A DENOTATIONAL APPROACH TO RELEASE/ACQUIRE CONCURRENCY

Yotam Dvir, Tel Aviv University, yotamdvir.github.io

(Advisors: Ohad Kammar, Ori Lahav)

Moggi semantics
effects denote monads
[Moggi 1991]

➕ [BHN 2016]

Brookes semantics
traces denote behaviors
[Brookes 1996]

➕ [JPR 2012] For TSO

Relaxed memory
weakly consistent concurrent shared state

GOAL Moggi-style Brookes semantics for the Release/Acquire relaxed memory model

NEW CHALLENGES ABOUND

- Linear traces for a decentralized model
- More abstract and nuanced traces
- More closure rules
- First-class parallelism with causal propagation

Monad-based Denotational Semantics [Moggi 1991]

Modular framework for effectful semantics

compositionality: homomorphic semantics

$$[\![(\mathtt{k} := 1\,;\mathtt{m} := 1) \parallel \langle \mathtt{m?}, \mathtt{k?}\rangle]\!] = [\![\mathtt{k} := 1\,; \mathtt{m} := 1]\!] \mathbin{|||} [\![\langle \mathtt{m?}, \mathtt{k?}\rangle]\!]$$

sequencing denotes monadic bind

$$= ([\![\mathtt{k} := 1]\!] \gg\!= \lambda\langle\rangle.\, [\![\mathtt{m} := 1]\!]) \mathbin{|||} ([\![\mathtt{m?}]\!] \gg\!= \lambda v_{\mathtt{m}}.\, [\![\mathtt{k?}]\!] \gg\!= \lambda v_{\mathtt{k}}.\, \langle v_{\mathtt{m}}, v_{\mathtt{k}}\rangle)$$

Built-in: higher-order functions & **structural reasoning**, e.g.

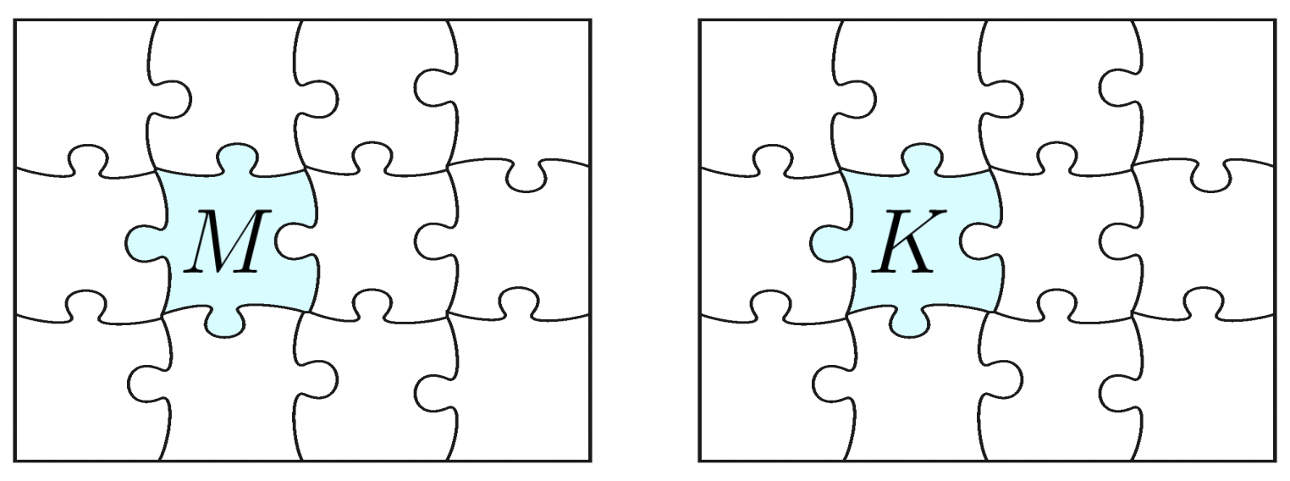
K effect-free $\Longrightarrow [\![\mathbf{if}\, K\, \mathbf{then}\, (M\,;N)\, \mathbf{else}\, (M\,;N')]\!] = [\![M\,; \mathbf{if}\, K\, \mathbf{then}\, N\, \mathbf{else}\, N']\!]$

