An algebraic framework to reason about concurrency

Alexandra Silva

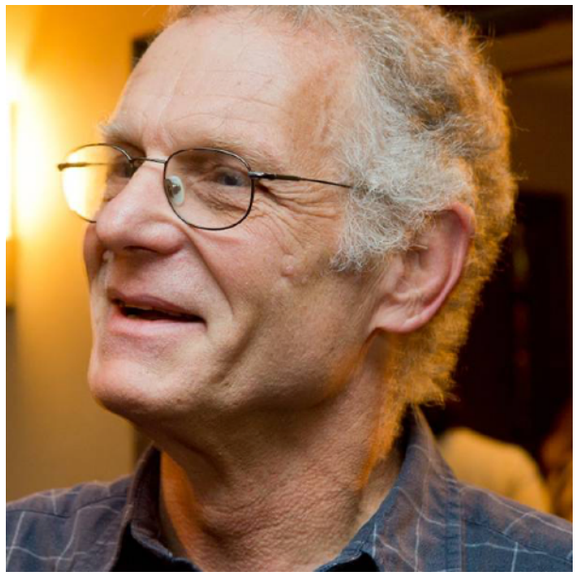
1

UCL ENGINEERING
Change the world

UCL

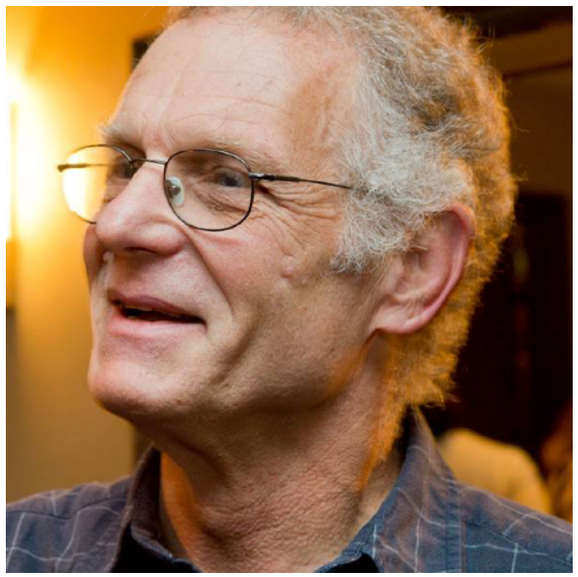
Context

Context



Kleene Algebra as an algebraic foundation for program verification

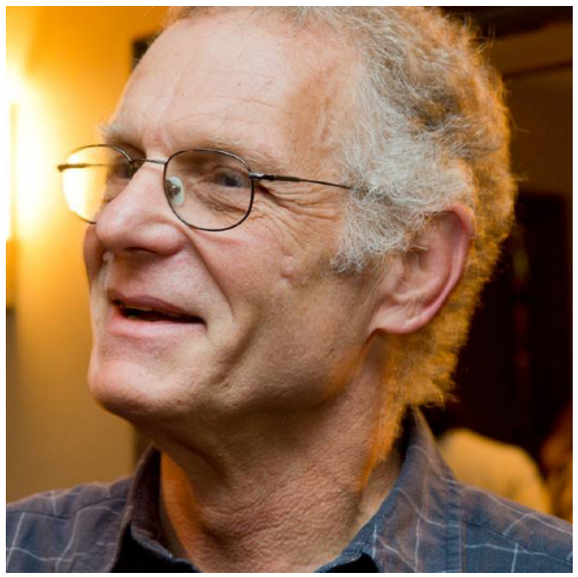
Context



Kleene Algebra as an algebraic foundation for program verification

Completeness

Context

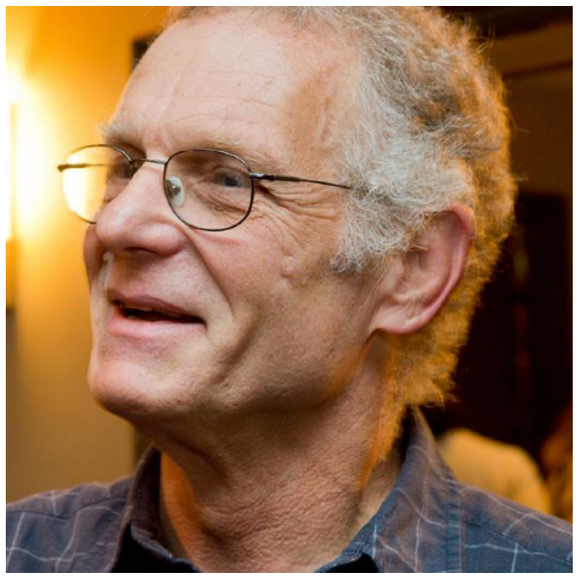


Kleene Algebra as an algebraic foundation for program verification

Completeness

Equivalence

Context



Kleene Algebra as an algebraic foundation for program verification

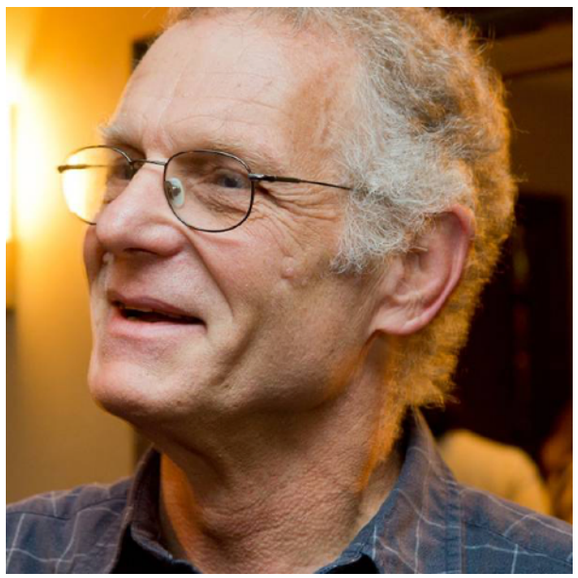
Completeness

Equivalence

Applications

Program Schematology
Compiler Optimization
and much more!

Context



Kleene Algebra as an algebraic foundation for program verification

Completeness

Equivalence

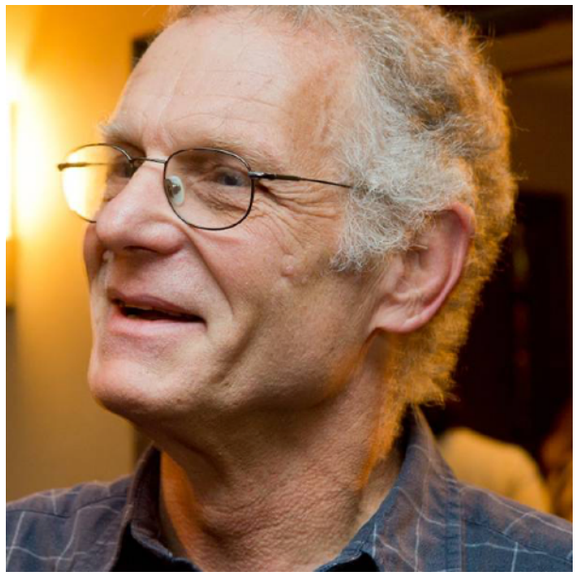
Applications

Program Schematology
Compiler Optimization
and much more!

