



COLIN S. GORDON

ECOOP 2017

BARCELONA, SPAIN

A GENERIC APPROACH TO FLOW- SENSITIVE POLYMORPHIC EFFECTS

TODAY'S TALK

TODAY'S TALK

- ▶ Goal: Give an algebraic characterization of *sequential* effect systems, *sufficient* to model prior systems

TODAY'S TALK

- ▶ Goal: Give an algebraic characterization of *sequential* effect systems, *sufficient* to model prior systems
 - ▶ *Guide design, implementation, communication*

TODAY'S TALK

- ▶ Goal: Give an algebraic characterization of *sequential* effect systems, *sufficient* to model prior systems
 - ▶ *Guide design, implementation, communication*
- ▶ A new algebraic characterization of sequential effects

TODAY'S TALK

- ▶ Goal: Give an algebraic characterization of *sequential* effect systems, *sufficient* to model prior systems
 - ▶ *Guide design, implementation, communication*
- ▶ A new algebraic characterization of sequential effects
- ▶ Derivation of a free effect iteration for **most** sequential effect systems

TODAY'S TALK

- ▶ Goal: Give an algebraic characterization of *sequential* effect systems, *sufficient* to model prior systems
 - ▶ *Guide design, implementation, communication*
- ▶ A new algebraic characterization of sequential effects
- ▶ Derivation of a free effect iteration for **most** sequential effect systems
- ▶ Mention of other results in the paper

REVIEW: EFFECT SYSTEMS

REVIEW: EFFECT SYSTEMS

- ▶ Extend type systems to describe *internals of computations* as well as shape of data:

REVIEW: EFFECT SYSTEMS

- ▶ Extend type systems to describe *internals of computations* as well as shape of data:
 - ▶ $\Gamma \vdash e : \tau \implies \Gamma \vdash e : \tau \mid \chi$

REVIEW: EFFECT SYSTEMS

- ▶ Extend type systems to describe *internals of computations* as well as shape of data:
 - ▶ $\Gamma \vdash e : \tau \implies \Gamma \vdash e : \tau \mid \chi$
- ▶ Locking, memory access, non-termination, Java's checked exceptions...

REVIEW: EFFECT SYSTEMS

- ▶ Extend type systems to describe *internals of computations* as well as shape of data:
 - ▶ $\Gamma \vdash e : \tau \implies \Gamma \vdash e : \tau \mid \chi$
 - ▶ Locking, memory access, non-termination, Java's checked exceptions...
- ▶ For *most* effect systems, we have a concise formulation:

REVIEW: EFFECT SYSTEMS

- ▶ Extend type systems to describe *internals of computations* as well as shape of data:
 - ▶ $\Gamma \vdash e : \tau \implies \Gamma \vdash e : \tau \mid \chi$
 - ▶ Locking, memory access, non-termination, Java's checked exceptions...
- ▶ For *most* effect systems, we have a concise formulation:
 - ▶ A join semilattice of effects (partial order w/ LUB)

REVIEW: EFFECT SYSTEMS

- ▶ Extend type systems to describe *internals of computations* as well as shape of data:
 - ▶ $\Gamma \vdash e : \tau \implies \Gamma \vdash e : \tau \mid \chi$
 - ▶ Locking, memory access, non-termination, Java's checked exceptions...
- ▶ For *most* effect systems, we have a concise formulation:
 - ▶ A join semilattice of effects (partial order w/ LUB)
 - ▶ (More needed for effect masking)

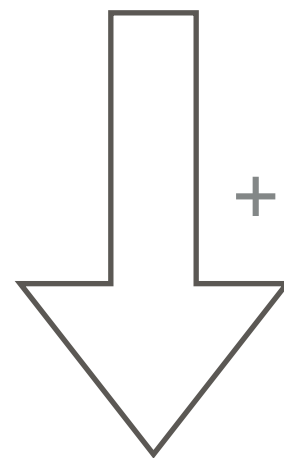
EFFECT SYSTEMS, GENERICALLY

EFFECT SYSTEMS, GENERICALLY

$$\text{T-Seq} \quad \frac{\Gamma \vdash e : \tau \mid \chi \quad \Gamma \vdash e' : \tau' \mid \chi'}{\Gamma \vdash e; e' : \tau' \mid \chi \sqcup \chi'}$$

EFFECT SYSTEMS, GENERICALLY

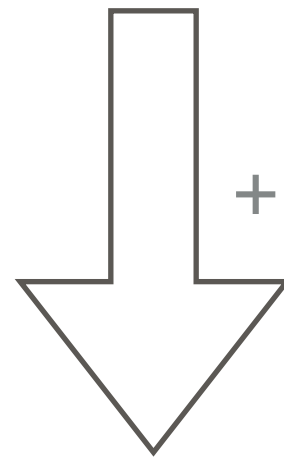
$$\text{T-Seq} \quad \frac{\Gamma \vdash e : \tau \mid \chi \quad \Gamma \vdash e' : \tau' \mid \chi'}{\Gamma \vdash e; e' : \tau' \mid \chi \sqcup \chi'}$$



+ plugin for checked exceptions

EFFECT SYSTEMS, GENERICALLY

$$\text{T-Seq} \quad \frac{\Gamma \vdash e : \tau \mid \chi \quad \Gamma \vdash e' : \tau' \mid \chi'}{\Gamma \vdash e; e' : \tau' \mid \chi \sqcup \chi'}$$



+ plugin for checked exceptions

$$\frac{\Gamma \vdash e : \tau \mid \{\text{IOException}\} \quad \Gamma \vdash e' : \tau' \mid \{\text{InvalidArgumentException}\}}{\Gamma \vdash e; e' : \tau' \mid \{\text{IOException}, \text{InvalidArgumentException}\}}$$

“MOST” EFFECT SYSTEMS: COMMUTATIVE EFFECT SYSTEMS

“MOST” EFFECT SYSTEMS: COMMUTATIVE EFFECT SYSTEMS

- ▶ Block-structured lock ownership (e.g., for data race freedom)

“MOST” EFFECT SYSTEMS: COMMUTATIVE EFFECT SYSTEMS

- ▶ Block-structured lock ownership (e.g., for data race freedom)
- ▶ Checked exceptions

“MOST” EFFECT SYSTEMS: COMMUTATIVE EFFECT SYSTEMS

- ▶ Block-structured lock ownership (e.g., for data race freedom)
- ▶ Checked exceptions
- ▶ Memory access (regions)

“MOST” EFFECT SYSTEMS: COMMUTATIVE EFFECT SYSTEMS

- ▶ Block-structured lock ownership (e.g., for data race freedom)
- ▶ Checked exceptions
- ▶ Memory access (regions)
- ▶ Use of capabilities

“MOST” EFFECT SYSTEMS: COMMUTATIVE EFFECT SYSTEMS

- ▶ Block-structured lock ownership (e.g., for data race freedom)
- ▶ Checked exceptions
- ▶ Memory access (regions)
- ▶ Use of capabilities
- ▶ Access to UI elements

“MOST” EFFECT SYSTEMS: COMMUTATIVE EFFECT SYSTEMS

- ▶ Block-structured lock ownership (e.g., for data race freedom)
- ▶ Checked exceptions
- ▶ Memory access (regions)
- ▶ Use of capabilities
- ▶ Access to UI elements
- ▶ Blocking calls

“MOST” EFFECT SYSTEMS: COMMUTATIVE EFFECT SYSTEMS

- ▶ Block-structured lock ownership (e.g., for data race freedom)
- ▶ Checked exceptions
- ▶ Memory access (regions)
- ▶ Use of capabilities
- ▶ Access to UI elements
- ▶ Blocking calls
- ▶ ...