Adequacy

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

Abstraction

$$M \overset{\checkmark}{\twoheadrightarrow} K \overset{?}{\implies} [\![M]\!] \geq [\![K]\!]$$

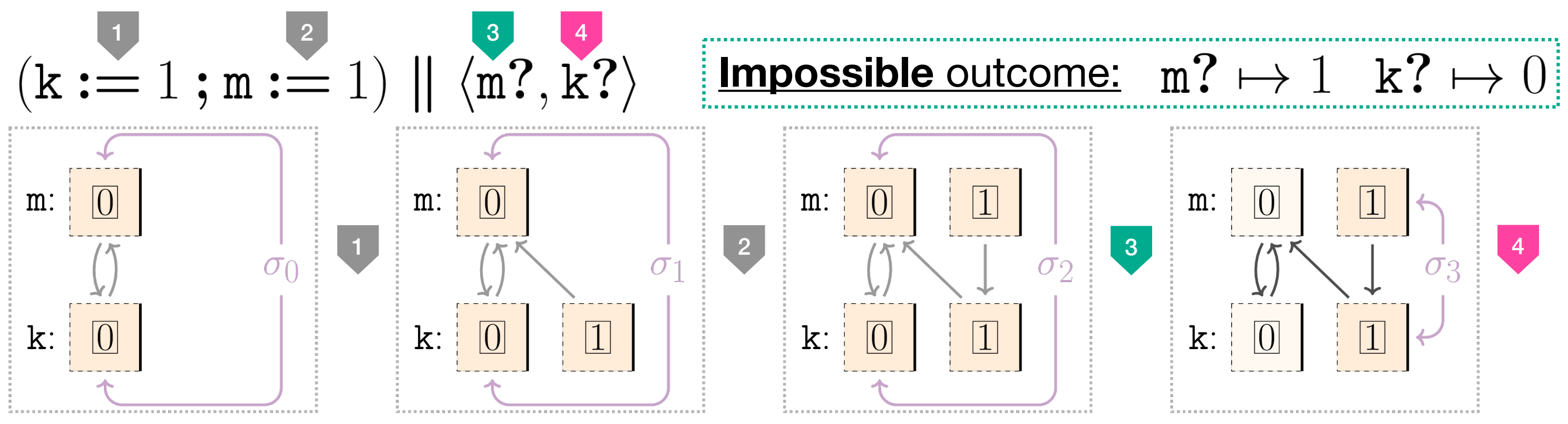


Release/Acquire Interleaving Semantics [KHLVD 2017]

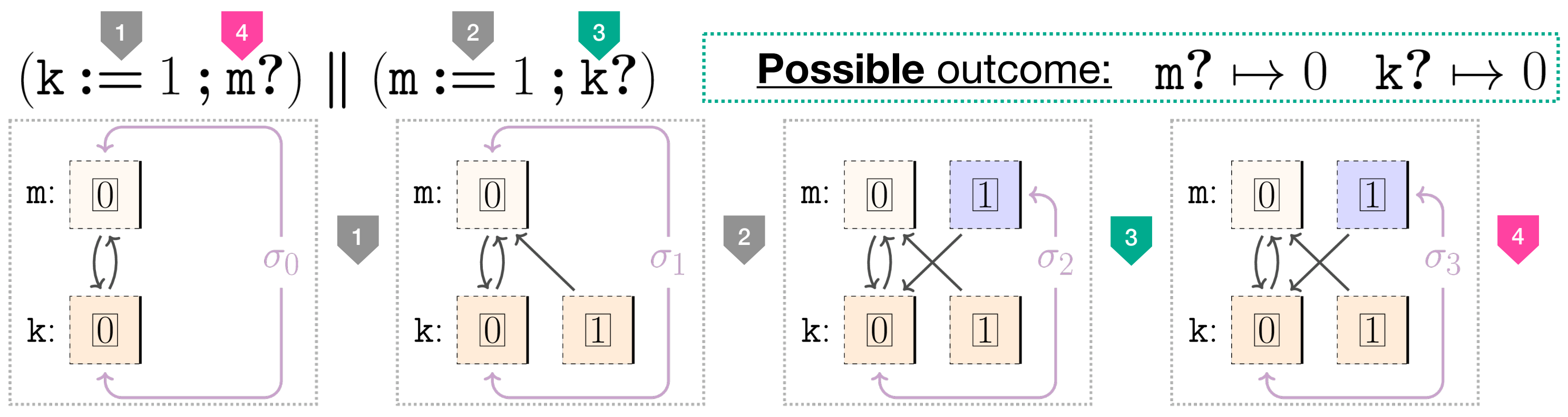
Fragment of the C/C++ model of causal propagation

