\![ M_1 ]\!] \boxed{|\!|\!|} [\![ M_2 ]\!]$$

A modular effect extension

# ADEQUACY

**Abstraction:**
We want this to hold as much as possible

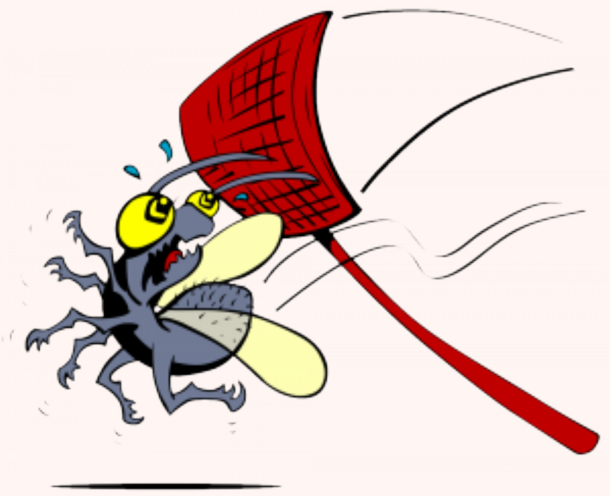
$$[\![ - ]\!] : \mathrm{Term} \to \mathrm{Deno}$$

$$[\![ M ]\!] \geq [\![ K ]\!] \implies M \twoheadrightarrow K$$

$K$ **denotationally refines** $M$

$K$ **contextually refines** $M$

**safe to replace within any context**



# ADEQUACY

## With non-determinism as sets

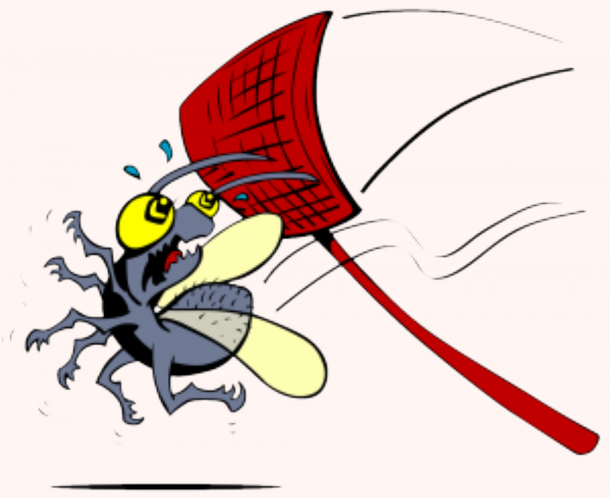
**Abstraction:**
We want this to hold as much as possible

$$\text{Deno} = \mathscr{P}(\text{Behavior})$$

$$[\![ M ]\!] \supseteq [\![ K ]\!] \implies M \twoheadrightarrow K$$

**Every possible behavior of $K$ is a possible behavior of $M$**

**$K$ contextually refines $M$**
**safe to replace within any context**



# GOAL

**RELEASE/ACQUIRE**

**For weak, shared-memory model**

**Using Brookes-style** [1996]**, totally-ordered traces**

**Design a standard, monad-based denotational semantics à la Moggi** [1991]

# TRACE-BASED SEMANTICS

Brookes [1996]

**Main ingredient:**

- linearly-ordered traces
- of local state-transitions
- that sequence and interleave

No interference    Possible interference

$$\langle\mu_1,\rho_1\rangle\langle\mu_2,\rho_2\rangle\ldots\langle\mu_{n-1},\rho_{n-1}\rangle\langle\mu_n,\rho_n\rangle$$

$\langle\mu_1,\mu_1'\rangle\ \langle\mu_2,\mu_2'\rangle\ \ldots\ \langle\mu_n,\mu_n'\rangle$    $\langle\rho_1,\rho_1'\rangle\ \langle\rho_2,\rho_2'\rangle\ \ldots\ \langle\rho_n,\rho_n'\rangle$

# TRACE-BASED SEMANTICS

Brookes [1996]

**Main ingredient:**

- linearly-ordered traces
- of local state-transitions
- that sequence and interleave

No interference

Possible interference

$$\langle \mu_1, \rho_1 \rangle \langle \mu_2, \rho_2 \rangle \ldots \langle \mu_{n-1}, \rho_{n-1} \rangle \langle \mu_n, \rho_n \rangle$$

$$\langle \mu_1, \mu_1' \rangle \ \langle \mu_2, \mu_2' \rangle \ \ldots \ \langle \mu_n, \mu_n' \rangle \ \langle \rho_1, \rho_1' \rangle \ \langle \rho_2, \rho_2' \rangle \ \ldots \ \langle \rho_n, \rho_n' \rangle$$

**SEQUENCE**

# TRACE-BASED SEMANTICS

Brookes [1996]

**Main ingredient:**

- linearly-ordered traces
- of local state-transitions
- that sequence and interleave

No interference Possible interference

$$\langle \mu_1, \rho_1 \rangle \langle \mu_2, \rho_2 \rangle \ldots \langle \mu_{n-1}, \rho_{n-1} \rangle \langle \mu_n, \rho_n \rangle$$

$$\langle \rho_1, \rho_1' \rangle \; \langle \mu_1, \mu_1' \rangle \; \langle \mu_2, \mu_2' \rangle \; \langle \rho_2, \rho_2' \rangle \; \ldots \; \langle \mu_n, \mu_n' \rangle \; \langle \rho_n, \rho_n' \rangle$$

**INTERLEAVE**

# CONTRIBUTION

- **Standard denotational semantics**
- **Adequate for Release/Acquire**
- **Abstract enough to verify every known RA-valid transformation in the literature (but no full-abstraction theorem)**
- **Subtlety: Rely/Guarantee interpretation of traces (our traces do not correspond directly to interrupted executions)**

$$\langle \mu_1, \rho_1 \rangle \langle \mu_2, \rho_2 \rangle \ldots \langle \mu_{n-1}, \rho_{n-1} \rangle \langle \mu_n, \rho_n \rangle$$

# RELEASE/ACQUIRE

# INTUITION VIA LITMUS TESTS

**Store Buffering**

$$x := 0; y := 0;$$

$$\begin{array}{l|l} x := 1; & y := 1; \\ y? & x? \end{array}$$

**Message Passing**

$$x := 0; y := 0;$$

$$\begin{array}{l|l} x := 1; & y?; \\ y := 1 & x? \end{array}$$

# INTUITION VIA LITMUS TESTS

Propagation is not instant

**Store Buffering**

$$x := 0; y := 0;$$

$$\begin{array}{l|l} x := 1; & y := 1; \\ y? \quad /\!/0 & x? \quad /\!/0 \end{array}$$