“MOST” EFFECT SYSTEMS: COMMUTATIVE EFFECT SYSTEMS

- ▶ Block-structured lock ownership (e.g., for data race freedom)
- ▶ Checked exceptions
- ▶ Memory access (regions)
- ▶ Use of capabilities
- ▶ Access to UI elements
- ▶ Blocking calls
- ▶ ...

**ORDER
DOESN'T
MATTER!**

WHAT ABOUT EFFECT SYSTEMS *WITH* ORDERING?

WHAT ABOUT EFFECT SYSTEMS *WITH* ORDERING?

- ▶ Unstructured locking

WHAT ABOUT EFFECT SYSTEMS *WITH* ORDERING?

- ▶ Unstructured locking
- ▶ Unstructured memory accesses (regions)

WHAT ABOUT EFFECT SYSTEMS *WITH* ORDERING?

- ▶ Unstructured locking
- ▶ Unstructured memory accesses (regions)
- ▶ Heap-shape-dependent locking

WHAT ABOUT EFFECT SYSTEMS *WITH* ORDERING?

- ▶ Unstructured locking
- ▶ Unstructured memory accesses (regions)
- ▶ Heap-shape-dependent locking
- ▶ ...

WHAT ABOUT EFFECT SYSTEMS *WITH* ORDERING?

- ▶ Unstructured locking
- ▶ Unstructured memory accesses (regions)
- ▶ Heap-shape-dependent locking
- ▶ ...
- ▶ We call such systems “sequential” (following Tate)

WHAT ABOUT EFFECT SYSTEMS *WITH* ORDERING?

- ▶ Unstructured locking
- ▶ Unstructured memory accesses (regions)
- ▶ Heap-shape-dependent locking
- ▶ ...
- ▶ We call such systems “sequential” (following Tate)
- ▶ These systems lack a common algebraic characterization

WHAT DO WE NEED TO MODEL PRIOR SEQUENTIAL EFFECT SYSTEMS?

WHAT DO WE NEED TO MODEL PRIOR SEQUENTIAL EFFECT SYSTEMS?

- ▶ Still need a join semilattice

WHAT DO WE NEED TO MODEL PRIOR SEQUENTIAL EFFECT SYSTEMS?

- ▶ Still need a join semilattice
- ▶ Need (partial) sequencing of effects

WHAT DO WE NEED TO MODEL PRIOR SEQUENTIAL EFFECT SYSTEMS?

- ▶ Still need a join semilattice
- ▶ Need (partial) sequencing of effects
- ▶ Need iteration of effects

WHAT DO WE NEED TO MODEL PRIOR SEQUENTIAL EFFECT SYSTEMS?

- ▶ Still need a join semilattice
- ▶ Need (partial) sequencing of effects
- ▶ Need iteration of effects
- ▶ Need equational theory for simplifying complex effects with effect variables

EFFECT QUANTALES

EFFECT QUANTALES

- ▶ A relaxation of *quantales* (see *paper for references*)

EFFECT QUANTALES

- ▶ A relaxation of *quantales* (see paper for references)
- ▶ A set E with binary join $\sqcup$, binary sequence $\triangleright$, top $\top$, seq-unit I

EFFECT QUANTALES

- ▶ A relaxation of *quantales* (see paper for references)
- ▶ A set E with binary join $\sqcup$, binary sequence $\triangleright$, top $\top$, sequential I
- ▶ $\triangleright$ distributes over $\sqcup$ on both sides:
$$a \triangleright (b \sqcup c) = (a \triangleright b) \sqcup (a \triangleright c)$$
$$(b \sqcup c) \triangleright a = (b \triangleright a) \sqcup (c \triangleright a)$$

EFFECT QUANTALES

- ▶ A relaxation of *quantales* (see paper for references)
- ▶ A set E with binary join $\sqcup$, binary sequence $\triangleright$, top $\top$, seq-unit I
- ▶ $\triangleright$ distributes over $\sqcup$ on both sides:
$$a \triangleright (b \sqcup c) = (a \triangleright b) \sqcup (a \triangleright c)$$
$$(b \sqcup c) \triangleright a = (b \triangleright a) \sqcup (c \triangleright a)$$
- ▶ $\top$ is nilpotent for $\triangleright$ ($a \triangleright \top = \top = \top \triangleright a$)

EFFECT QUANTALES

- ▶ A relaxation of *quantales* (see paper for references)
- ▶ A set E with binary join $\sqcup$, binary sequence $\triangleright$, top $\top$, seq-unit I
- ▶ $\triangleright$ distributes over $\sqcup$ on both sides:
$$a \triangleright (b \sqcup c) = (a \triangleright b) \sqcup (a \triangleright c)$$
$$(b \sqcup c) \triangleright a = (b \triangleright a) \sqcup (c \triangleright a)$$
- ▶ $\top$ is nilpotent for $\triangleright$ ($a \triangleright \top = \top = \top \triangleright a$)

MANY USEFUL PROPERTIES
FOLLOW FROM THIS DEFINITION.

E.G.,
A PARTIAL ORDER $\sqsubseteq$
MONOTONICITY OF $\triangleright$

EFFECT QUANTALES

- ▶ A relaxation of *quantales* (see paper for references)
- ▶ A set E with binary join $\sqcup$, binary sequence $\triangleright$, top $\top$, seq-unit I
- ▶ $\triangleright$ distributes over $\sqcup$ on both sides:
$$a \triangleright (b \sqcup c) = (a \triangleright b) \sqcup (a \triangleright c)$$
$$(b \sqcup c) \triangleright a = (b \triangleright a) \sqcup (c \triangleright a)$$
- ▶ $\top$ is nilpotent for $\triangleright$ ($a \triangleright \top = \top = \top \triangleright a$)

MANY USEFUL PROPERTIES
FOLLOW FROM THIS DEFINITION.

E.G.,
A PARTIAL ORDER $\sqsubseteq$
MONOTONICITY OF $\triangleright$

THIS IS ENOUGH TO MODEL PRIOR
SYSTEMS!