Extensions

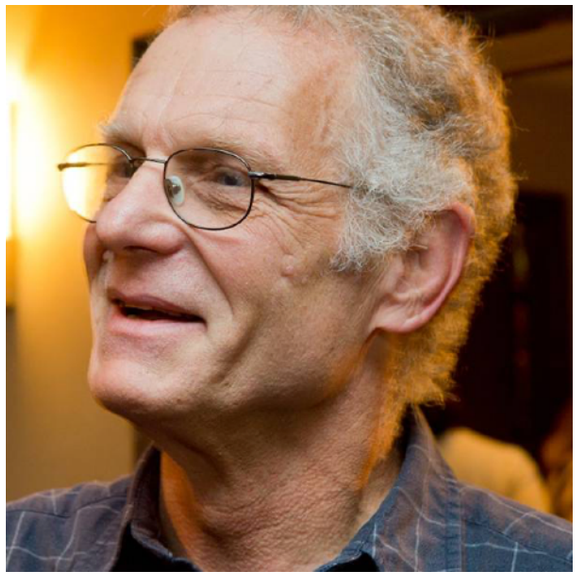
KAT, SKAT
KAT+B! ,…

Context



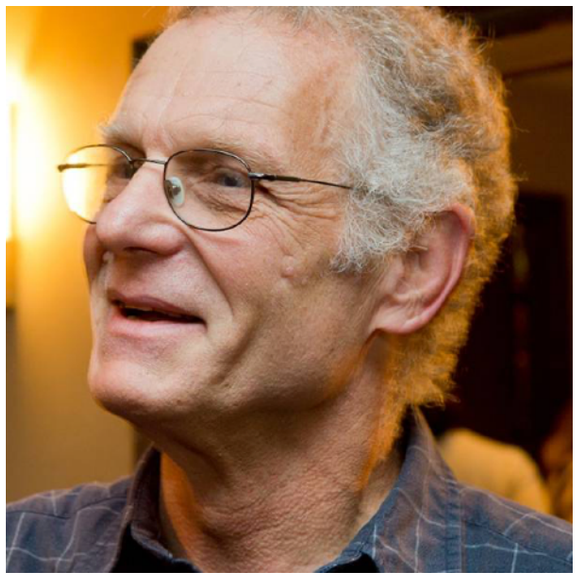
Kleene Algebra as an algebraic foundation for program verification

Context



Kleene Algebra as an algebraic foundation for program verification

Context



Kleene Algebra as an algebraic foundation for program verification

PROBABILISTIC NETKAT

Kleene Algebra: why?

Kleene Algebra: why?

Compositional Semantics

Kleene Algebra: why?

Compositional Semantics

Scalable verification

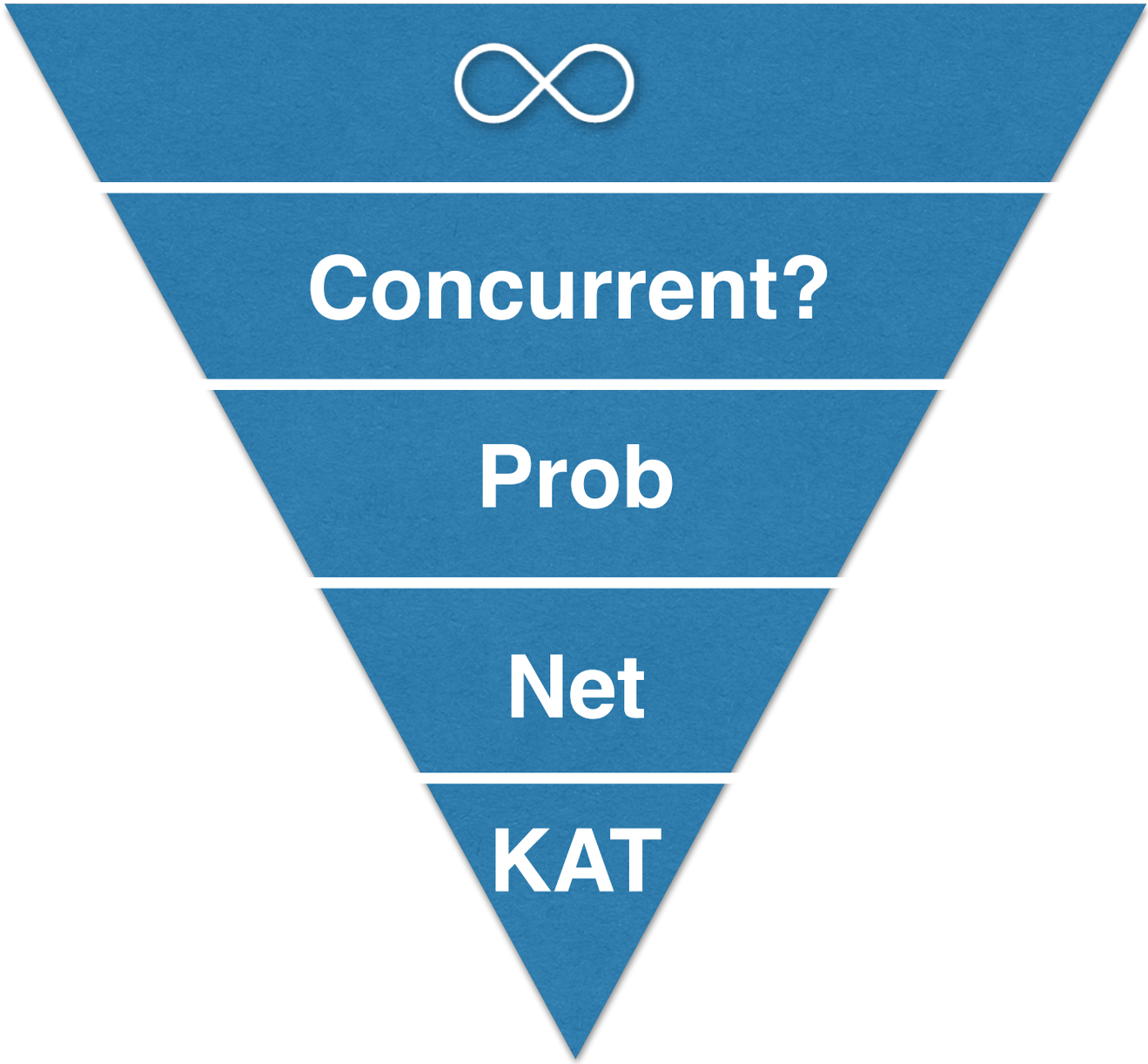
Kleene Algebra: why?