✓

**Message Passing**

$$x := 0; y := 0;$$

$$\begin{array}{l|l} x := 1; & y?; \\ y := 1 & x? \end{array}$$

# INTUITION VIA LITMUS TESTS

## Store Buffering

Propagation is not instant

$$x := 0; y := 0;$$

$$\begin{array}{l||l} x := 1; & y := 1; \\ y? \quad /\!/0 & x? \quad /\!/0 \end{array}$$

## Message Pas

Propagation respects causality

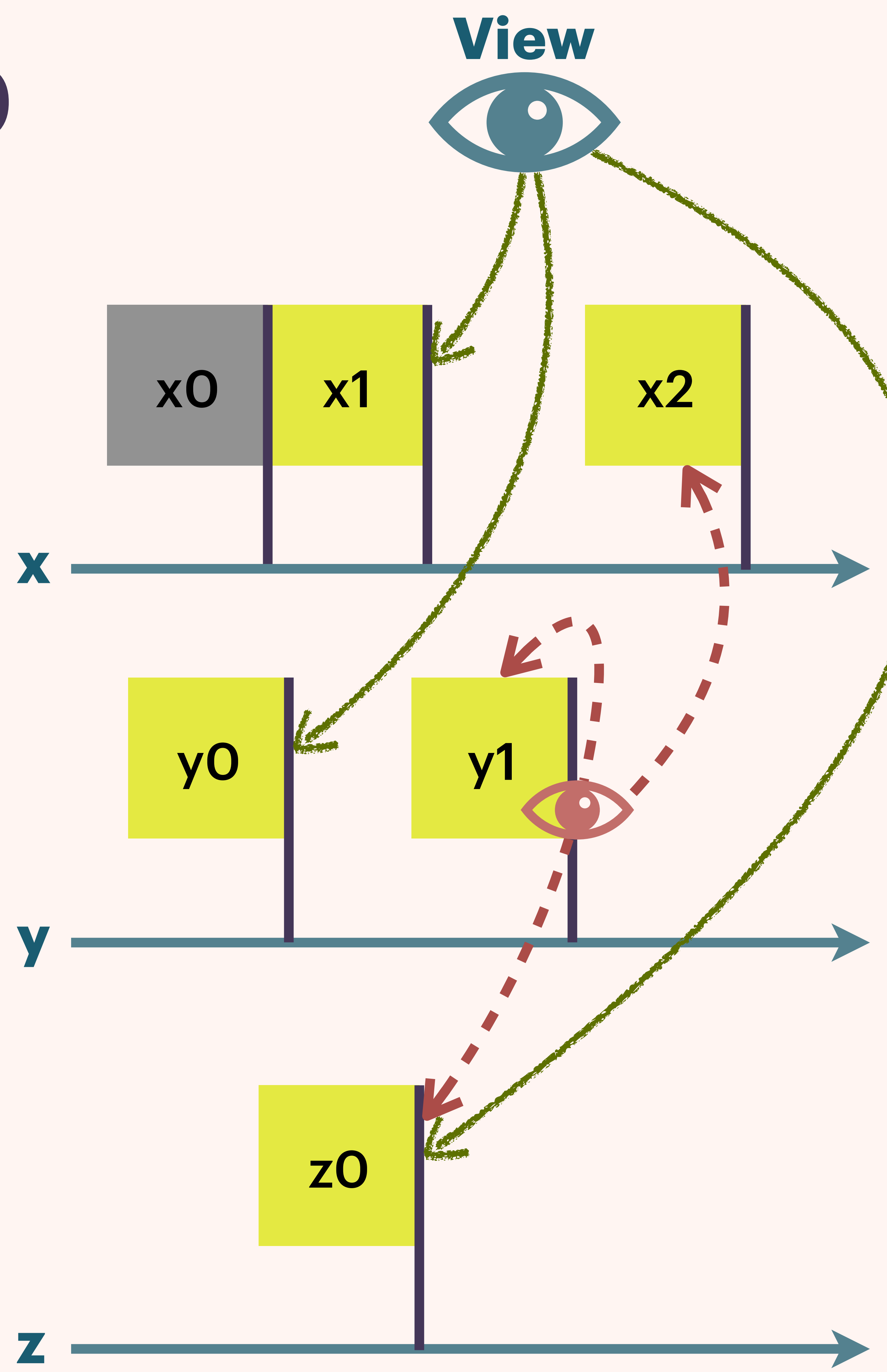
$$x := 0; y := 0;$$

$$\begin{array}{l||l} x := 1; & y?; \quad /\!/1 \\ y := 1 & x? \quad /\!/0 \end{array}$$

# RELEASE/ACQUIRE VIEW-BASED OPERATIONAL SEMANTICS

Kang et al. [2017]

- **Memory: Timeline per location**
- **Populated with immutable messages holding values**
- **Each view points to msgs on each timeline**
- **Threads have views — cannot read from "the past"**
- **Msgs have views for enforcing causal propagation**