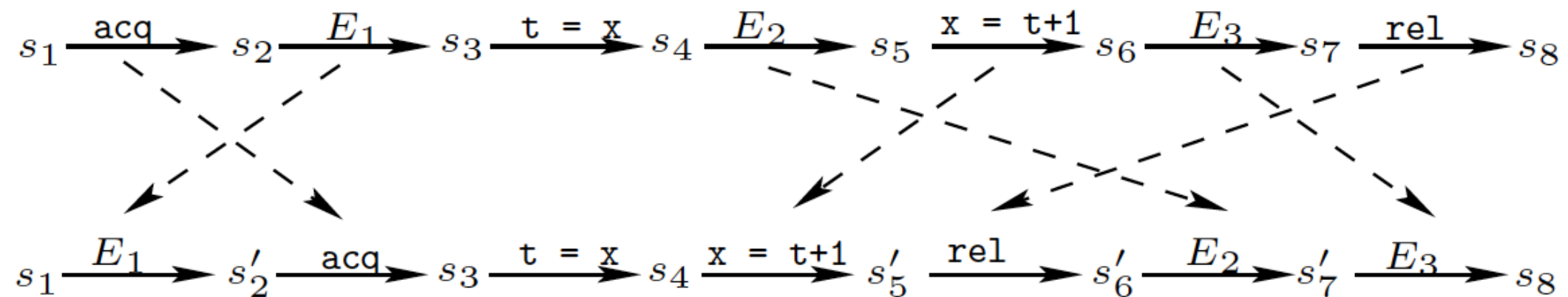
EXAMPLE: AN EFFECT SYSTEM FOR ATOMICITY

EXAMPLE: AN EFFECT SYSTEM FOR ATOMICITY

- ▶ Flanagan and Qadeer wrote *two* atomicity effect systems – let's model the simpler one (TLDI 2003)

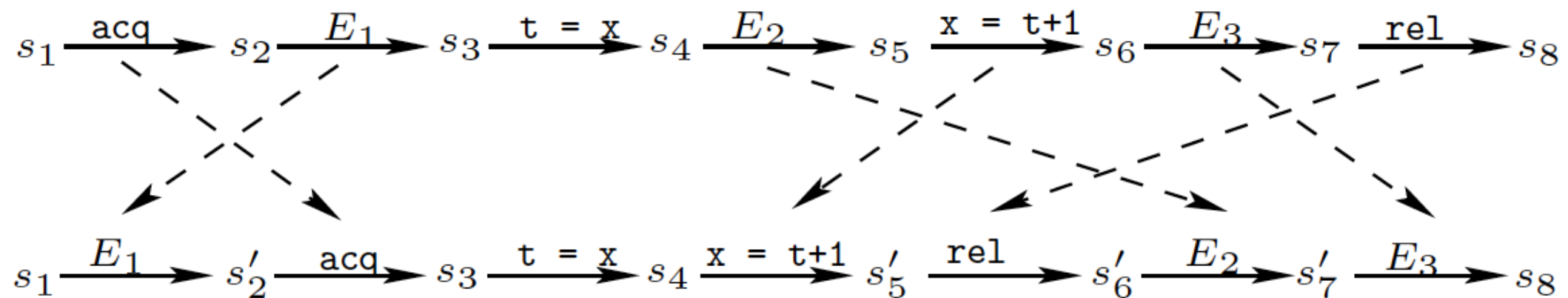
EXAMPLE: AN EFFECT SYSTEM FOR ATOMICITY

- ▶ Flanagan and Qadeer wrote *two* atomicity effect systems – let's model the simpler one (TLDI 2003)
- ▶ Movers (Lipton '75) are a way to reason about atomicity by considering how local actions *commute* with interference:



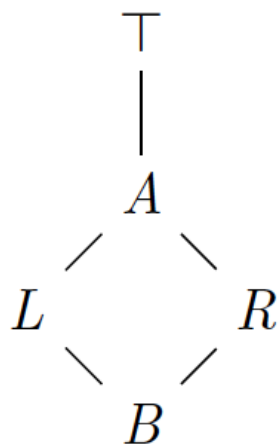
EXAMPLE: AN EFFECT SYSTEM FOR ATOMICITY

- ▶ Flanagan and Qadeer wrote *two* atomicity effect systems – let's model the simpler one (TLDI 2003)
- ▶ Movers (Lipton '75) are a way to reason about atomicity by considering how local actions *commute* with interference:



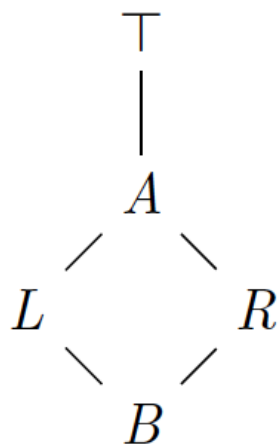
- ▶ The mover types become effects (B, L, R, A, C), with requisite sequencing

EXAMPLE: AN ATOMICITY EFFECT QUANTALE



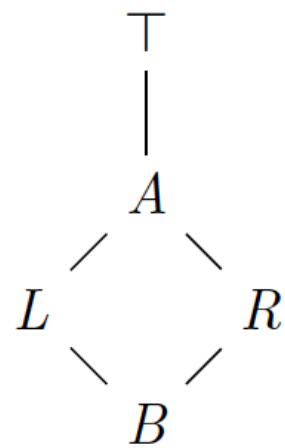
<i>;</i>	<i>B</i>	<i>L</i>	<i>R</i>	<i>A</i>	$\top$
<i>B</i>	<i>B</i>	<i>L</i>	<i>R</i>	<i>A</i>	$\top$
<i>R</i>	<i>R</i>	<i>A</i>	<i>R</i>	<i>A</i>	$\top$
<i>L</i>	<i>L</i>	<i>L</i>	$\top$	$\top$	$\top$
<i>A</i>	<i>A</i>	<i>A</i>	$\top$	$\top$	$\top$
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$

EXAMPLE: AN ATOMICITY EFFECT QUANTALE



$\top$	B	L	R	A	$\top$
B	B	L	R	A	$\top$
R	R	A	R	A	$\top$
L	L	L	$\top$	$\top$	$\top$
A	A	A	$\top$	$\top$	$\top$
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$

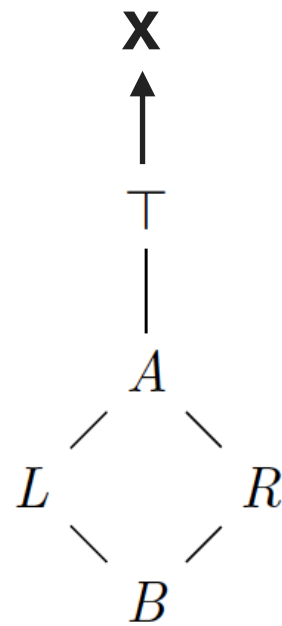
EXAMPLE: AN ATOMICITY EFFECT QUANTALE



$\cdot$	B	L	R	A	$\top$
B	B	L	R	A	$\top$
R	R	A	R	A	$\top$
L	L	L	$\top$	$\top$	$\top$
A	A	A	$\top$	$\top$	$\top$
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$