Memory: msgs on timelines | **View:** accessible memory | **Threads** store/load views

$(\mathtt{k} := 1\,; \mathtt{m} := 1) \parallel \langle \mathtt{m?}, \mathtt{k?}\rangle$ **Impossible outcome:** $\mathtt{m?} \mapsto 1 \quad \mathtt{k?} \mapsto 0$



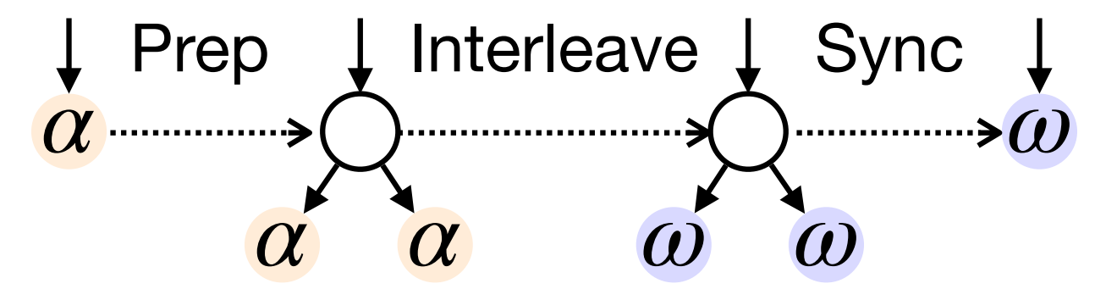
$(\mathtt{k} := 1\,; \mathtt{m?}) \parallel (\mathtt{m} := 1\,; \mathtt{k?})$ **Possible outcome:** $\mathtt{m?} \mapsto 0 \quad \mathtt{k?} \mapsto 0$



RA state invariants, e.g.
view σ point to msg $\nu \Longrightarrow \nu.\mathrm{view} \leq \sigma$

Admissible step: ADVANCE (pretend to load)

Thread views in trees (first-class parallelism)