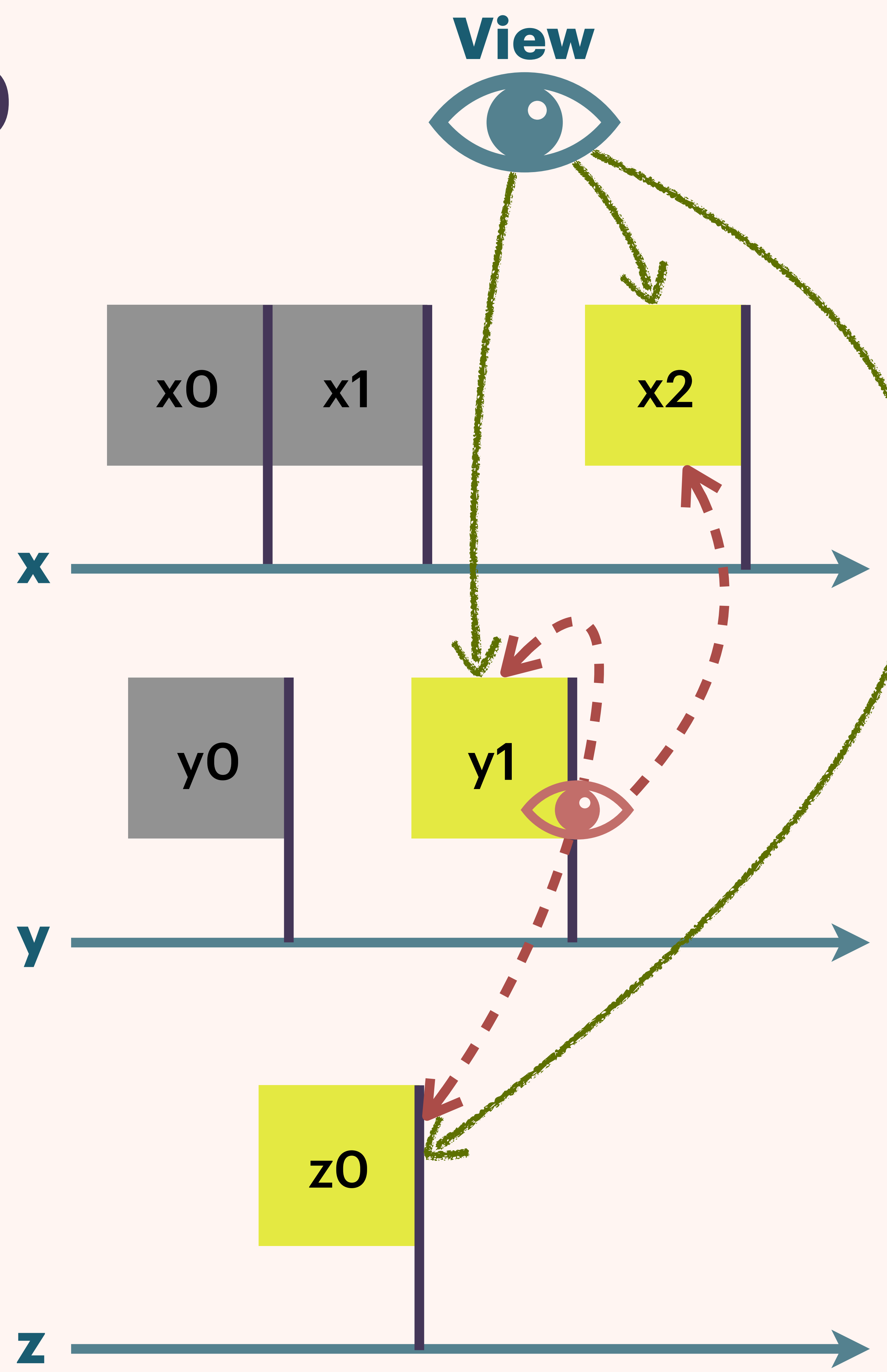

Propagation is not instant

Propagation respects causality

# RELEASE/ACQUIRE VIEW-BASED OPERATIONAL SEMANTICS

Kang et al. [2017]

- **Memory:** **Timeline per location**
- **Populated with immutable messages holding values**
- **Each view points to msgs on each timeline**
- **Threads have views — cannot read from "the past"**
- **Msgs have views for enforcing causal propagation**




Propagation is not instant

Propagation respects causality

# SUPPORTING FIRST-CLASS PARALLELISM

In the operational semantics

**Traditional op-sem: static view-array**

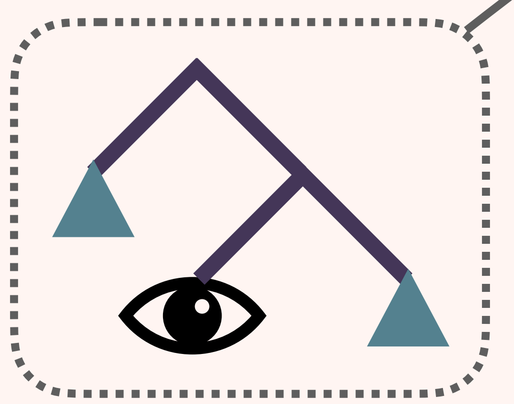
**Laws of Parallel Programming, e.g. Left Neutrality**

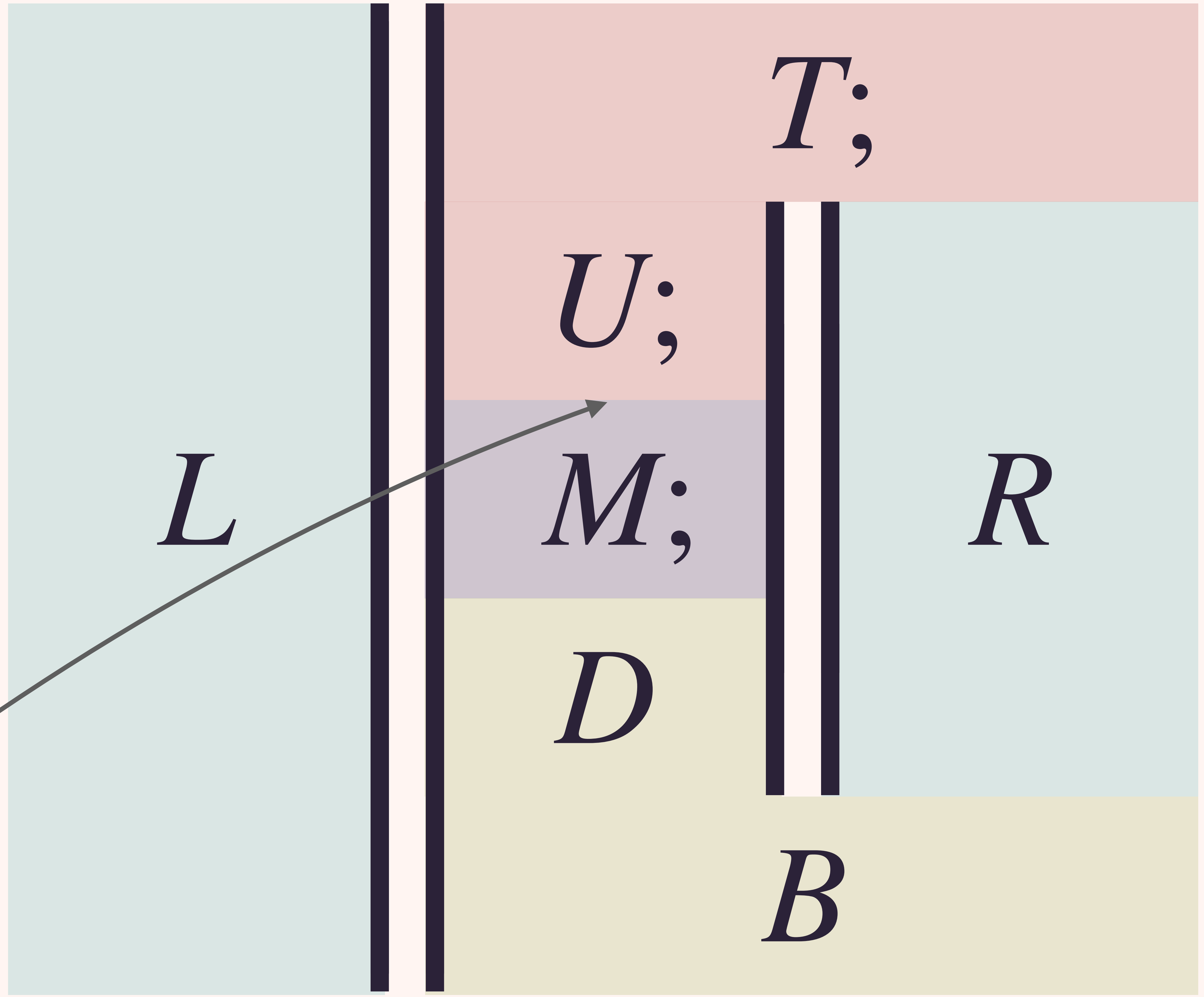
$$[\![ M ]\!] = [\![ (\langle\rangle \parallel M).\mathrm{snd} ]\!]$$