- The set is the mover effects + ERR

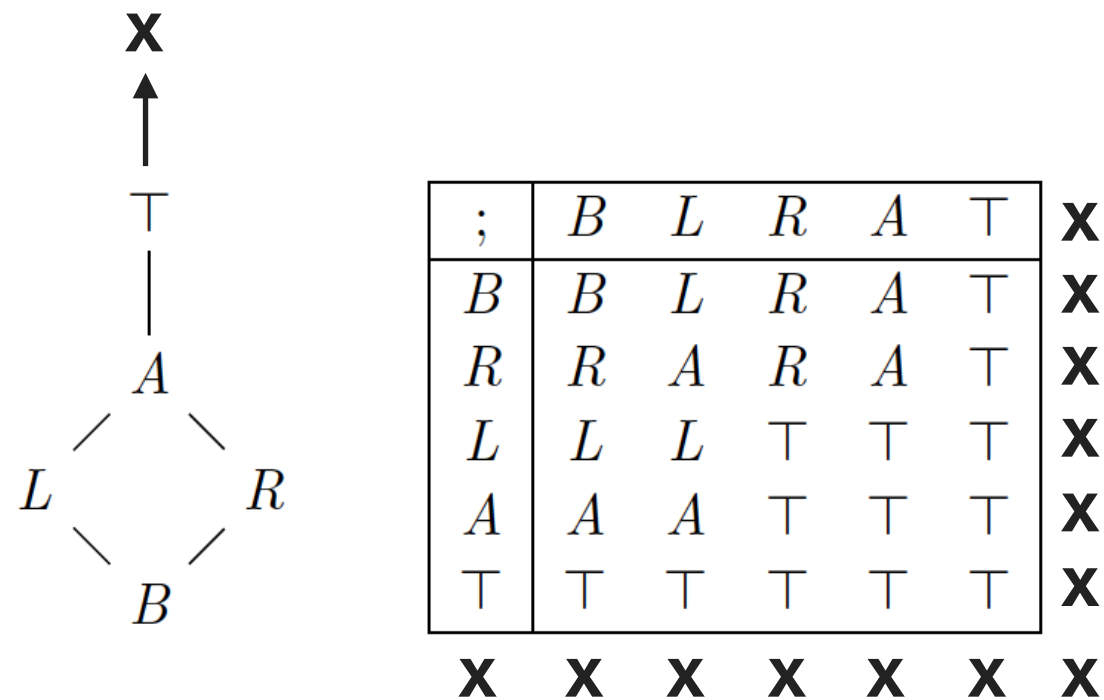
EXAMPLE: AN ATOMICITY EFFECT QUANTALE



$;$	B	L	R	A	T
B	B	L	R	A	T
R	R	A	R	A	T
L	L	L	T	T	T
A	A	A	T	T	T
T	T	T	T	T	T

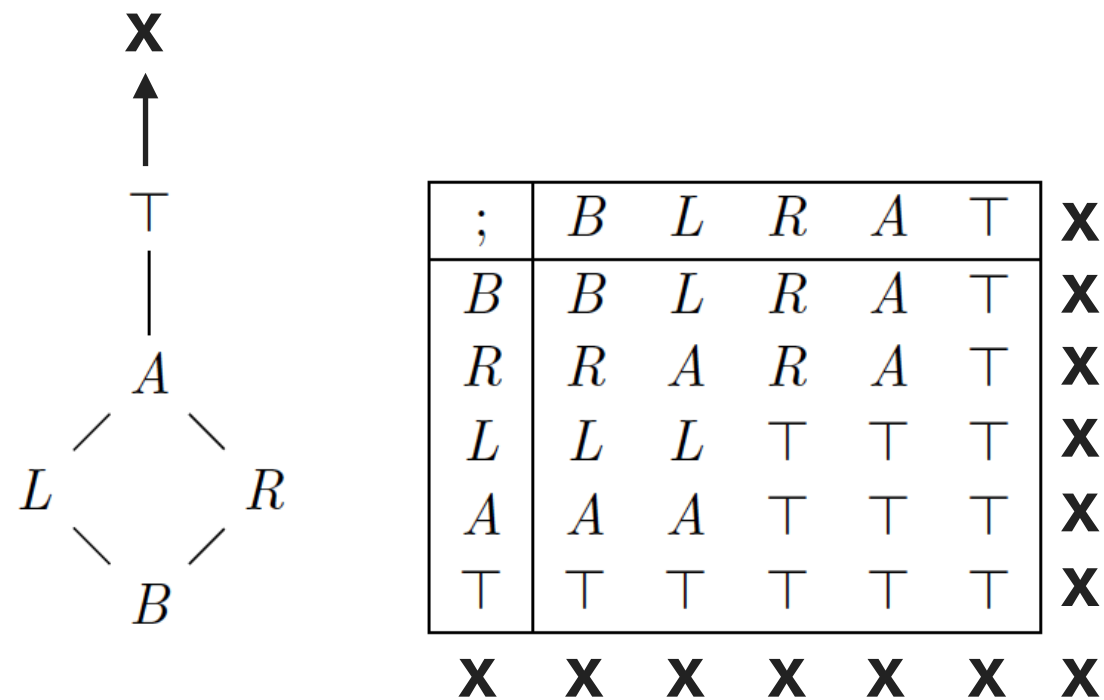
- ▶ The set is the mover effects + ERR
- ▶ Join follows Flanagan and Qadeer (plus ERR)

EXAMPLE: AN ATOMICITY EFFECT QUANTALE



- ▶ The set is the mover effects + ERR
- ▶ Join follows Flanagan and Qadeer (plus ERR)
- ▶ Sequencing follows Flanagan and Qadeer (plus ERR)

EXAMPLE: AN ATOMICITY EFFECT QUANTALE



- ▶ The set is the mover effects + ERR
- ▶ Join follows Flanagan and Qadeer (plus ERR)
- ▶ Sequencing follows Flanagan and Qadeer (plus ERR)
- ▶ Flanagan and Qadeer already proved the EQ laws

HOW GENERAL ARE EFFECT QUANTALES?

HOW GENERAL ARE EFFECT QUANTALES?

- ▶ EQs cover more than just Flanagan and Qadeer's atomicity

HOW GENERAL ARE EFFECT QUANTALES?

- ▶ EQs cover more than just Flanagan and Qadeer's atomicity
- ▶ Derived from prior systems' type judgments (see paper)

HOW GENERAL ARE EFFECT QUANTALES?