JUSTIFIED TRANSFORMATIONS

Laws of Parallel Programming				
Symmetry	$M \parallel N$	$\twoheadrightarrow$	$\mathbf{swap}\ (N \parallel M)$	
Generalized Sequencing				
	$(\mathbf{let}\, a = M_1\, \mathbf{in}\, M_2) \parallel (\mathbf{let}\, b = N_1\, \mathbf{in}\, N_2)$	$\twoheadrightarrow$	$\mathbf{match}\, M_1 \parallel N_1\, \mathbf{with}\, \langle a, b\rangle.\, M_2 \parallel N_2$	
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	$\mathbf{let}\, a = \ell?\, \mathbf{in}\, \ell := (a + v)\,; a$	$\twoheadrightarrow$	$\mathrm{FAA}\,(\ell, v)$	
Read-Read	$\langle \ell?, \ell?\rangle$	$\twoheadrightarrow$	$\mathbf{let}\, a = \ell?\, \mathbf{in}\, \langle a, a\rangle$	
Read-FAA	$\langle \ell?, \mathrm{FAA}\,(\ell, v)\rangle$	$\twoheadrightarrow$	$\mathbf{let}\, a = \mathrm{FAA}\,(\ell, v)\ \mathbf{in}\ \langle a, a\rangle$	
FAA-Read	$\langle \mathrm{FAA}\,(\ell, v)\,, \ell?\rangle$	$\twoheadrightarrow$	$\mathbf{let}\, a = \mathrm{FAA}\,(\ell, v)\ \mathbf{in}\ \langle a, a + v\rangle$	
FAA-FAA	$\langle \mathrm{FAA}\,(\ell, v)\,, \mathrm{FAA}\,(\ell, w)\rangle$	$\overset{\mathsf{Ab}}{\twoheadrightarrow}$	$\mathbf{let}\, a = \mathrm{FAA}\,(\ell, v + w)\ \mathbf{in}\ \langle a, a + v\rangle$	
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	$(\ell := v)\,; \ell'?$	$\overset{\mathsf{Ti}}{\twoheadrightarrow}$	$\mathbf{fst}\ \langle \ell'?, (\ell := v)\rangle$	$(\ell \neq \ell')$

Trace-based Denotational Semantics [Brookes 1996]

Sequences of guarantees to/from the environment

Transition

$$\langle \mu, \varrho \rangle$$

Rely on memory — Guarantee memory

Trace

Sequence of transitions — Guarantee to return

$$\alpha \boxed{\xi} \omega \mathrel{.\!\cdot\!.} r$$

Rely on accessibility — Guarantee accessibility

Composition

Sequential

$$\frac{(\alpha \boxed{\xi} \sigma \mathrel{.\!\cdot\!.} r) \in P \qquad (\sigma \boxed{\eta} \omega \mathrel{.\!\cdot\!.} s) \in f(r)}{(\alpha \boxed{\xi\eta} \omega \mathrel{.\!\cdot\!.} s) \in P \gg\!= f}$$

Denotations

Sets of traces closed under REWRITE RULES

$$\tau \in [\![M]\!] \xRightarrow{\tau \overset{\star}{\to} \pi} \pi \in [\![M]\!]$$

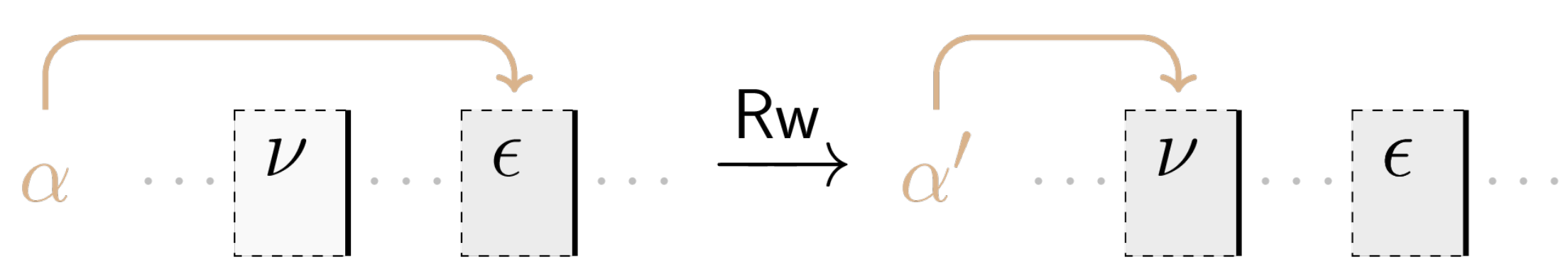
$$\star \in \{\mathsf{St}, \mathsf{Mu}, \mathsf{Rw}, \mathsf{Fw}, \mathsf{Ti}, \mathsf{Ab}, \mathsf{Di}\}$$