**Write-Read Deorder (Crucial RA refinement)**

$$[\![ x := 1; y? ]\!] \supseteq [\![ (x := 1 \parallel y?).\mathrm{snd} ]\!]$$

**Extended op-sem: dynamic view-tree**



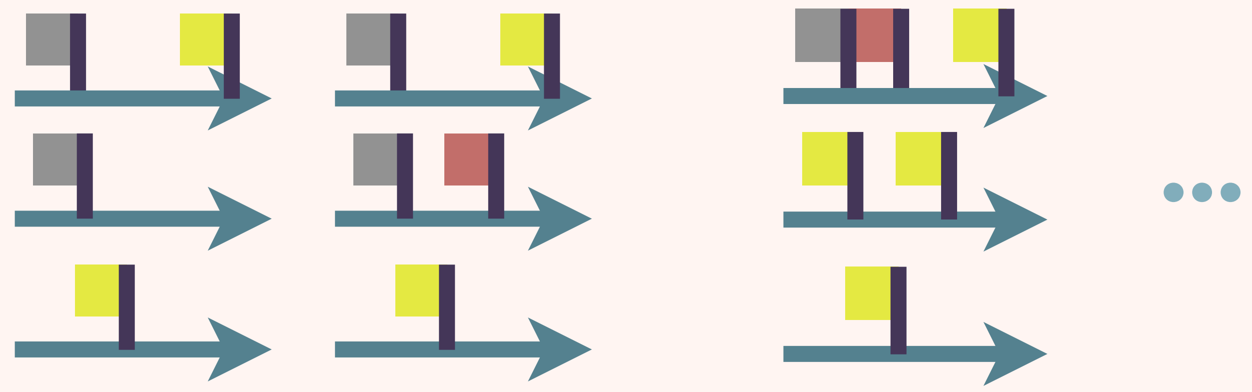


# RELEASE/ACQUIRE TRACES

# TRACE-BASED SEMANTICS IN RA

Initial View

Final View



$$\alpha \underbrace{\langle \mu_1, \rho_1 \rangle \langle \mu_2, \rho_2 \rangle \ldots \langle \mu_{n-1}, \rho_{n-1} \rangle \langle \mu_n, \rho_n \rangle}_{\text{Sequence of Transitions}} \omega \therefore r$$

Returns

# TRACE-BASED SEMANTICS IN RA

**Rely On $\mu_1$**

**To Guarantee $\rho_1$**

**Then**

**Rely On $\mu_2$**

**To Guarantee $\rho_2$**

**Then...**

**Guarantee to the sequential environment to return $r$**

Before or ||  ||  ||  After

$$\underbrace{\alpha}_{\text{Initial View}}\ \underbrace{\langle \mu_1, \rho_1\rangle \langle \mu_2, \rho_2\rangle \ldots \langle \mu_{n-1}, \rho_{n-1}\rangle \langle \mu_n, \rho_n\rangle}_{\text{Sequence of Transitions}}\ \underbrace{\omega}_{\text{Final View}} \therefore \underbrace{r}_{\text{Returns}}$$

# TRACE-BASED SEMANTICS IN RA

**Rely on** the **sequential environment** to **reveal messages**

**Guarantee to** the **sequential environment** to **reveal messages**

Before

After

**Avoid including whole state in transitions**

$$\underbrace{\alpha}_{\text{Initial View}}\ \underbrace{\langle \mu_1, \rho_1 \rangle \langle \mu_2, \rho_2 \rangle \ldots \langle \mu_{n-1}, \rho_{n-1} \rangle \langle \mu_n, \rho_n \rangle}_{\text{Sequence of Transitions}}\ \underbrace{\omega}_{\text{Final View}} \therefore \underbrace{r}_{\text{Returns}}$$

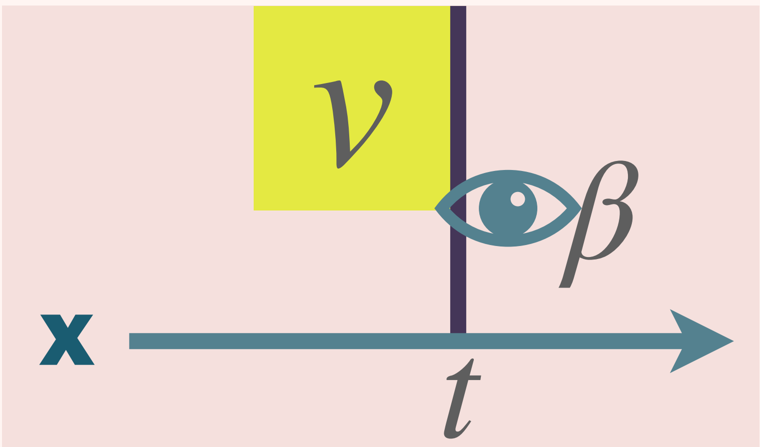
# RA DENOTATIONS

$$[\![ - ]\!] : \mathrm{Term} \to \mathrm{Deno}$$

# MEMORY ACCESS

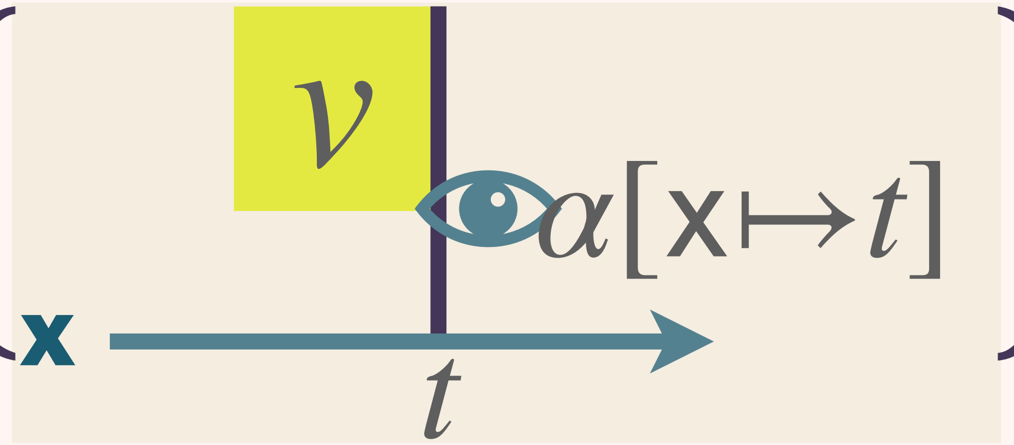
## Read

$$\frac{\alpha(x) \leq t \qquad \text{[timeline]} \in \mu}{\alpha\ \boxed{\langle \mu, \mu \rangle}\ \alpha \sqcup \beta \ \therefore\ v \in [\![\, x? \,]\!]}$$