- ▶ EQs cover more than just Flanagan and Qadeer's atomicity
- ▶ Derived from prior systems' type judgments (see paper)
- ▶ Trickier examples: unstructured locking with recursive acquisition, product of effect quantales

HOW GENERAL ARE EFFECT QUANTALES?

- ▶ EQs cover more than just Flanagan and Qadeer's atomicity
- ▶ Derived from prior systems' type judgments (see paper)
- ▶ Trickier examples: unstructured locking with recursive acquisition, product of effect quantales
- ▶ Clear relationship to more "foundational" work

HOW GENERAL ARE EFFECT QUANTALES?

- ▶ EQs cover more than just Flanagan and Qadeer's atomicity
- ▶ Derived from prior systems' type judgments (see paper)
- ▶ Trickier examples: unstructured locking with recursive acquisition, product of effect quantales
- ▶ Clear relationship to more "foundational" work
 - ▶ Short version: similar algebras, EQs are slightly more restrictive, EQs induce the other algebras

HOW GENERAL ARE EFFECT QUANTALES?

- ▶ EQs cover more than just Flanagan and Qadeer's atomicity
- ▶ Derived from prior systems' type judgments (see paper)
- ▶ Trickier examples: unstructured locking with recursive acquisition, product of effect quantales
- ▶ Clear relationship to more "foundational" work
 - ▶ Short version: similar algebras, EQs are slightly more restrictive, EQs induce the other algebras
- ▶ Free iteration construct for most EQs!

ITERATING SEQUENTIAL EFFECTS: HARDER THAN IT LOOKS

$$\frac{\Gamma \vdash e : \text{bool} \mid \chi \quad \Gamma \vdash e' : \tau \mid \chi'}{\Gamma \vdash \text{while } (e) e' : \tau \mid \chi \triangleright (\chi' \triangleright \chi)^*}$$

ITERATING SEQUENTIAL EFFECTS: HARDER THAN IT LOOKS

$$\frac{\Gamma \vdash e : \text{bool} \mid \chi \quad \Gamma \vdash e' : \tau \mid \chi'}{\Gamma \vdash \text{while } (e) e' : \tau \mid \chi \triangleright (\chi' \triangleright \chi)^*}$$

- ▶ Prior abstract work on sequential effects defers iteration

ITERATING SEQUENTIAL EFFECTS: HARDER THAN IT LOOKS

$$\frac{\Gamma \vdash e : \text{bool} \mid \chi \quad \Gamma \vdash e' : \tau \mid \chi'}{\Gamma \vdash \text{while } (e) e' : \tau \mid \chi \triangleright (\chi' \triangleright \chi)^*}$$

- ▶ Prior abstract work on sequential effects defers iteration
- ▶ Mycroft et al. note that a naive fixed point operator makes every effect *idempotent* ($\forall X, X \triangleright X = X$), which is too strong

ITERATING SEQUENTIAL EFFECTS: HARDER THAN IT LOOKS

$$\frac{\Gamma \vdash e : \text{bool} \mid \chi \quad \Gamma \vdash e' : \tau \mid \chi'}{\Gamma \vdash \text{while } (e) e' : \tau \mid \chi \triangleright (\chi' \triangleright \chi)^*}$$

- ▶ Prior abstract work on sequential effects defers iteration
- ▶ Mycroft et al. note that a naive fixed point operator makes every effect *idempotent* ($\forall X, X \triangleright X = X$), which is too strong
- ▶ Many prior sequential effect systems with iteration are incompatible with that: e.g., Flanagan and Qadeer's work:
 $B \triangleright B = B \quad L \triangleright L = L \quad R \triangleright R = R \quad \underline{\mathbf{A} \triangleright \mathbf{A} = \mathbf{C}} \quad C \triangleright C = C$

ITERATING SEQUENTIAL EFFECTS: HARDER THAN IT LOOKS

$$\frac{\Gamma \vdash e : \text{bool} \mid \chi \quad \Gamma \vdash e' : \tau \mid \chi'}{\Gamma \vdash \text{while } (e) e' : \tau \mid \chi \triangleright (\chi' \triangleright \chi)^*}$$

- ▶ Prior abstract work on sequential effects defers iteration
- ▶ Mycroft et al. note that a naive fixed point operator makes every effect *idempotent* ($\forall X, X \triangleright X = X$), which is too strong
- ▶ Many prior sequential effect systems with iteration are incompatible with that: e.g., Flanagan and Qadeer's work:
 $B \triangleright B = B \quad L \triangleright L = L \quad R \triangleright R = R \quad \underline{\mathbf{A} \triangleright \mathbf{A} = \mathbf{C}} \quad C \triangleright C = C$

EFFECT QUANTALES INDUCE AN ITERATION OPERATOR
COMPATIBLE WITH PRIOR WORK!

A LITTLE BIT OF LATTICE THEORY: CLOSURE OPERATORS

A LITTLE BIT OF LATTICE THEORY: CLOSURE OPERATORS

- ▶ A *closure operator* on a poset P is a function $f:P \rightarrow P$ that is
 - ▶ Extensive: $\forall e, e \sqsubseteq f(e)$
 - ▶ Idempotent: $\forall e, f(f(e)) \sqsubseteq f(e)$
 - ▶ Monotone: $\forall e, e', e \sqsubseteq e' \Rightarrow f(e) \sqsubseteq f(e')$

A LITTLE BIT OF LATTICE THEORY: CLOSURE OPERATORS

- ▶ A *closure operator* on a poset P is a function $f:P \rightarrow P$ that is
 - ▶ Extensive: $\forall e, e \sqsubseteq f(e)$
 - ▶ Idempotent: $\forall e, f(f(e)) \sqsubseteq f(e)$
 - ▶ Monotone: $\forall e, e', e \sqsubseteq e' \Rightarrow f(e) \sqsubseteq f(e')$
- ▶ $\text{Codomain}(f)$ is also the set of fixed points of f

A LITTLE BIT OF LATTICE THEORY: CLOSURE OPERATORS

- ▶ A *closure operator* on a poset P is a function $f:P \rightarrow P$ that is
 - ▶ Extensive: $\forall e, e \sqsubseteq f(e)$
 - ▶ Idempotent: $\forall e, f(f(e)) \sqsubseteq f(e)$
 - ▶ Monotone: $\forall e, e', e \sqsubseteq e' \Rightarrow f(e) \sqsubseteq f(e')$
- ▶ $\text{Codomain}(f)$ is also the set of fixed points of f
- ▶ A closure operator (if it exists) is uniquely defined by its range
 - ▶ Simple check, constructive proof

A LITTLE BIT OF LATTICE THEORY: CLOSURE OPERATORS

- ▶ A *closure operator* on a poset P is a function $f:P \rightarrow P$ that is
 - ▶ Extensive: $\forall e, e \sqsubseteq f(e)$
 - ▶ Idempotent: $\forall e, f(f(e)) \sqsubseteq f(e)$
 - ▶ Monotone: $\forall e, e', e \sqsubseteq e' \Rightarrow f(e) \sqsubseteq f(e')$
- ▶ Codomain(f) is also the set of fixed points of f
- ▶ A closure operator (if it exists) is uniquely defined by its range
 - ▶ Simple check, constructive proof

} 2/5 laws required for iteration!