Compositional Semantics

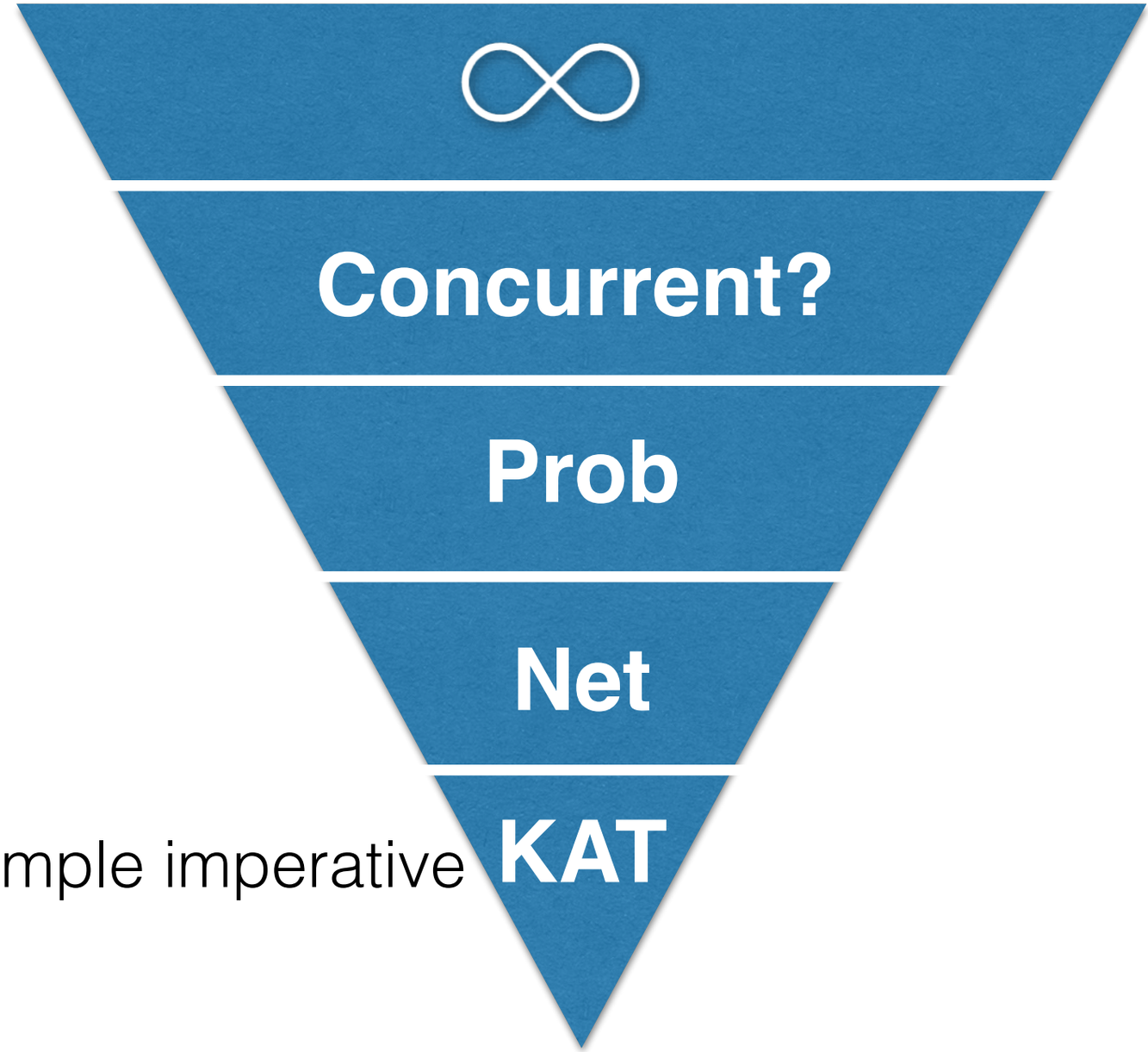
Scalable verification

Fundamental correspondence to automata and language theory

The KA(T) tower principle

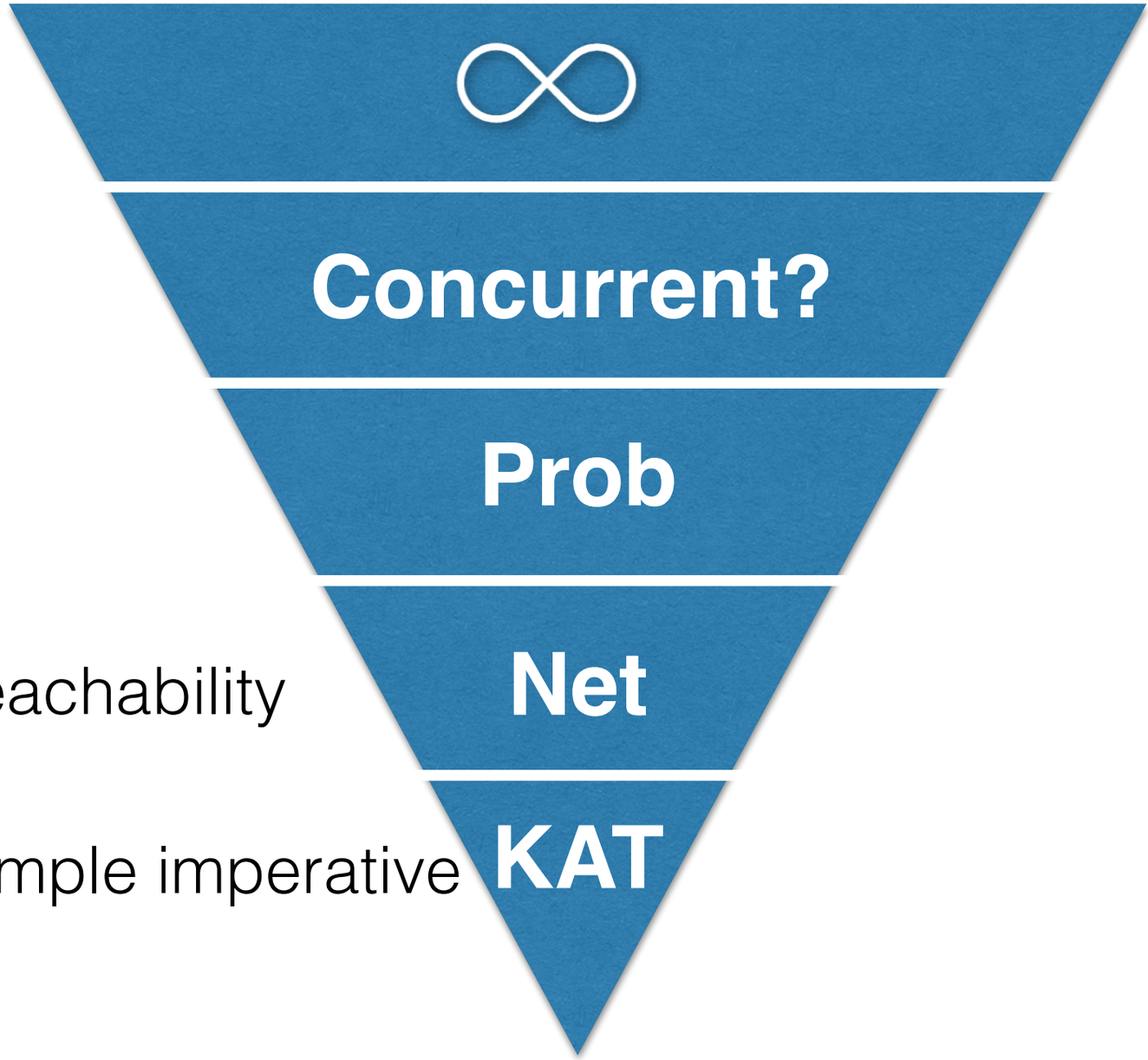


The KA(T) tower principle

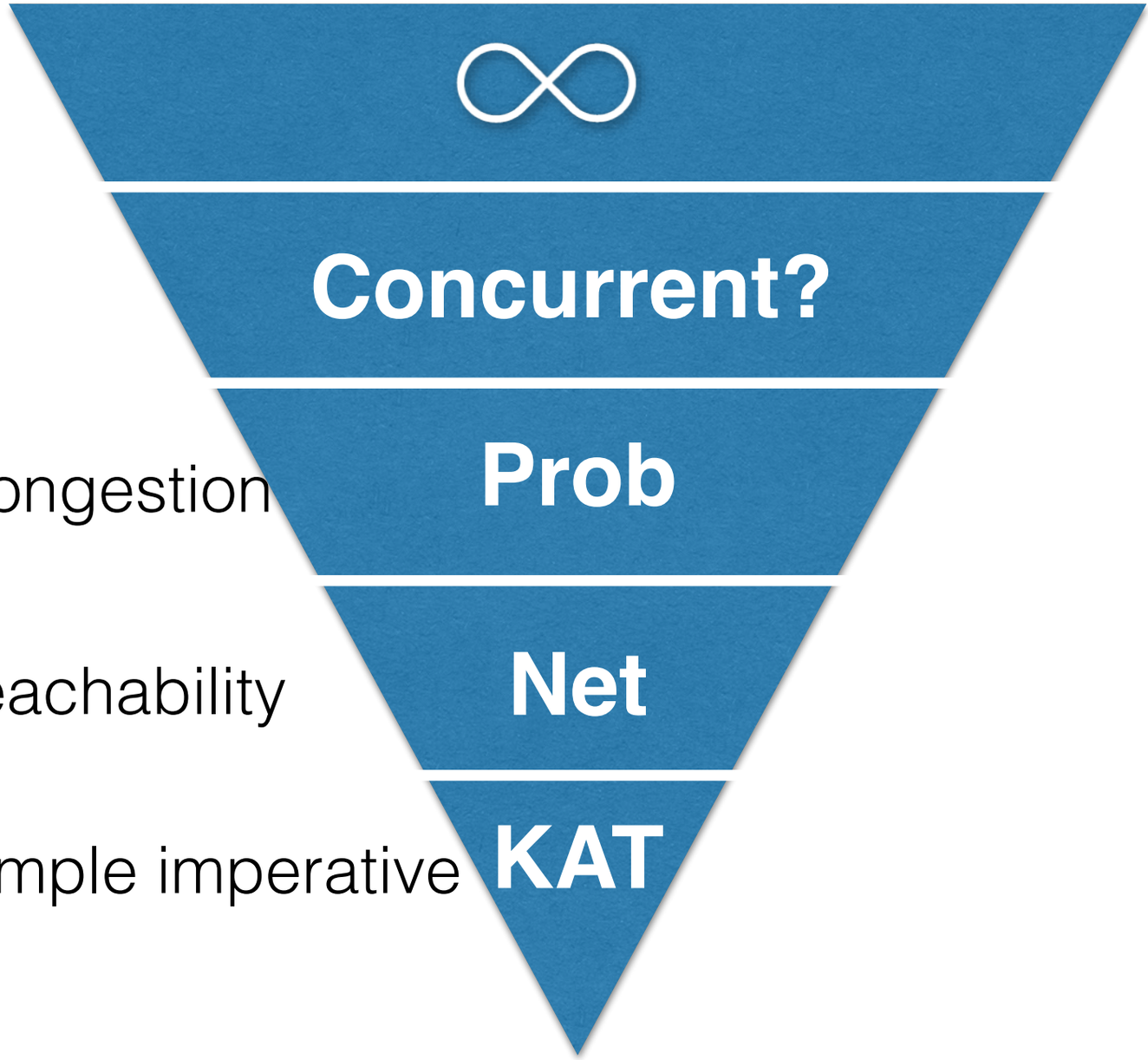