## Write

$$\frac{\alpha(x) < t \qquad \rho = \mu \uplus \{\text{[timeline]}\}}{\alpha\ \boxed{\langle \mu, \rho \rangle}\ \alpha[\mathsf{x} \mapsto t] \ \therefore\ \langle\rangle \in [\![\, x := v \,]\!]}$$




## RMW

**Read the extended paper**

٩(^‿^)۶

# COMPOSITION

**Sequential**

$$\succcurlyeq \quad \frac{\alpha\,\boxed{\xi_1}\,\kappa \therefore r_1 \in [\![ M_1 ]\!] \qquad \kappa\,\boxed{\xi_2}\,\omega \therefore r_2 \in [\![ M_2 ]\!][x \mapsto r_1]}{\alpha\,\boxed{\xi_1\xi_2}\,\omega \therefore r_2 \in [\![ \mathbf{let}\, x = M_1 \,\mathbf{in}\, M_2 ]\!]}$$

SEQUENCING TRANSITIONS

**Parallel**

INTERLEAVING TRANSITIONS

$$||| \quad \frac{\forall i \in \{1,2\}\,.\ \alpha\,\boxed{\xi_i}\,\omega \therefore r_i \in [\![ M_i ]\!] \qquad \xi \in \xi_1 \,||\, \xi_2}{\alpha\,\boxed{\xi}\,\omega \therefore \langle r_1, r_2 \rangle \in [\![ M_1 \,||\, M_2 ]\!]}$$

# REWRITE CLOSURE RULES

- **Close denotations under rewrite rules**

**x-Rewrite Closure**

$$\frac{\pi \therefore r \in [\![ M ]\!]}{\tau \therefore r \in [\![ M ]\!]}\ \pi \xrightarrow{\mathsf{x}} \tau$$

- **Never introduced externally observable behavior**

Brookes

$$\alpha\,\boxed{\xi\eta}\,\omega \xrightarrow{\textbf{stutter}} \alpha\,\boxed{\xi\langle\mu,\mu\rangle\eta}\,\omega$$

Propagate Reliance as a Guarantee

$$\alpha\,\boxed{\xi\langle\mu,\rho\rangle\langle\rho,\theta\rangle\eta}\,\omega \xrightarrow{\textbf{mumble}} \alpha\,\boxed{\xi\langle\mu,\theta\rangle\eta}\,\omega$$

Rely on an omitted Guarantee

# REWRITE CLOSURE RULES

- **Close denotations under rewrite rules**

**x-Rewrite Closure**

$$\frac{\pi \therefore r \in [\![ M ]\!]}{\tau \therefore r \in [\![ M ]\!]} \; \pi \xrightarrow{x} \tau$$

- **Never introduced externally observable behavior**

$\alpha' \leq \alpha$ $\qquad \alpha\,\boxed{\xi}\,\omega \xrightarrow{\text{rewind}} \alpha'\,\boxed{\xi}\,\omega$

RA

Relying on more being revealed

$\omega \leq \omega'$ $\qquad \alpha\,\boxed{\xi}\,\omega \xrightarrow{\text{forward}} \alpha\,\boxed{\xi}\,\omega'$

Guaranteeing less being revealed

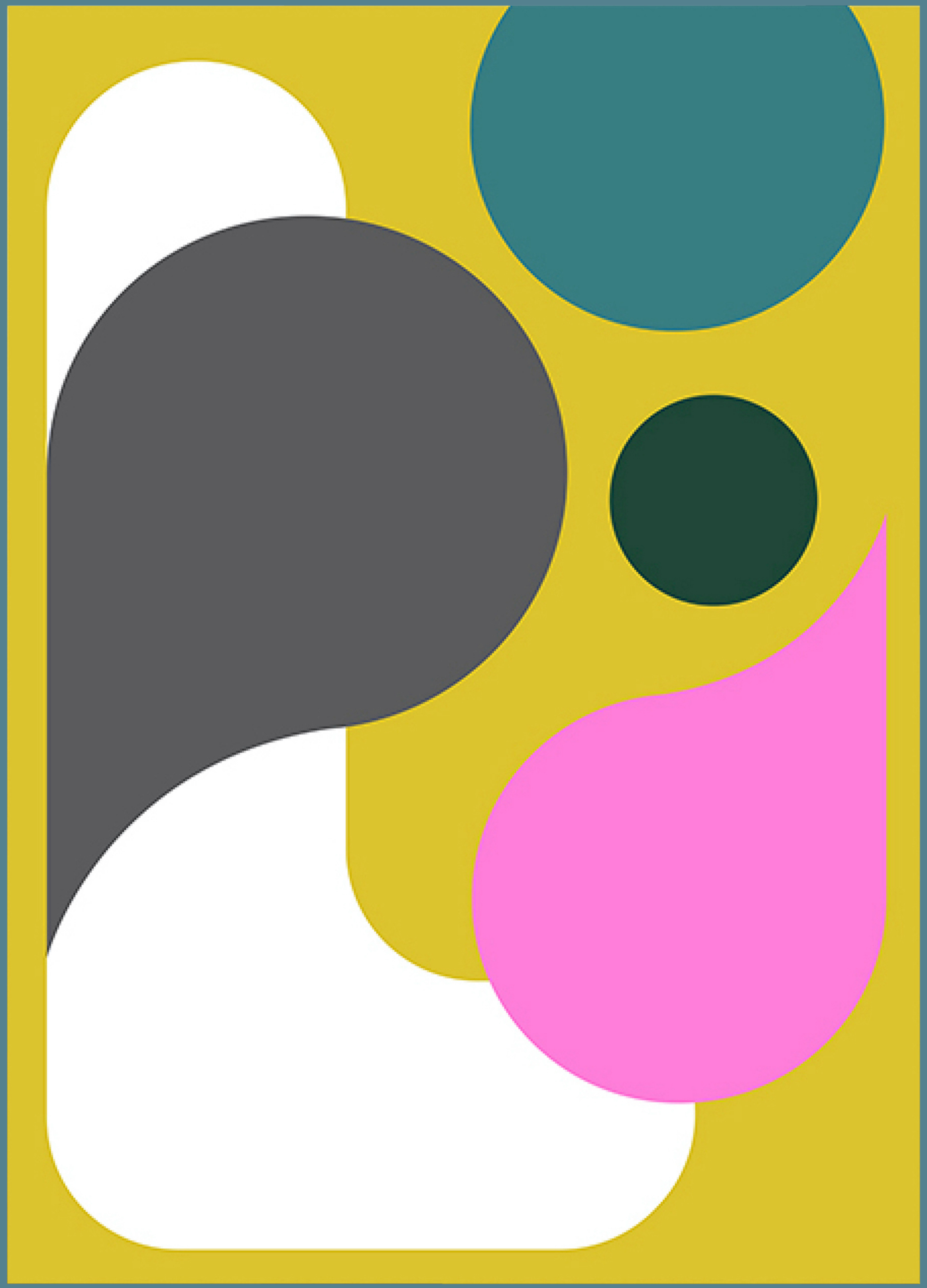
# STRUCTURAL AND PARALLEL LAWS

**Monad laws — structural equivalences for free, e.g. Hoisting**

$$[\![ \, \textbf{if } K_{\text{pure}} \textbf{ then } M; P_1 \textbf{ else } M; P_2 \, ]\!] = [\![ \, M; \textbf{if } K_{\text{pure}} \textbf{ then } P_1 \textbf{ else } P_2 \, ]\!]$$