Parallel

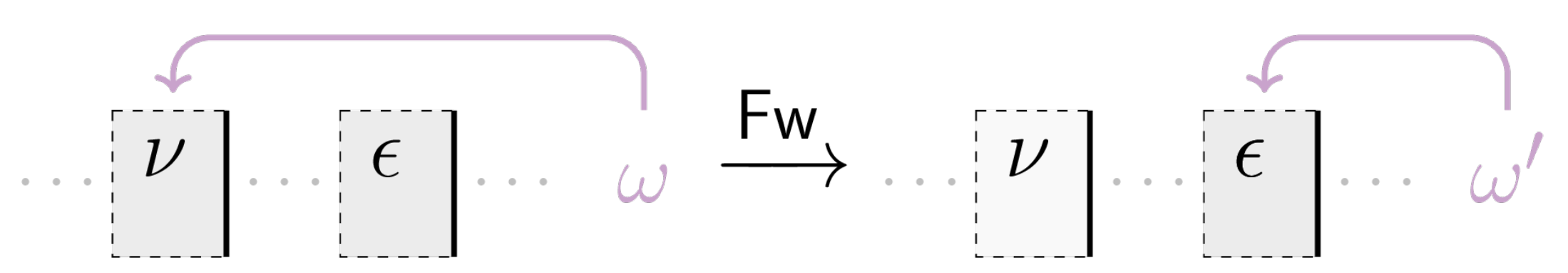
$$\frac{(\alpha \boxed{\xi_1} \omega \mathrel{.\!\cdot\!.} r_1) \in P_1 \quad (\alpha \boxed{\xi_2} \omega \mathrel{.\!\cdot\!.} r_2) \in P_2 \quad \xi \in \xi_1 \parallel \xi_2}{(\alpha \boxed{\xi} \omega \mathrel{.\!\cdot\!.} \langle r_1, r_2\rangle) \in P_1 \mathbin{|||} P_2}$$

REWRITE RULES

4 CONCRETE



Rewind (Rw) strengthens reliance on initial accessibility



Forward (Fw) weakens the guarantee of final accessibility

Stutter (St) propagates reliance as a guarantee

$$\boxed{\xi\eta} \xrightarrow{\mathsf{St}} \boxed{\xi\langle\mu, \mu\rangle\eta}$$