simple imperative

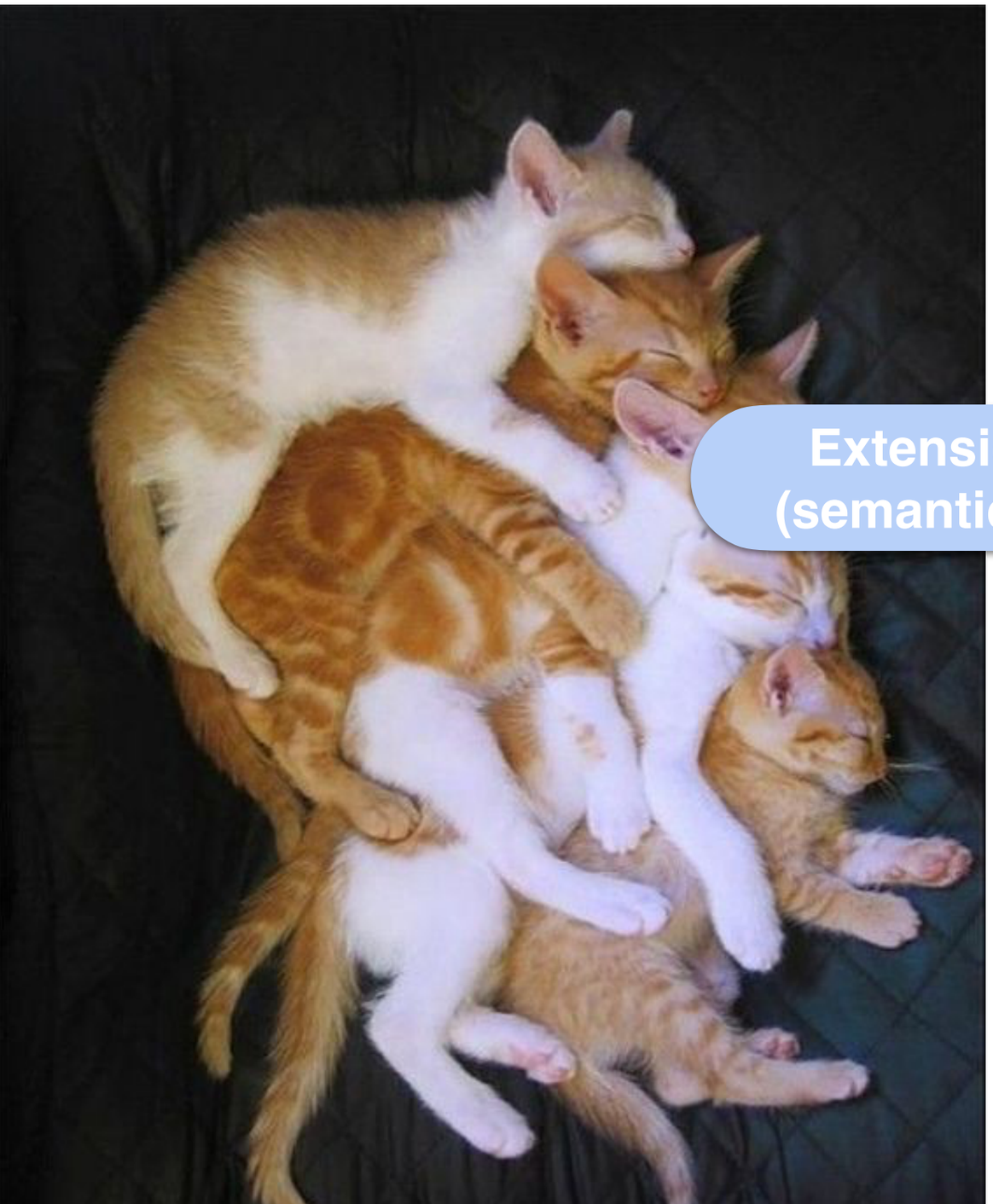
The KA(T) tower principle



The KA(T) tower principle



The KA(T) tower principle



Extensions conservative
(semantics and algorithms)

∞

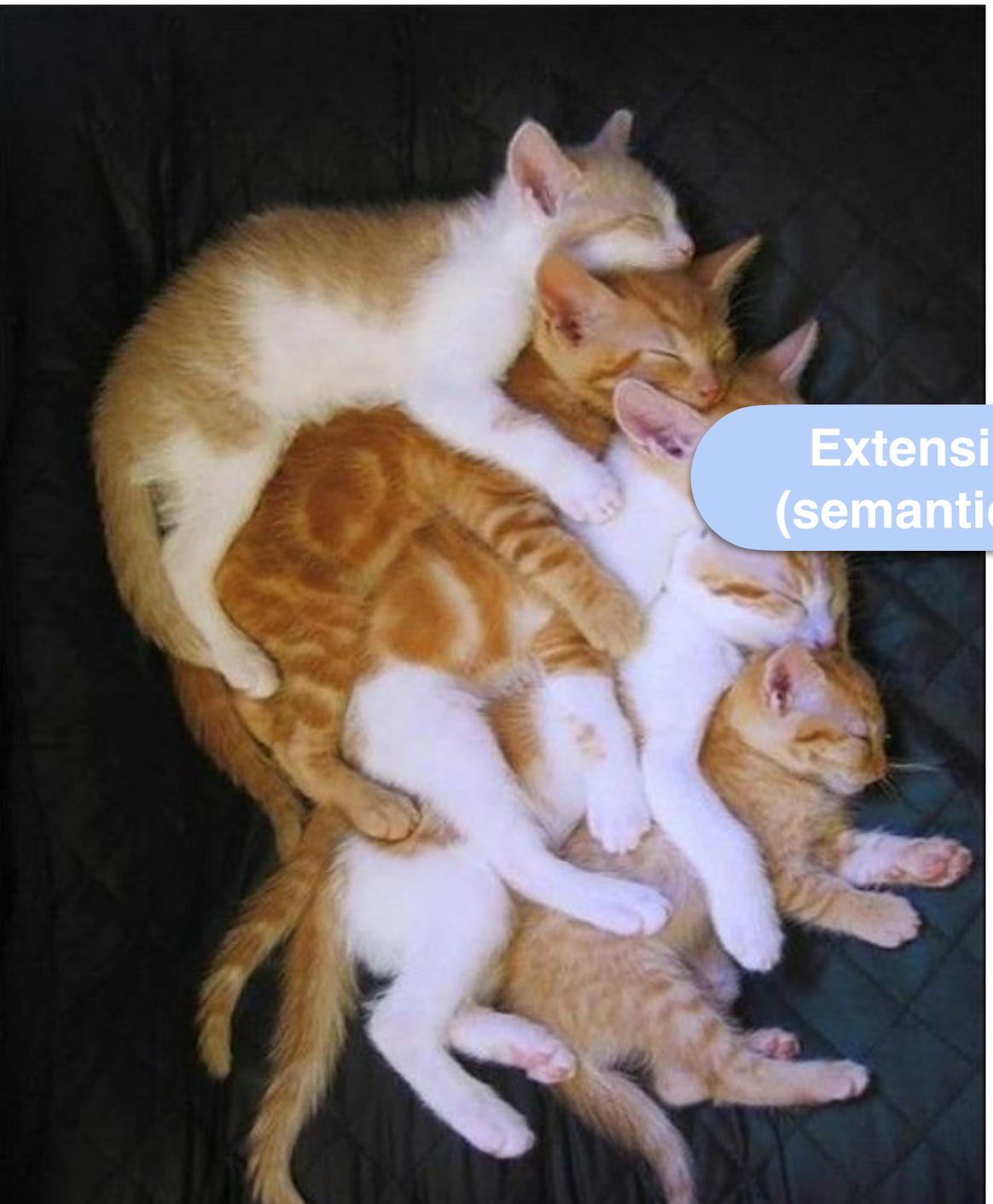
Concurrent?