**Interleaving — properties of parallel composition, e.g. generalized sequencing**

$$[\![ \, (M_1; M_2) \parallel (K_1; K_2) \, ]\!] \supseteq [\![ \, (M_1 \parallel K_1); (M_2 \parallel K_2) \, ]\!]$$



# ABSTRACTION

# SOPHISTICATION REQUIRED

Some transformations are valid due to **more complicated reasons**, e.g.:

**Redundant Read Elimination**

$$y?; M \twoheadrightarrow M$$

holds due to **delicate semantic invariants**

**Overwritten Write Elimination**

$$x := 0; x := 1 \twoheadrightarrow x := 1$$

holds even though **state diverges**

# DELICATE SEMANTIC INVARIANTS

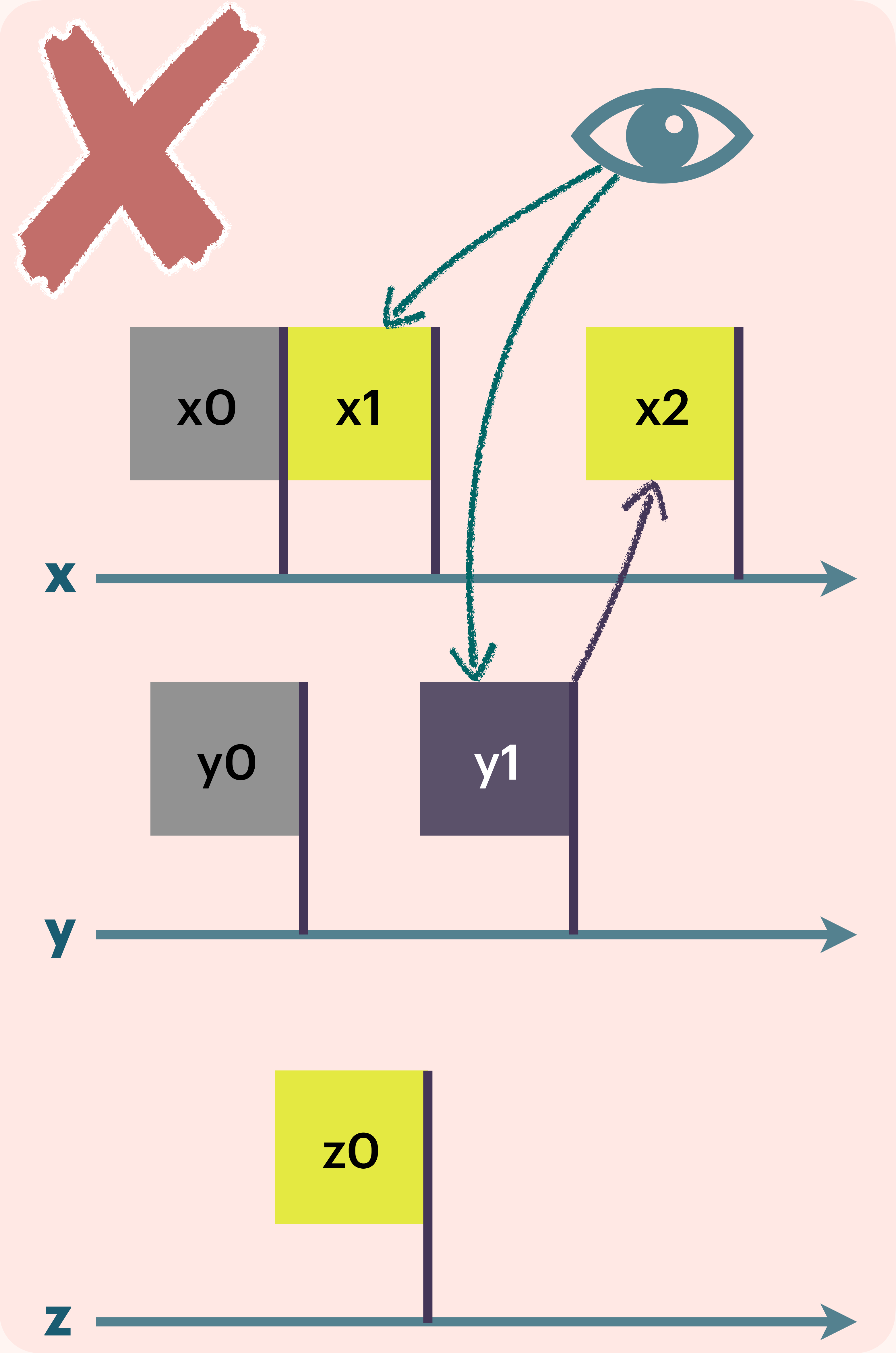
**Redundant Read Elimination**

$$y?;M \twoheadrightarrow M$$

**we identify operational invariants**

**and impose them as denotational requirements**

$$\kappa\boxed{\langle\mu,\mu\rangle}\kappa \therefore \langle\rangle \in [\![\langle\rangle]\!] \implies \exists v\,.\,\kappa\boxed{\langle\mu,\mu\rangle}\kappa \therefore v \in [\![y?]\!]$$



# DIVERGING STATE

**Overwritten Write Elimination**

$$x := 0; x := 1 \twoheadrightarrow x := 1$$

$$\alpha \, \boxed{\langle \mu, \mu \uplus \{ \; \boxed{1} \mid \; \} \rangle} \, \omega \therefore \langle \rangle$$

$$\Cup$$

$$[\![x := 0; x := 1]\!] \supseteq [\![x := 1]\!]$$

# DIVERGING STATE

**Overwritten Write Elimination**

$$x := 0; x := 1 \twoheadrightarrow x := 1$$

$$\alpha \langle \mu, \mu \uplus \{ \boxed{1} \} \rangle \omega \therefore \langle \rangle$$

$$\Cup$$

$$[\![x := 0; x := 1]\!] \supseteq [\![x := 1]\!]$$

$$\Cap$$

$$\alpha \langle \mu, \mu \uplus \{ \boxed{0} \} \rangle \langle \mu \uplus \{ \boxed{0} \}, \mu \uplus \{ \boxed{0} \boxed{1} \} \rangle \omega \therefore \langle \rangle$$