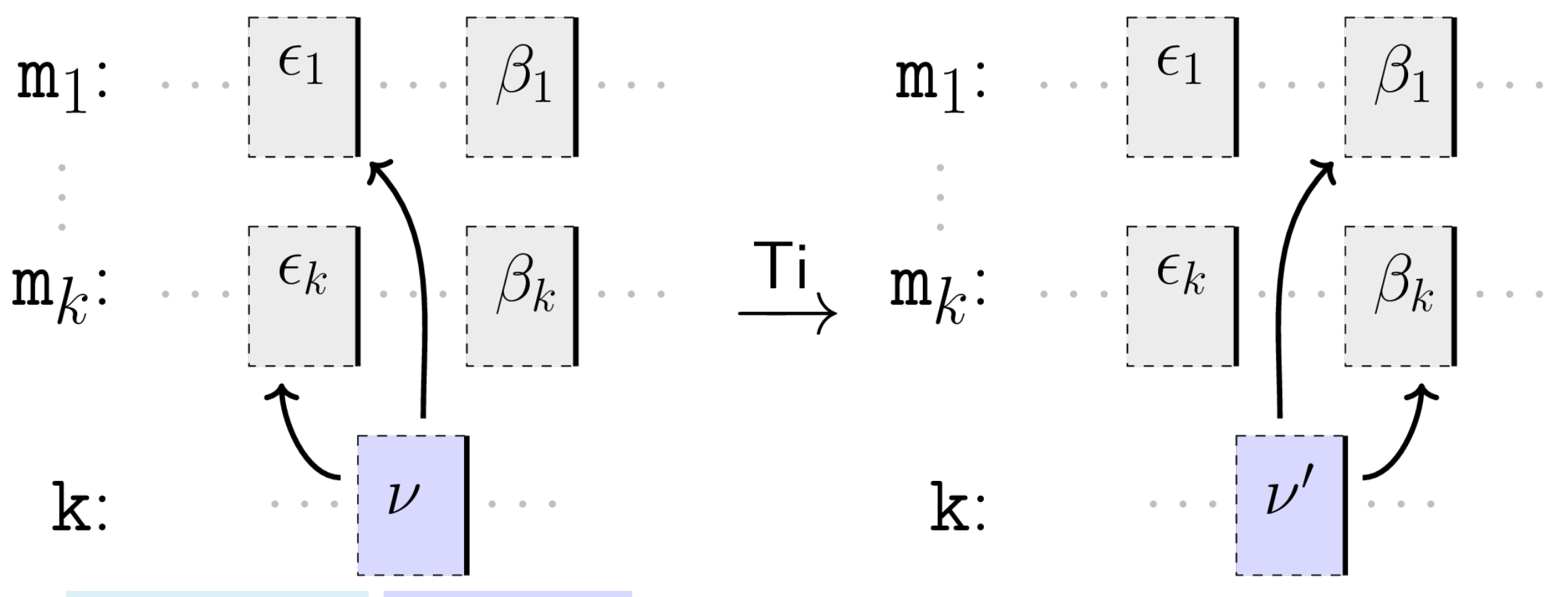
$$\boxed{\xi\langle\mu, \varrho\rangle\langle\varrho, \theta\rangle\eta} \xrightarrow{\mathsf{Mu}} \boxed{\xi\langle\mu, \theta\rangle\eta}$$

Mumble (Mu) omits a guarantee and relies on it internally

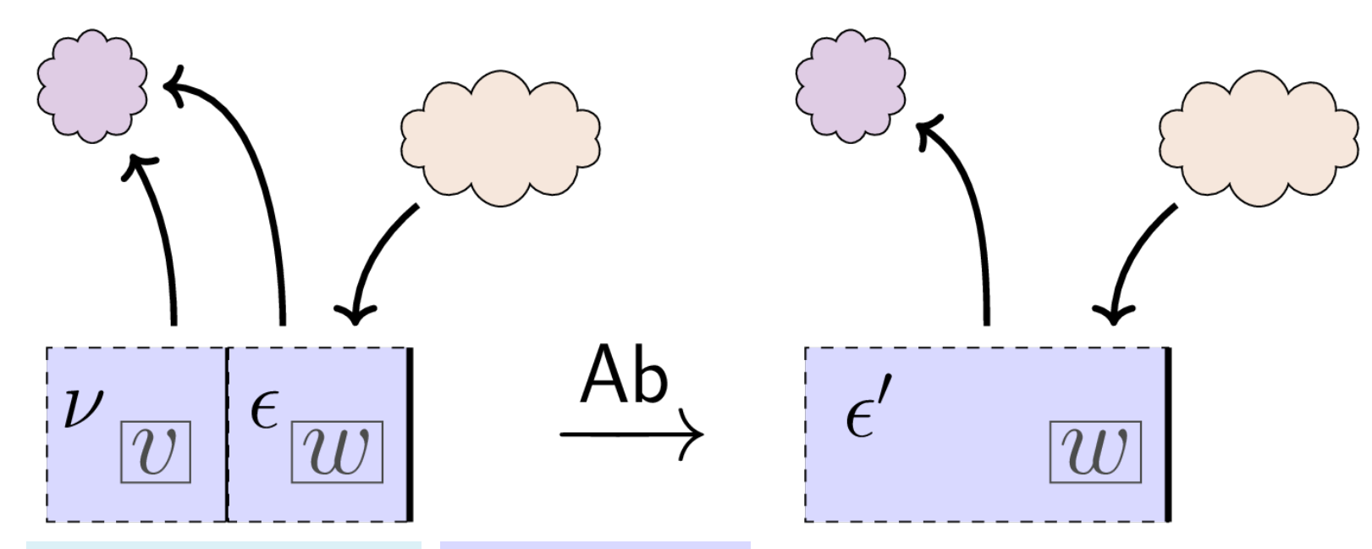
With only the CONCRETE rules the traces have an operational interpretation