Prob

Net

KAT

The KA(T) tower principle



Extensions conservative
(semantics and algorithms)

∞

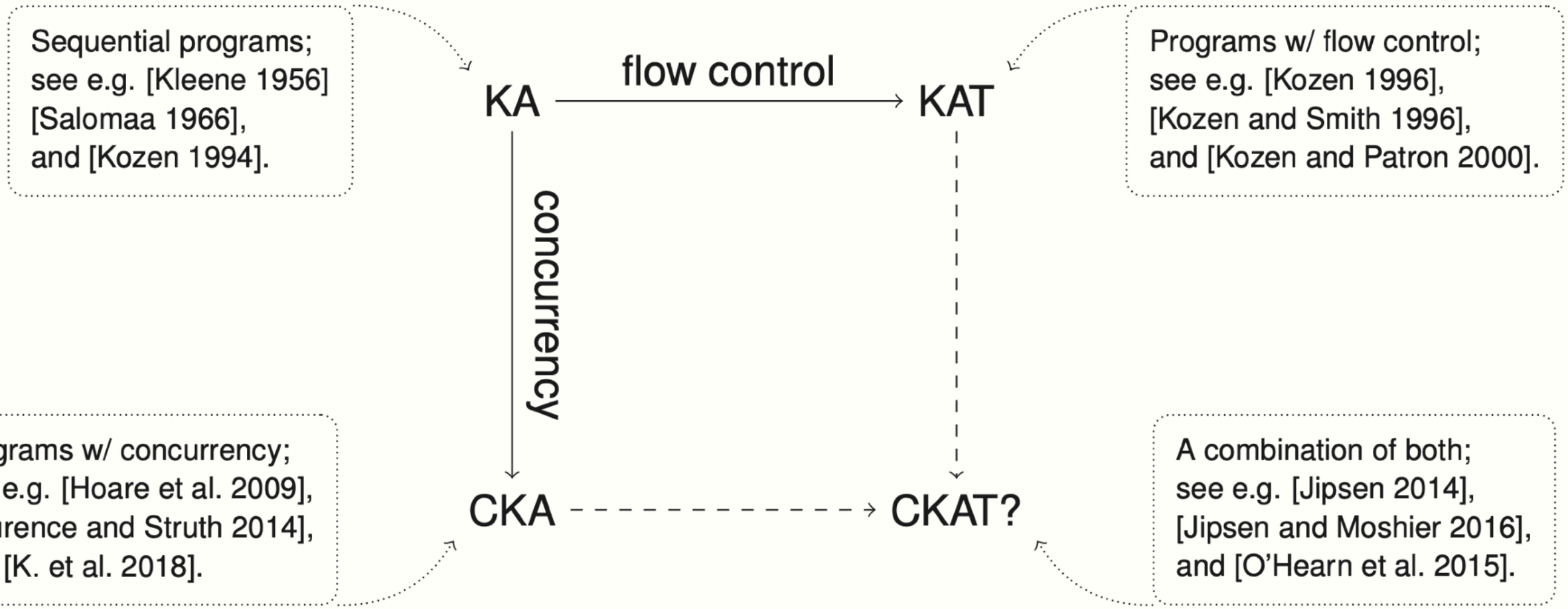
Concurrent?

Prob

Net

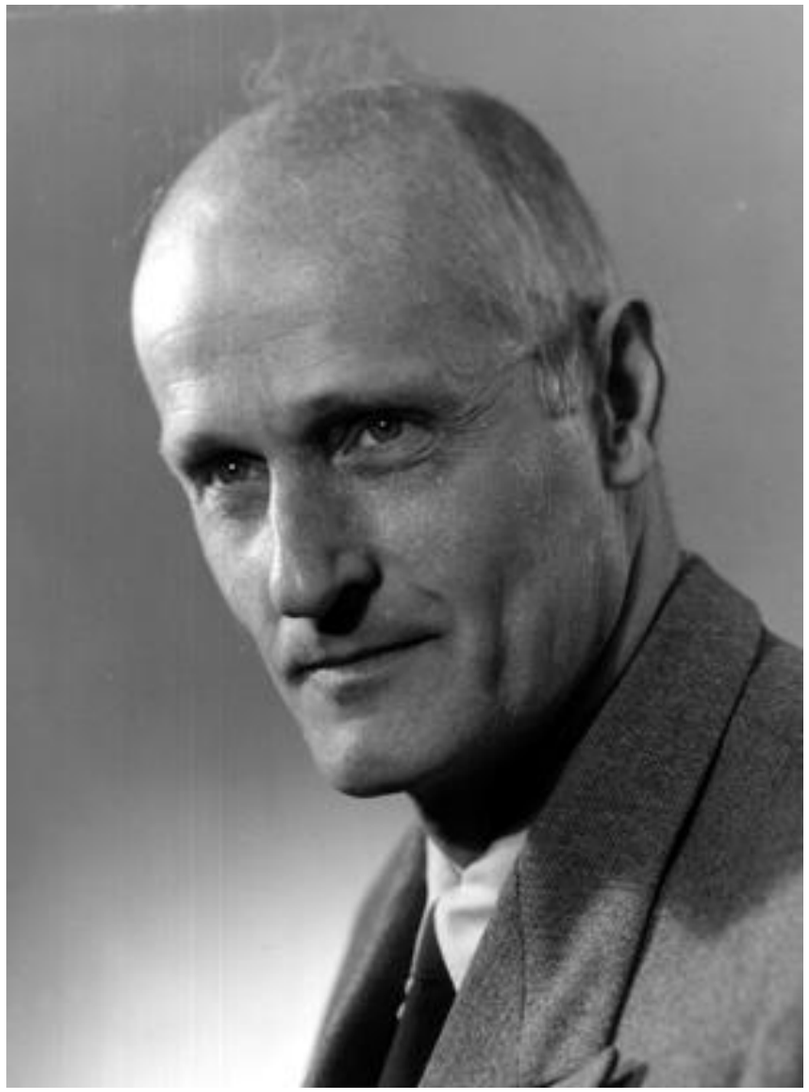
CKAT

This talk


Sequential programs;
see e.g. [Kleene 1956]
[Salomaa 1966],
and [Kozen 1994].
KA
flow control
KAT
Programs w/ flow control;
see e.g. [Kozen 1996],
[Kozen and Smith 1996],
and [Kozen and Patron 2000].
concurrency
Programs w/ concurrency;
see e.g. [Hoare et al. 2009],
[Laurence and Struth 2014],
and [K. et al. 2018].
CKA
CKAT?
A combination of both;
see e.g. [Jipsen 2014],
[Jipsen and Moshier 2016],
and [O'Hearn et al. 2015].