# DIVERGING STATE

**Overwritten Write Elimination**

$$x := 0; x := 1 \twoheadrightarrow x := 1$$

$$\alpha \langle \mu, \mu \uplus \{ \boxed{1} \} \rangle \omega \therefore \langle \rangle$$

$$\Cup$$

$$[\![ x := 0; x := 1 ]\!] \supseteq [\![ x := 1 ]\!]$$

$$\Cap$$

$$\alpha \langle \mu, \mu \uplus \{ \boxed{0} \boxed{1} \} \rangle \omega \therefore \langle \rangle$$

**mumble**

$$\alpha \langle \mu, \mu \uplus \{ \boxed{0} \} \rangle \langle \mu \uplus \{ \boxed{0} \}, \mu \uplus \{ \boxed{0} \boxed{1} \} \rangle \omega \therefore \langle \rangle$$

# DIVERGING STATE

**Overwritten Write Elimination**

$$x := 0; x := 1 \twoheadrightarrow x := 1$$

$$\alpha \langle \mu, \mu \uplus \{ \boxed{1} \} \rangle \omega \therefore \langle \rangle$$

$$\Cup$$

$$[\![x := 0; x := 1]\!] \supseteq [\![x := 1]\!]$$

**absorb**

$$\Cap$$

$$\alpha \langle \mu, \mu \uplus \{ \boxed{0 \mid 1} \} \rangle \omega \therefore \langle \rangle$$

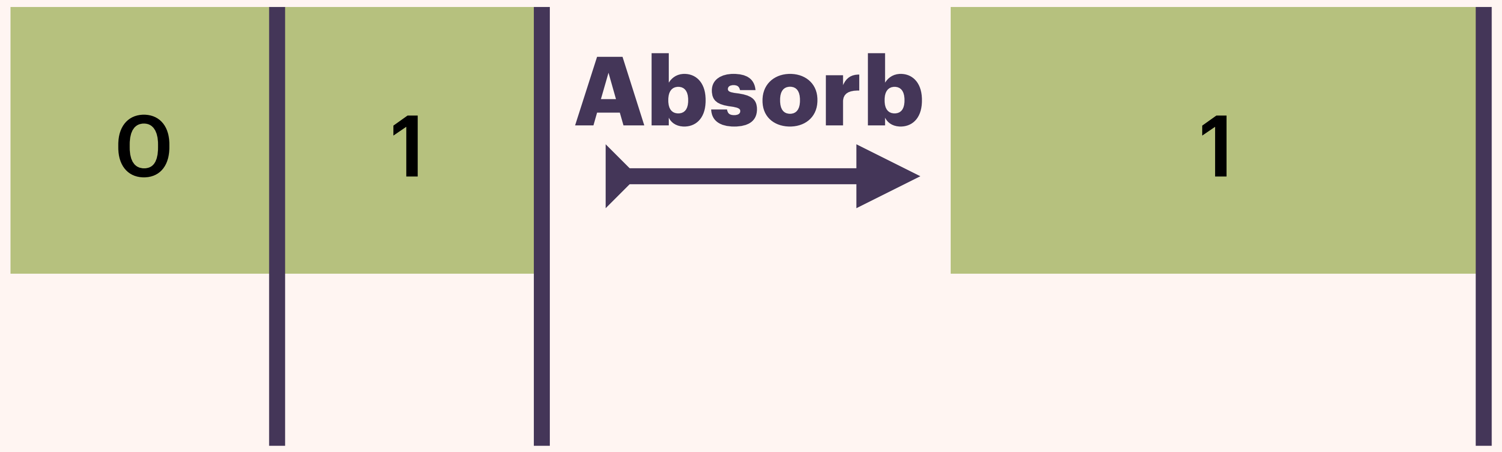
**mumble**

$$\alpha \langle \mu, \mu \uplus \{ \boxed{0} \} \rangle \langle \mu \uplus \{ \boxed{0} \}, \mu \uplus \{ \boxed{0 \mid 1} \} \rangle \omega \therefore \langle \rangle$$

# NO CORRESPONDENCE WITH INTERRUPTED EXECUTIONS

$$\alpha \, \langle \mu_1, \rho_1 \rangle \langle \mu_2, \rho_2 \rangle \ldots \langle \mu_{n-1}, \rho_{n-1} \rangle \langle \mu_n, \rho_n \rangle \, \omega \therefore r$$

$$\cdots \, \langle \mu_2, - \rangle, M_1 \to^* \langle \rho_2, - \rangle, M_2 \, \cdots$$



# ALL REWRITE RULES

| | | | | |
|---|---|---|---|---|
| Loosen | $\alpha \boxed{\xi\,(\eta \mathbin{\overline{\uplus}} \{\epsilon\})} \omega$ | $\xrightarrow{\mathsf{Ls}}$ | $\alpha \boxed{\xi\,(\eta \mathbin{\overline{\uplus}} \{\nu\})} \omega$ | $\nu \leq_{\mathtt{vw}} \epsilon$ |
| Expel | $\alpha \boxed{\xi\,(\eta \mathbin{\overline{\uplus}} \{\epsilon_{\mathtt{i}}^{\nu.\mathtt{i}}\})} \omega$ | $\xrightarrow{\mathsf{Ex}}$ | $\alpha \boxed{\xi\,(\eta \mathbin{\overline{\uplus}} \{\nu, \epsilon\})} \omega$ | $\nu \mathrel{-\!\!\subset} \epsilon$ |
| Condense | $\alpha \boxed{\xi\,(\eta \mathbin{\overline{\uplus}} \{\nu, \epsilon\})} \omega$ | $\xrightarrow{\mathsf{Cn}}$ | $\left(\alpha \boxed{\xi\,(\eta \mathbin{\overline{\uplus}} \{\nu\})} \omega\right) [\uparrow\epsilon]$ | $\nu \mathrel{-\!\!\in} \epsilon$ |
| Stutter | $\alpha \boxed{\xi\eta} \omega$ | $\xrightarrow{\mathsf{St}}$ | $\alpha \boxed{\xi \langle\mu, \mu\rangle \eta} \omega$ | |
| Mumble | $\alpha \boxed{\xi \langle\mu, \rho\rangle \langle\rho, \theta\rangle \eta} \omega$ | $\xrightarrow{\mathsf{Mu}}$ | $\alpha \boxed{\xi \langle\mu, \theta\rangle \eta} \omega$ | |
| Rewind | $\kappa \boxed{\xi} \omega$ | $\xrightarrow{\mathsf{Rw}}$ | $\alpha \boxed{\xi} \omega$ | $\alpha \leq \kappa$ |
| Forward | $\alpha \boxed{\xi} \kappa$ | $\xrightarrow{\mathsf{Fw}}$ | $\alpha \boxed{\xi} \omega$ | $\kappa \leq \omega$ |
| Tighten | $\alpha \boxed{\xi \langle\mu, \rho \uplus \{\nu\}\rangle \eta \mathbin{\overline{\uplus}} \{\nu\}} \omega$ | $\xrightarrow{\mathsf{Ti}}$ | $\alpha \boxed{\xi \langle\mu, \rho \uplus \{\epsilon\}\rangle \eta \mathbin{\overline{\uplus}} \{\epsilon\}} \omega$ | $\nu \leq_{\mathtt{vw}} \epsilon$ |
| Absorb | $\alpha \boxed{\xi \langle\mu, \rho \uplus \{\nu, \epsilon\}\rangle \eta \mathbin{\overline{\uplus}} \{\nu, \epsilon\}} \omega$ | $\xrightarrow{\mathsf{Ab}}$ | $\alpha \boxed{\xi \left\langle\mu, \rho \uplus \left\{\epsilon_{\mathtt{i}}^{\nu.\mathtt{i}}\right\}\right\rangle \eta \mathbin{\overline{\uplus}} \left\{\epsilon_{\mathtt{i}}^{\nu.\mathtt{i}}\right\}} \omega$ | $\nu \mathrel{-\!\!\subset} \epsilon$ |
| Dilute | $\left(\alpha \boxed{\xi \langle\mu, \rho \uplus \{\nu\}\rangle \eta \mathbin{\overline{\uplus}} \{\nu\}} \omega\right) [\uparrow\epsilon]$ | $\xrightarrow{\mathsf{Di}}$ | $\alpha \boxed{\xi \langle\mu, \rho \uplus \{\nu, \epsilon\}\rangle \eta \mathbin{\overline{\uplus}} \{\nu, \epsilon\}} \omega$ | $\nu \mathrel{-\!\!\in} \epsilon$ |