ITERATION VIA CLOSURE OPERATORS

ITERATION VIA CLOSURE OPERATORS

- ▶ Picking the results of iteration is easier to think about, constrained by properties

ITERATION VIA CLOSURE OPERATORS

- ▶ Picking the results of iteration is easier to think about, constrained by properties
- ▶ Other 3/5 iteration laws require the range elements are idempotent, closed under joins, and above I

ITERATION VIA CLOSURE OPERATORS

- ▶ Picking the results of iteration is easier to think about, constrained by properties
 - ▶ Other 3/5 iteration laws require the range elements are idempotent, closed under joins, and above I
- ▶ *Taking X to the least idempotent element above $X \sqcup I$ is a valid closure operator satisfying all 5 iteration laws*

ITERATION VIA CLOSURE OPERATORS

- ▶ Picking the results of iteration is easier to think about, constrained by properties
 - ▶ Other 3/5 iteration laws require the range elements are idempotent, closed under joins, and above I
- ▶ *Taking X to the least idempotent element above $X \sqcup I$ is a valid closure operator satisfying all 5 iteration laws*
 - ▶ Under some mild conditions

ITERATION VIA CLOSURE OPERATORS

- ▶ Picking the results of iteration is easier to think about, constrained by properties
 - ▶ Other 3/5 iteration laws require the range elements are idempotent, closed under joins, and above I
- ▶ *Taking X to the least idempotent element above $X \sqcup I$ is a valid closure operator satisfying all 5 iteration laws*
- ▶ Under some mild conditions

CLOSURE OPERATORS ALSO
APPLY TO SEMANTIC
APPROACHES

DOES ITERATION DO WHAT WE WANT? YES!

DOES ITERATION DO WHAT WE WANT? YES!

- ▶ For the EQ induced by a commutative system (i.e., reuse join as sequencing), iteration is the identity function, as expected

DOES ITERATION DO WHAT WE WANT? YES!

- ▶ For the EQ induced by a commutative system (i.e., reuse join as sequencing), iteration is the identity function, as expected
- ▶ For the atomicity EQ, the derived operator coincides with Flanagan and Qadeer's hand-constructed version

DOES ITERATION DO WHAT WE WANT? YES!

- ▶ For the EQ induced by a commutative system (i.e., reuse join as sequencing), iteration is the identity function, as expected
- ▶ For the atomicity EQ, the derived operator coincides with Flanagan and Qadeer's hand-constructed version
- ▶ For lock ownership:
 - ▶ Iterating acquire/release is an error
 - ▶ Iterating something that preserves lock ownership is the identity
 - ▶ i.e., iteration is valid only for loop-invariant lock ownership

ALSO IN THE PAPER

ALSO IN THE PAPER

- ▶ An *abstract* core language with singleton effects and effect polymorphism, parameterized by effect quantale and primitives

ALSO IN THE PAPER

- ▶ An *abstract* core language with singleton effects and effect polymorphism, parameterized by effect quantale and primitives
- ▶ Effect-preserving translation between Flanagan-Qadeer calculus and (instantiation of) our abstract core language

ALSO IN THE PAPER

- ▶ An *abstract* core language with singleton effects and effect polymorphism, parameterized by effect quantale and primitives
- ▶ Effect-preserving translation between Flanagan-Qadeer calculus and (instantiation of) our abstract core language
- ▶ Precise (formal) relationship to prior semantic work

ALSO IN THE PAPER

- ▶ An *abstract* core language with singleton effects and effect polymorphism, parameterized by effect quantale and primitives
- ▶ Effect-preserving translation between Flanagan-Qadeer calculus and (instantiation of) our abstract core language
- ▶ Precise (formal) relationship to prior semantic work
- ▶ Subtleties related to substitution with singleton effects

ALSO IN THE PAPER

- ▶ An *abstract* core language with singleton effects and effect polymorphism, parameterized by effect quantale and primitives
- ▶ Effect-preserving translation between Flanagan-Qadeer calculus and (instantiation of) our abstract core language
- ▶ Precise (formal) relationship to prior semantic work
- ▶ Subtleties related to substitution with singleton effects

THANKS! QUESTIONS?

BACKUP SLIDES

SEQUENTIAL EFFECT SYSTEMS

SEQUENTIAL EFFECT SYSTEMS

$$\frac{\Gamma \vdash e : \tau \mid \chi \quad \Gamma \vdash e' : \tau' \mid \chi'}{\Gamma \vdash e; e' : \tau' \mid \chi \triangleright \chi'}$$

SEQUENTIAL EFFECT SYSTEMS

$$\frac{\Gamma \vdash e : \tau \mid \chi \quad \Gamma \vdash e' : \tau' \mid \chi'}{\Gamma \vdash e; e' : \tau' \mid \chi \triangleright \chi'}$$

$$\frac{\Gamma \vdash e : \text{bool} \mid \chi \quad \Gamma \vdash e' : \tau \mid \chi'}{\Gamma \vdash \text{while } (e) e' : \tau \mid \chi \triangleright (\chi' \triangleright \chi)^*}$$

OTHER SEQUENTIAL EFFECT SYSTEMS

OTHER SEQUENTIAL EFFECT SYSTEMS

- ▶ Some effect systems have “pre” and “post” states Δ , like lock sets, or heap shapes

OTHER SEQUENTIAL EFFECT SYSTEMS

- ▶ Some effect systems have “pre” and “post” states Δ , like lock sets, or heap shapes

$$\frac{\Gamma; \Delta \vdash e : \tau \dashv \Delta' \mid \chi \quad \Gamma; \Delta' \vdash e' : \tau' \dashv \Delta'' \mid \chi'}{\Gamma; \Delta \vdash e; e' : \tau' \dashv \Delta'' \mid \chi \triangleright \chi'}$$

OTHER SEQUENTIAL EFFECT SYSTEMS

- Some effect systems have “pre” and “post” states Δ , like lock sets, or heap shapes

$$\frac{\Gamma; \Delta \vdash e : \tau \dashv \Delta' \mid \chi \quad \Gamma; \Delta' \vdash e' : \tau' \dashv \Delta'' \mid \chi'}{\Gamma; \Delta \vdash e; e' : \tau' \dashv \Delta'' \mid \chi \triangleright \chi'}$$

$$\frac{\Gamma; \Delta \vdash e : \text{bool} \dashv \Delta' \mid \chi \quad \Gamma; \Delta' \vdash e' : \tau \dashv \Delta \mid \chi'}{\Gamma; \Delta \vdash \text{while } (e) e' : \tau \dashv \Delta' \mid \chi \triangleright (\chi' \triangleright \chi)^*}$$