Let’s start from the beginning…

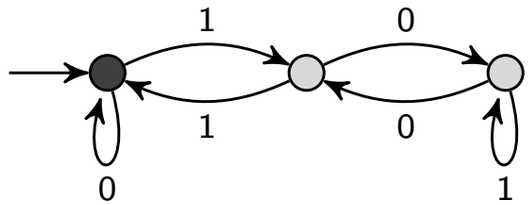
Kleene algebra - **KA**



Stephen Cole Kleene
(1909–1994)

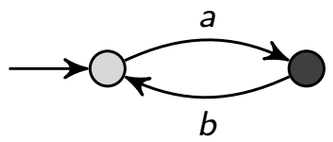
$(0 + 1(01^*0)^*1)^*$

{multiples of 3 in binary}



$(ab)^*a = a(ba)^*$

$\{a, aba, ababa, \ldots\}$



$(a + b)^* = a^*(ba^*)^*$

{all strings over $\{a, b\}$}

Kleene algebra - **KA**

program	expression
atomic action	$a, b, \ldots \in \Sigma$
abort execution	0
no-operation	1
nondeterministic choice	$e + f$
sequential composition	$e \cdot f$
repetition	e^*

Kleene algebra - **KA**

$$e + 0 \equiv e \qquad e + e \equiv e \qquad e + f \equiv f + e \qquad e + (f + g) \equiv (e + f) + g$$

Kleene algebra - **KA**

$$e + 0 \equiv e \qquad e + e \equiv e \qquad e + f \equiv f + e \qquad e + (f + g) \equiv (e + f) + g$$

$$e \cdot 0 \equiv 0 \equiv 0 \cdot e \qquad e \cdot 1 \equiv e \equiv 1 \cdot e \qquad e \cdot (f \cdot g) \equiv (e \cdot f) \cdot g$$

Kleene algebra - **KA**

$$e + 0 \equiv e \qquad e + e \equiv e \qquad e + f \equiv f + e \qquad e + (f + g) \equiv (e + f) + g$$

$$e \cdot 0 \equiv 0 \equiv 0 \cdot e \qquad e \cdot 1 \equiv e \equiv 1 \cdot e \qquad e \cdot (f \cdot g) \equiv (e \cdot f) \cdot g$$

$$e \cdot (f + g) \equiv e \cdot f + e \cdot g \qquad (e + f) \cdot g \equiv e \cdot g + f \cdot g$$

Kleene algebra - **KA**

$$e + 0 \equiv e \qquad e + e \equiv e \qquad e + f \equiv f + e \qquad e + (f + g) \equiv (e + f) + g$$

$$e \cdot 0 \equiv 0 \equiv 0 \cdot e \qquad e \cdot 1 \equiv e \equiv 1 \cdot e \qquad e \cdot (f \cdot g) \equiv (e \cdot f) \cdot g$$

$$e \cdot (f + g) \equiv e \cdot f + e \cdot g \qquad (e + f) \cdot g \equiv e \cdot g + f \cdot g$$

$$1 + e \cdot e^* \equiv e^* \qquad e \cdot f + g \leqq f \implies e^* \cdot g \leqq f$$

Kleene algebra - **KA**

$$e + 0 \equiv e \qquad e + e \equiv e \qquad e + f \equiv f + e \qquad e + (f + g) \equiv (e + f) + g$$

$$e \cdot 0 \equiv 0 \equiv 0 \cdot e \qquad e \cdot 1 \equiv e \equiv 1 \cdot e \qquad e \cdot (f \cdot g) \equiv (e \cdot f) \cdot g$$

$$e \cdot (f + g) \equiv e \cdot f + e \cdot g \qquad (e + f) \cdot g \equiv e \cdot g + f \cdot g$$

$$1 + e \cdot e^* \equiv e^* \qquad e \cdot f + g \leqq f \implies e^* \cdot g \leqq f$$

Theorem [Kozen'90]

KA axioms are sound and complete

Equivalence

Via Completeness
using KA axioms

Via Kleene's Theorem
using automata

Near-linear algorithms
Hopcroft-Karp, Bisimulation up-to

Equivalence

Via Completeness
using KA axioms

Via Kleene's Theorem
using automata

Near-linear algorithms
Hopcroft-Karp, Bisimulation up-to

Equivalence is essential to many verification questions!

RESEARCH-ARTICLE FREE ACCESS

NetKAT: semantic foundations for networks

Authors: Carolyn Jane Anderson, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, David Walker Authors Info & Affiliations

Equivalence

Via Completeness
using KA axioms

Via Kleene's Theorem
using automata

Near-linear algorithms
Hopcroft-Karp, Bisimulation up-to

Equivalence is essential to many verification questions!

Efficient fragments are still being studied!

Equivalence

Via Completeness
using KA axioms

Via Kleene's Theorem
using automata

Near-linear algorithms
Hopcroft-Karp, Bisimulation up-to

Equivalence is essential to many verification questions!

Efficient fragments are still being studied!

ARTICLE OPEN ACCESS

Guarded Kleene algebra with tests: verification of uninterpreted programs in nearly linear time

Authors: Steffen Smolka, Nate Foster, Justin Hsu, Tobias Kappé, Dexter Kozen, Alexandra Silva Authors Info & Affiliations

KAT

$(K, B, +, \cdot, ^*, \bar{}, 0, 1), \quad B \subseteq K$

- $(K, +, \cdot, ^*, 0, 1)$ is a Kleene algebra
- $(B, +, \cdot, \bar{}, 0, 1)$ is a Boolean algebra
- $(B, +, \cdot, 0, 1)$ is a subalgebra of $(K, +, \cdot, 0, 1)$

- $p, q, r, \ldots$ range over K
- $a, b, c, \ldots$ range over B

KAT

KAT = simple imperative language

If b **then** p **else** q = b;p + !b;q

While b **do** p = (bp)*!b

Boolean algebra

KAT

KAT = simple imperative language

If b **then** p **else** q = b;p + !b;q

While b **do** p = (bp)*!b

KAT

KAT = simple imperative language

If b **then** p **else** q = b;p + !b;q

While b **do** p = (bp)*!b

$$[\![p]\!] \subseteq At \cdot (\Sigma \cdot At)^*$$

KAT

$$[\![p]\!] \subseteq At \cdot (\Sigma \cdot At)^*$$

If b **then** p **else** q = b;p + !b;q

While b **do** p = (bp)*!b

$$\{\alpha_b \mathsf{p} \beta, \alpha_{!b} \mathsf{p} \beta\}$$

$$\{\alpha_{!b}, \alpha_b \mathsf{p} \alpha_{!b}, \alpha_b \mathsf{p} \alpha_b \mathsf{p} \alpha_{!b}, \ldots\}$$

valuations
variables

Equivalence

Theorem [Kozen'96]

KA+BA axioms are sound and complete

Via Completeness
using KAT axioms

Via Kleene's Theorem
using *guarded* automata

Equivalence

Theorem [Kozen'96]

KA+BA axioms are sound and complete

Via Completeness
using KAT axioms

Via Kleene's Theorem
using *guarded* automata

$$\{b\}\, p\, \{c\} \overset{\triangle}{\iff} bp \leqslant pc$$
$$\iff bp = bpc$$
$$\iff bp\bar{c} = 0$$

Concurrent programs

initial state: x = 0, y = 0	
T0	T1
a: x ← 1;	c: y ← 1;
b: r1 ← y;	d: r2 ← x;
assert: $r_1 \neq 0$ xor $r_2 \neq 0$	

Concurrent programs

initial state: x = 0, y = 0	
T0	T1
a: x ← 1;	c: y ← 1;
b: r1 ← y;	d: r2 ← x;
assert: $r_1 \neq 0$ xor $r_2 \neq 0$	

Can we reason about these programs (co)algebraically?