# NEW ADEQUACY PROOF IDEA

**Traces are not operational — adequacy proof is *significantly* more challenging:**

1. **We first define a denotational semantics $\underline{[\![ M ]\!]}$ but without the abstract rules**

2. **We show it is adequate — easier: traces correspond to interrupted executions**
   **(with an admissible view-advancing rule)**

3. **We show it is enough to apply the abstract closure $\dagger$ on top** $[\![ M ]\!] = \underline{[\![ M ]\!]}^{\dagger}$

   - ***This is the main technical challenge — complicated commutativity property***


4. **We show that the abstract rewrites preserve observable results**
   **(rather than interrupted executions)**

## Laws of Parallel Programming

| | | | |
|---|---|---|---|
| Symmetry | $M \parallel N$ | $\twoheadrightarrow$ | $\mathbf{match}\, N \parallel M\, \mathbf{with}\, \langle y, x \rangle .\ \langle x, y \rangle$ |
| Generalized Sequencing | | | |
| | $(\mathbf{let}\, x = M_1\, \mathbf{in}\, M_2) \parallel (\mathbf{let}\, y = N_1\, \mathbf{in}\, N_2)$ | $\twoheadrightarrow$ | $\mathbf{match}\, M_1 \parallel N_1\, \mathbf{with}\, \langle x, y \rangle .\, M_2 \parallel N_2$ |

## Eliminations

| | | | |
|---|---|---|---|
| Irrelevant Read | $\ell?\,;\langle\rangle$ | $\twoheadrightarrow$ | $\langle\rangle$ |
| Write-Write | $\ell := v\,;\ell := w$ | $\overset{\mathsf{Ab}}{\twoheadrightarrow}$ | $\ell := w$ |
| Write-Read | $\ell := v\,;\ell?$ | $\twoheadrightarrow$ | $\ell := v\,;v$ |
| Write-FAA | $\ell := v\,;\mathrm{FAA}\,(\ell, w)$ | $\overset{\mathsf{Ab}}{\twoheadrightarrow}$ | $\ell := (v + w)\,;v$ |
| Read-Write | $\mathbf{let}\, x = \ell?\, \mathbf{in}\, \ell := (x + v)\,;x$ | $\twoheadrightarrow$ | $\mathrm{FAA}\,(\ell, v)$ |
| Read-Read | $\langle \ell?, \ell? \rangle$ | $\twoheadrightarrow$ | $\mathbf{let}\, x = \ell?\, \mathbf{in}\, \langle x, x \rangle$ |
| Read-FAA | $\langle \ell?, \mathrm{FAA}\,(\ell, v) \rangle$ | $\twoheadrightarrow$ | $\mathbf{let}\, x = \mathrm{FAA}\,(\ell, v)\, \mathbf{in}\, \langle x, x \rangle$ |
| FAA-Read | $\langle \mathrm{FAA}\,(\ell, v)\,, \ell? \rangle$ | $\twoheadrightarrow$ | $\mathbf{let}\, x = \mathrm{FAA}\,(\ell, v)\, \mathbf{in}\, \langle x, x + v \rangle$ |
| FAA-FAA | $\langle \mathrm{FAA}\,(\ell, v)\,, \mathrm{FAA}\,(\ell, w) \rangle$ | $\overset{\mathsf{Ab}}{\twoheadrightarrow}$ | $\mathbf{let}\, x = \mathrm{FAA}\,(\ell, v + w)\, \mathbf{in}\, \langle x, x + v \rangle$ |

## Others

| | | | | |
|---|---|---|---|---|
| Irrelevant Read Introduction | $\langle\rangle$ | $\twoheadrightarrow$ | $\ell?\,;\langle\rangle$ | |
| Read to FAA | $\ell?$ | $\overset{\mathsf{Di}}{\twoheadrightarrow}$ | $\mathrm{FAA}\,(\ell, 0)$ | |
| Write-Read Deorder | $\langle (\ell := v)\,, \ell'? \rangle$ | $\overset{\mathsf{Ti}}{\twoheadrightarrow}$ | $(\ell := v) \parallel \ell'?$ | $(\ell \neq \ell')$ |
| Write-Read Reorder | $\langle (\ell := v)\,, \ell'? \rangle$ | $\overset{\mathsf{Ti}}{\twoheadrightarrow}$ | $\mathbf{let}\, x = \ell'?\, \mathbf{in}\, (\ell := v)\,;x$ | $(\ell \neq \ell')$ |

# CONCLUSION

- **Standard, adequate and fully-compositional denotational semantic for RA**
- **Sufficiently abstract: validates all RA transformations that we know of (memory access, laws of parallel programming, structural transformations)**
- **More nuanced, complicated traces**
  - **interpreted as Rely/Guarantee sequences**
  - **denotations closed under 10 rewrite rules**
- **Extended RA view-based machine with compositional (i.e. first-class) parallelism (weak-memory models are usually studied with top-level parallelism)**

# OPPORTUNITIES

- **Language features (e.g. recursion)**
- **Type-and-effect system (e.g. regions)**
- **Algebraic presentation (refines monad approach)**
- **Full C11 model (e.g. non-atomics)**
- **Full abstraction theorem (for first-order)?**