3 ABSTRACT

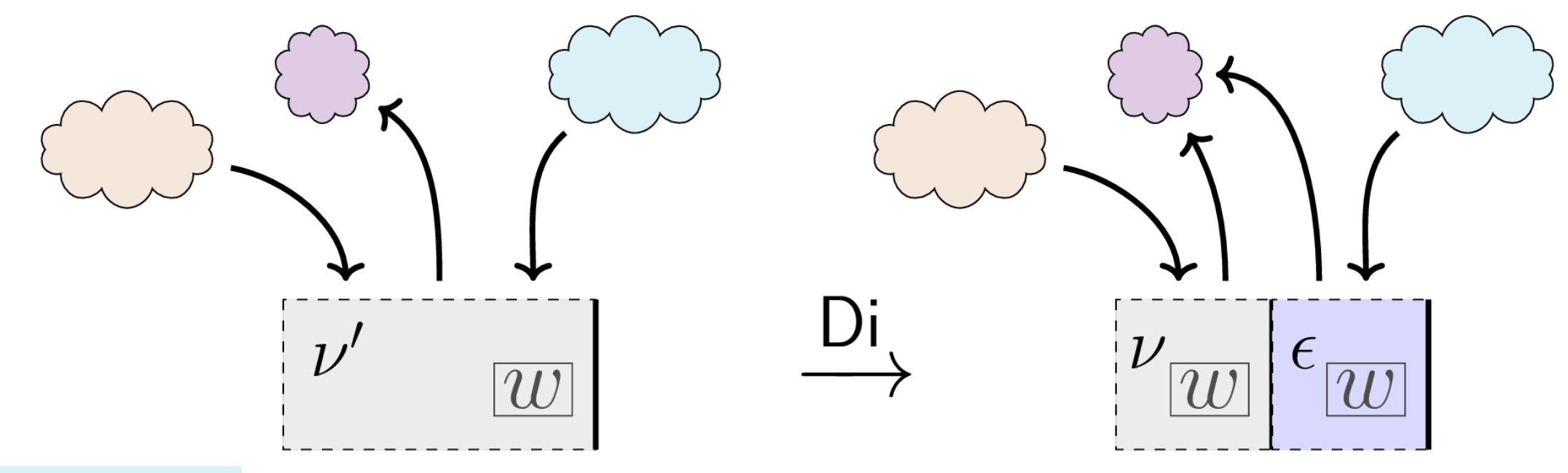
Challenge: non-operational traces **Solution:** percolate ABSTRACT rewrites out to induct



Tighten (Ti) guarantee less accessibility after reading



Absorb (Ab) guarantee fewer messages



Dilute (Di) touching value-view equals are indistinguishable

MAIN RESULTS

Moggi ➕ Brookes ➕ RA

A denotational semantics for Release/Acquire based on linear traces that is:

- ✸ **Standard** (monad base, truly compositional)
- ✸ **Adequate**
- ✸ **Abstract** (supports known transformations)

[Moggi 1991] Moggi, E.: Notions of computation and monads, Inf. Comput. [Brookes 1996] Brookes, S.D.: Full abstraction for a shared-variable parallel language, Inf. Comput. [JPR 2012] Jagadeesan, R., Petri, G., Riely, J.: Brookes is relaxed, almost!, FOSSACS
[KHLVD 2017] Kang, J., Hur, C., Lahav, O., Vafeiadis, V., Dreyer, D.: A promising semantics for relaxed-memory concurrency, POPL [BHN 2016] Benton, N., Hofmann, M., Nigam, V.: Effect-dependent transformations for concurrent programs, PPDP