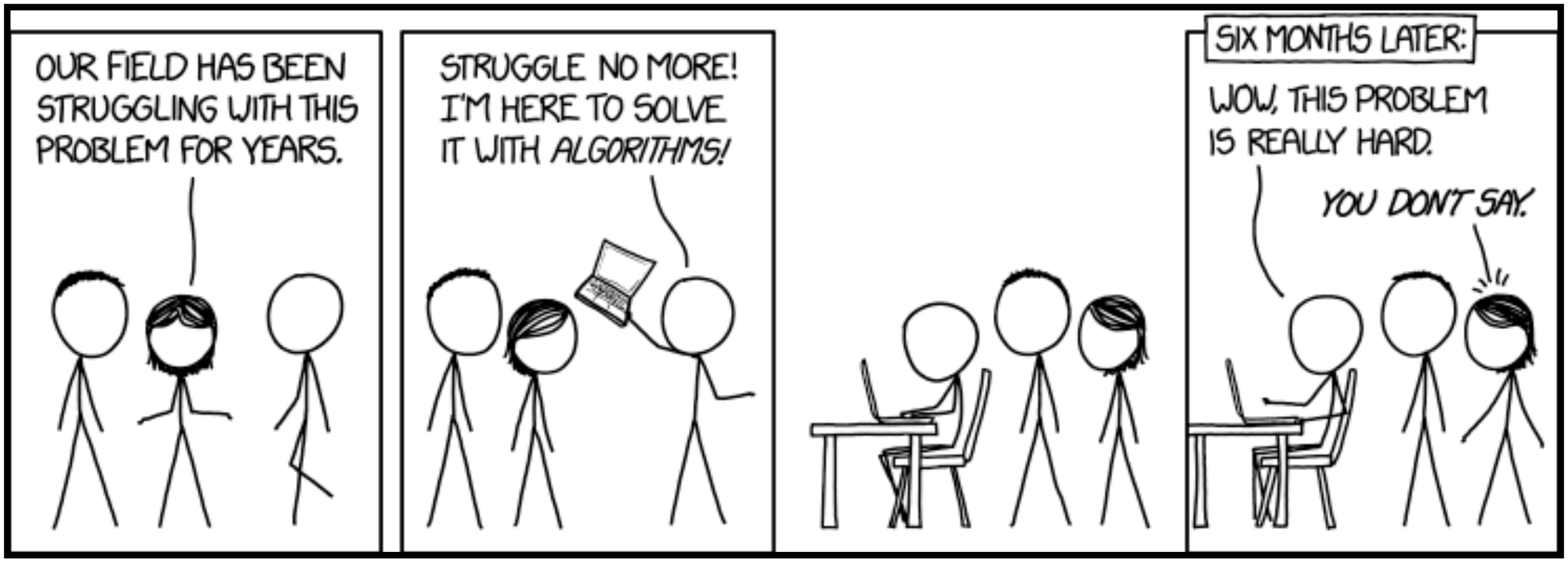
Concurrent programs

initial state: x = 0, y = 0	
T0	T1
a: x ← 1;	c: y ← 1;
b: r1 ← y;	d: r2 ← x;
assert: $r_1 \neq 0$ xor $r_2 \neq 0$	

Can we reason about these programs (co)algebraically?

$$(x = 0 \wedge y = 0) \cdot (T0 \parallel T1) \leq \neg(r1 = 0 \wedge r2 = 0)$$

Concurrency



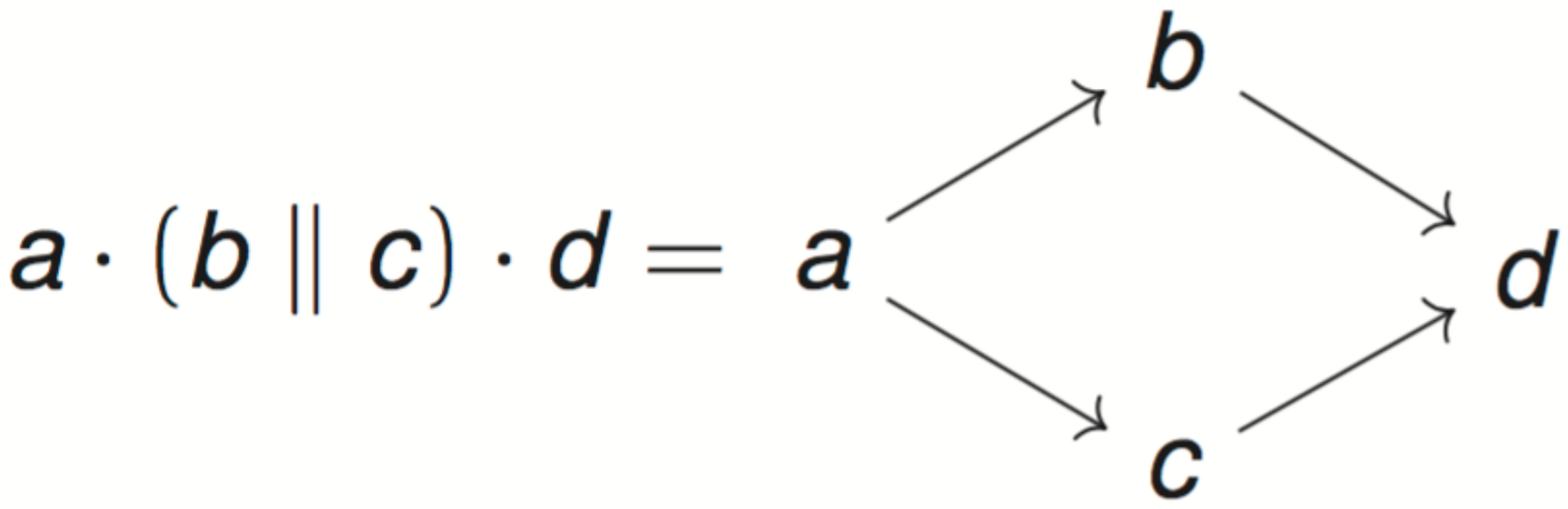
Concurrent Kleene Algebra
(Hoare, Moeller, Struth, Wehrman'09)

$$p \;||\; q$$

Concurrent Kleene Algebra
(Hoare, Moeller, Struth, Wehrman'09)

$$p \mid\mid q$$

Semantics = languages of PomSets



Concurrent Kleene Algebra
(Hoare, Moeller, Struth, Wehrman'09)

$$p \;||\; q$$

Kleene Theorem

Axiomatisation

Decision Procedure

Concurrent Kleene Algebra
(Hoare, Moeller, Struth, Wehrman'09)

$$p \;||\; q$$

Kleene Theorem

Axiomatisation

Decision Procedure

Concurrent Kleene Algebra
(Hoare, Moeller, Struth, Wehrman'09)

$$e \parallel f \equiv f \parallel e \qquad e \parallel (f \parallel g) \equiv (e \parallel f) \parallel g$$

$$e \parallel 1 \equiv e \qquad e \parallel 0 \equiv 0 \qquad e \parallel (f + g) \equiv e \parallel f + e \parallel g$$

Concurrent Kleene Algebra
(Hoare, Moeller, Struth, Wehrman'09)

$$e \parallel f \equiv f \parallel e \qquad e \parallel (f \parallel g) \equiv (e \parallel f) \parallel g$$

$$e \parallel 1 \equiv e \qquad e \parallel 0 \equiv 0 \qquad e \parallel (f + g) \equiv e \parallel f + e \parallel g$$