OTHER SEQUENTIAL EFFECT SYSTEMS

- ▶ Some effect systems have “pre” and “post” states Δ , like lock sets, or heap shapes

$$\frac{\Gamma; \Delta \vdash e : \tau \dashv \Delta' \mid \chi \quad \Gamma; \Delta' \vdash e' : \tau' \dashv \Delta'' \mid \chi'}{\Gamma; \Delta \vdash e; e' : \tau' \dashv \Delta'' \mid \chi \triangleright \chi'}$$

$$\frac{\Gamma; \Delta \vdash e : \text{bool} \dashv \Delta' \mid \chi \quad \Gamma; \Delta' \vdash e' : \tau \dashv \Delta \mid \chi'}{\Gamma; \Delta \vdash \text{while } (e) e' : \tau \dashv \Delta' \mid \chi \triangleright (\chi' \triangleright \chi)^*}$$

- ▶ This obscures the fact that Δ and χ are managed the same way!

OTHER SEQUENTIAL EFFECT SYSTEMS — REWRITTEN

OTHER SEQUENTIAL EFFECT SYSTEMS — REWRITTEN

$$\frac{\Gamma \vdash e : \tau \mid (\Delta \leadsto \Delta') \otimes \chi \quad \Gamma \vdash e' : \tau' \mid (\Delta' \leadsto \Delta'') \otimes \chi'}{\Gamma \vdash e; e' : \tau' \mid ((\Delta \leadsto \Delta') \triangleright (\Delta' \leadsto \Delta'')) \otimes (\chi \triangleright \chi')}$$

OTHER SEQUENTIAL EFFECT SYSTEMS — REWRITTEN

$$\frac{\Gamma \vdash e : \tau \mid (\Delta \leadsto \Delta') \otimes \chi \quad \Gamma \vdash e' : \tau' \mid (\Delta' \leadsto \Delta'') \otimes \chi'}{\Gamma \vdash e; e' : \tau' \mid ((\Delta \leadsto \Delta') \triangleright (\Delta' \leadsto \Delta'')) \otimes (\chi \triangleright \chi')}$$

$$\frac{\Gamma \vdash e : \text{bool} \mid (\Delta \leadsto \Delta') \otimes \chi \quad \Gamma \vdash e' : \tau \mid (\Delta' \leadsto \Delta) \otimes \chi'}{\Gamma \vdash \text{while } (e) e' : \tau \mid ((\Delta \leadsto \Delta') \triangleright ((\Delta' \leadsto \Delta) \triangleright (\Delta \leadsto \Delta'))^*) \otimes (\chi \triangleright (\chi' \triangleright \chi)^*)}$$

OTHER SEQUENTIAL EFFECT SYSTEMS — REWRITTEN

$$\frac{\Gamma \vdash e : \tau \mid (\Delta \rightsquigarrow \Delta') \otimes \chi \quad \Gamma \vdash e' : \tau' \mid (\Delta' \rightsquigarrow \Delta'') \otimes \chi'}{\Gamma \vdash e; e' : \tau' \mid ((\Delta \rightsquigarrow \Delta') \triangleright (\Delta' \rightsquigarrow \Delta'')) \otimes (\chi \triangleright \chi')}$$

$$\frac{\Gamma \vdash e : \text{bool} \mid (\Delta \rightsquigarrow \Delta') \otimes \chi \quad \Gamma \vdash e' : \tau \mid (\Delta' \rightsquigarrow \Delta) \otimes \chi'}{\Gamma \vdash \text{while } (e) e' : \tau \mid ((\Delta \rightsquigarrow \Delta') \triangleright ((\Delta' \rightsquigarrow \Delta) \triangleright (\Delta \rightsquigarrow \Delta')^*)) \otimes (\chi \triangleright (\chi' \triangleright \chi)^*)}$$

- ▶ We can run two effect systems at once!
- ▶ Look at the $(\Delta \rightsquigarrow \Delta')$ effects – there is no natural bottom for their lattice!

DESIDERATA FOR ITERATED EFFECTS: X^*

DESIDERATA FOR ITERATED EFFECTS: X^*

► P1: $\forall e, e \sqsubseteq e^*$  EXTENSIVE

DESIDERATA FOR ITERATED EFFECTS: X^*

- ▶ P1: $\forall e, e \sqsubseteq e^*$ **EXTENSIVE**
- ▶ P2: $\forall e, e \triangleright e^* \sqsubseteq e^*$ and $e^* \triangleright e \sqsubseteq e^*$ **FOLDING**

DESIDERATA FOR ITERATED EFFECTS: X^*

- ▶ P1: $\forall e, e \sqsubseteq e^*$ **EXTENSIVE**
- ▶ P2: $\forall e, e \triangleright e^* \sqsubseteq e^*$ and $e^* \triangleright e \sqsubseteq e^*$ **FOLDING**
- ▶ P3: $\forall e, (e^*)^* = e^*$ **IDEMPOTENT**

DESIDERATA FOR ITERATED EFFECTS: X^*

- ▶ P1: $\forall e, e \sqsubseteq e^*$ **EXTENSIVE**
- ▶ P2: $\forall e, e \triangleright e^* \sqsubseteq e^*$ and $e^* \triangleright e \sqsubseteq e^*$ **FOLDING**
- ▶ P3: $\forall e, (e^*)^* = e^*$ **IDEMPOTENT**
- ▶ P4: $\forall e, f, (e \sqcup f)^* = e^* \sqcup f^*$ **DISTRIBUTIVE**

DESIDERATA FOR ITERATED EFFECTS: X^*

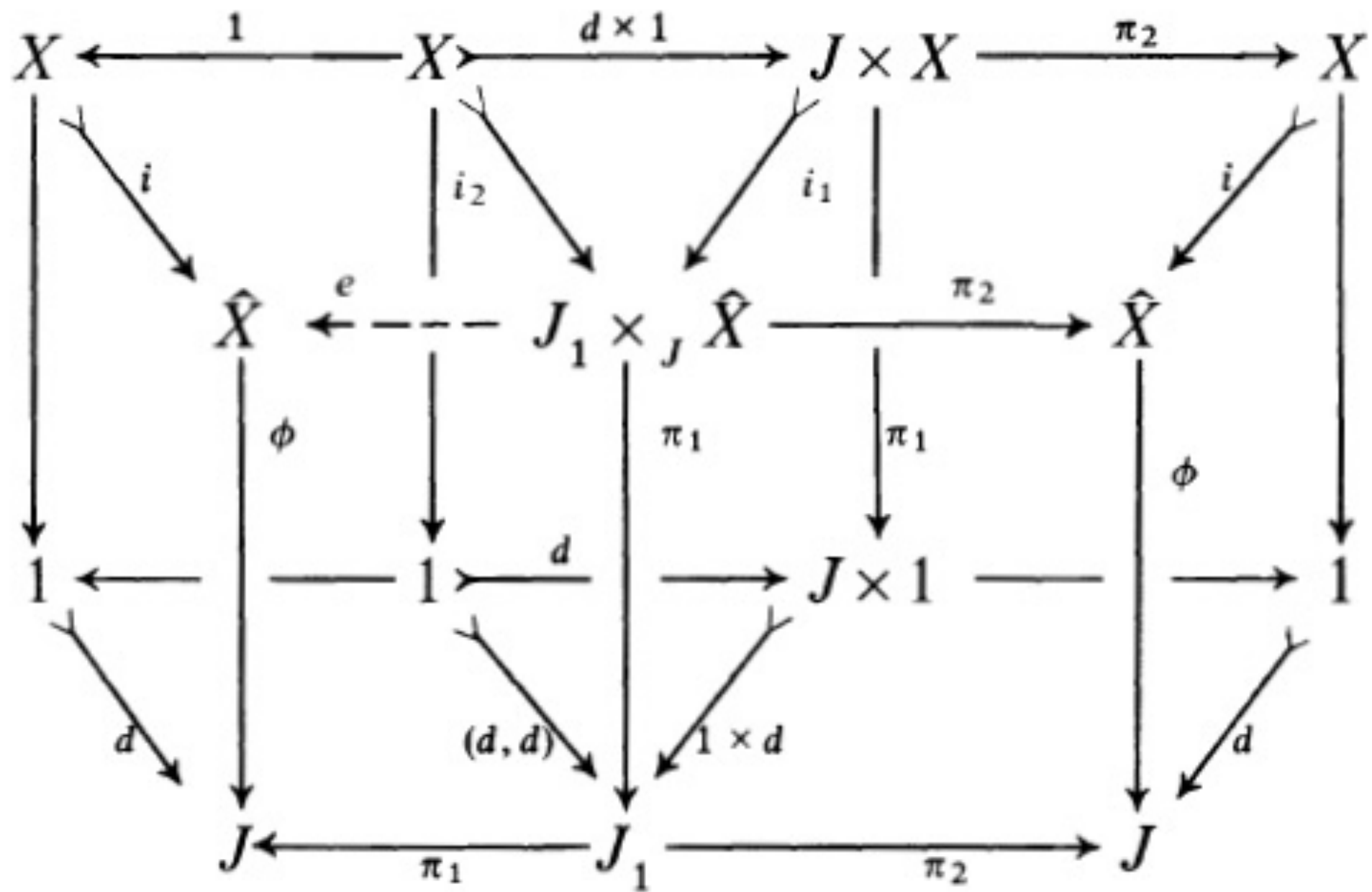
- ▶ P1: $\forall e, e \sqsubseteq e^*$ **EXTENSIVE**
- ▶ P2: $\forall e, e \triangleright e^* \sqsubseteq e^*$ and $e^* \triangleright e \sqsubseteq e^*$ **FOLDING**
- ▶ P3: $\forall e, (e^*)^* = e^*$ **IDEMPOTENT**
- ▶ P4: $\forall e, f, (e \sqcup f)^* = e^* \sqcup f^*$ **DISTRIBUTIVE**
- ▶ P5: $\forall e, I \sqsubseteq e^*$ **"SIMPLE"**

DESIDERATA FOR ITERATED EFFECTS: X^*

- ▶ P1: $\forall e, e \sqsubseteq e^*$ **EXTENSIVE**
 - ▶ P2: $\forall e, e \triangleright e^* \sqsubseteq e^*$ and $e^* \triangleright e \sqsubseteq e^*$ **FOLDING**
 - ▶ P3: $\forall e, (e^*)^* = e^*$ **IDEMPOTENT**
 - ▶ P4: $\forall e, f, (e \sqcup f)^* = e^* \sqcup f^*$ **DISTRIBUTIVE**
 - ▶ P5: $\forall e, I \sqsubseteq e^*$ **"SIMPLE"**
- } Hand-IDed by Flanagan & Qadeer

DESIDERATA FOR ITERATED EFFECTS: X^*

- ▶ P1: $\forall e, e \sqsubseteq e^*$ **EXTENSIVE**
 - ▶ P2: $\forall e, e \triangleright e^* \sqsubseteq e^*$ and $e^* \triangleright e \sqsubseteq e^*$ **FOLDING**
 - ▶ P3: $\forall e, (e^*)^* = e^*$ **IDEMPOTENT**
 - ▶ P4: $\forall e, f, (e \sqcup f)^* = e^* \sqcup f^*$ **DISTRIBUTIVE**
 - ▶ P5: $\forall e, I \sqsubseteq e^*$ **"SIMPLE"**
- Hand-IDed by
Flanagan & Qadeer
- Byproduct of $I = \perp$
in Flanagan and Qadeer



BRING ON THE MONADS!

THE SEQUENTIAL SEMANTICS OF PRODUCER EFFECT SYSTEMS

- ▶ Ross Tate, POPL 2013
- ▶ Derived effectoids: algebraic structure with sequencing, “subeffecting”
 - ▶ Non-deterministic sequencing operation
 - ▶ Coherence condition ~ “non-determinism respects subeffects”
- ▶ Every effect quantale induces an effectoid
 - ▶ Effectoids lack an explicit join
- ▶ Many (most reasonable) effectoids induce an effect quantale

PARAMETRIC EFFECT MONADS AND SEMANTICS OF EFFECT SYSTEMS

- ▶ Shin-ya Katsumata, POPL 2014
- ▶ Index a monad by an algebra for sequencing: a partially-ordered monoid
- ▶ Now called “graded monads”
- ▶ “Most of the time” equivalent to effectoids
- ▶ Every effect quantale induces a graded monad
- ▶ Most partially-ordered monoids induce an effect quantale

EFFECT SYSTEMS REVISITED — CONTROL-FLOW ALGEBRA AND SEMANTICS

- ▶ Mycroft, Orchard, & Petricek, *Semantics, Logics, and Calculi*, 2016
- ▶ Extend graded monads to graded joinads: index by a *joinoid* rather than a *po-monoid*
 - ▶ *monoid* + *parallel composition* + *ordered-conditional* $?(-,-,-)$
 - ▶ $?(I,-,-)$ induces a form of join
- ▶ Similar, but weaker equations to effect quantales (only right distributive laws for $?(-,-,-)$)
- ▶ Every *total* effect quantale induces a joinoid (w/ degenerate parallelism)
- ▶ Joinoids can model control effects (effect quantales can't)