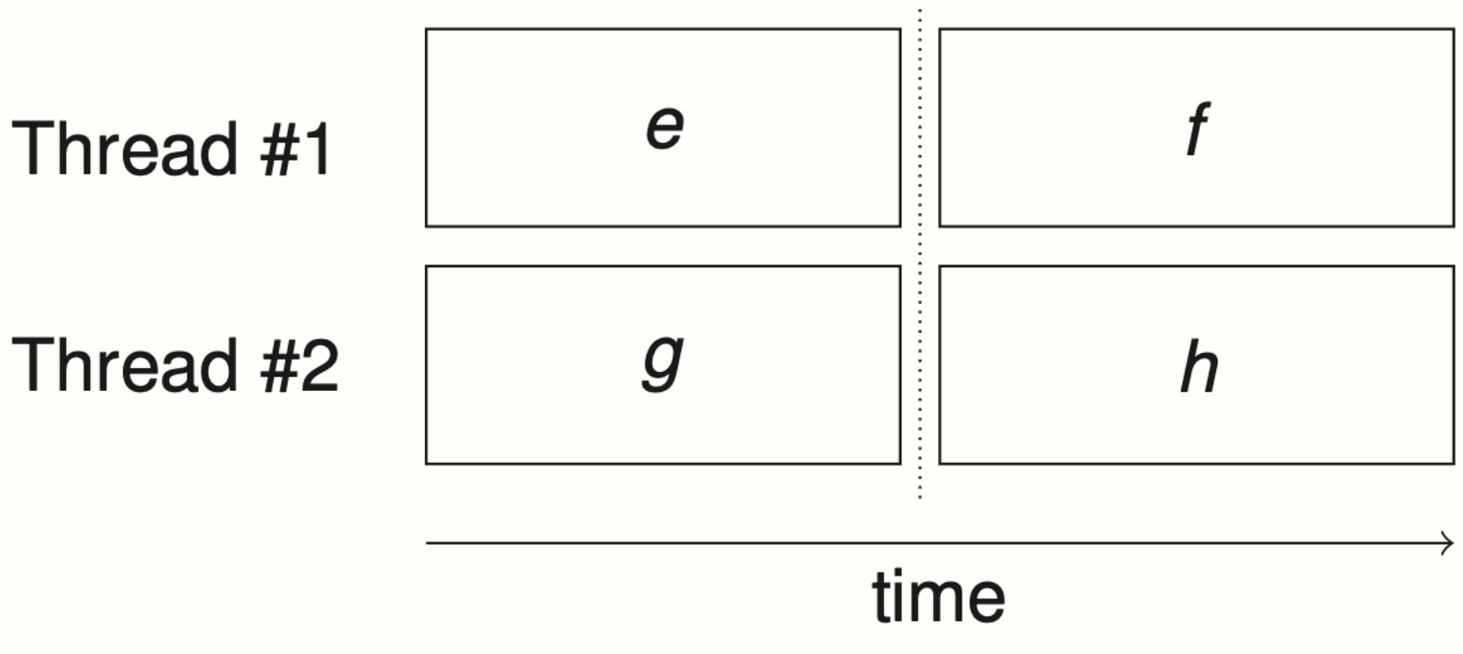
$$(e \parallel g) \cdot (f \parallel h) \leqq (e \cdot f) \parallel (g \cdot h). \qquad \textbf{Exchange law}$$

Concurrent Kleene Algebra
(Hoare, Moeller, Struth, Wehrman'09)

$$e \parallel f \equiv f \parallel e \qquad e \parallel (f \parallel g) \equiv (e \parallel f) \parallel g$$

$$e \parallel 1 \equiv e \qquad e \parallel 0 \equiv 0 \qquad e \parallel (f + g) \equiv e \parallel f + e \parallel g$$

$$(e \parallel g) \cdot (f \parallel h) \leqq (e \cdot f) \parallel (g \cdot h). \qquad \textbf{Exchange law}$$

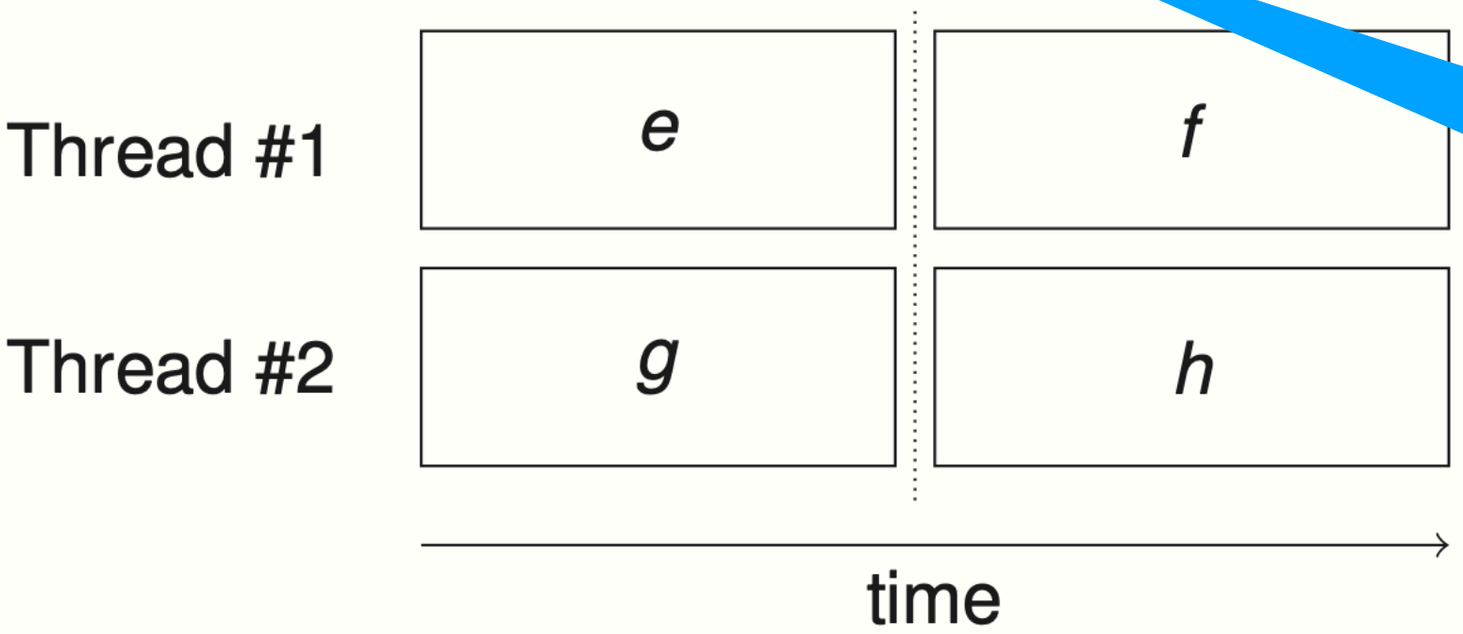


Concurrent Kleene Algebra
(Hoare, Moeller, Struth, Wehrman'09)

$$e \parallel f \equiv f \parallel e \qquad e \parallel (f \parallel g) \equiv (e \parallel f) \parallel g$$

$$e \parallel 1 \equiv e \qquad e \parallel 0 \equiv 0 \qquad e \parallel (f + g) \equiv e \parallel f + e \parallel g$$

$$(e \parallel g) \cdot (f \parallel h) \leqq (e \cdot f) \parallel (g \cdot h). \qquad \textbf{Exchange law}$$



Interleaving as special case: p||q=p.q+q.p


GoNeCo

Kleene Theorem

Axiomatisation

Decision Procedure

GoNeCo

Brzozowski goes concurrent: a Kleene theorem for Pomset languages (CONCUR 2017)

Tobias Kappé, Paul Brunet, Bas Luttik, Alexandra Silva, and Fabio Zanasi

Kleene Theorem

Axiomatisation

Decision Procedure

Brzozowski goes concurrent: a Kleene theorem for Pomset languages (CONCUR 2017)

Tobias Kappé, Paul Brunet, Bas Luttik, Alexandra Silva, and Fabio Zanasi

Kleene Theorem

Axiomatisation

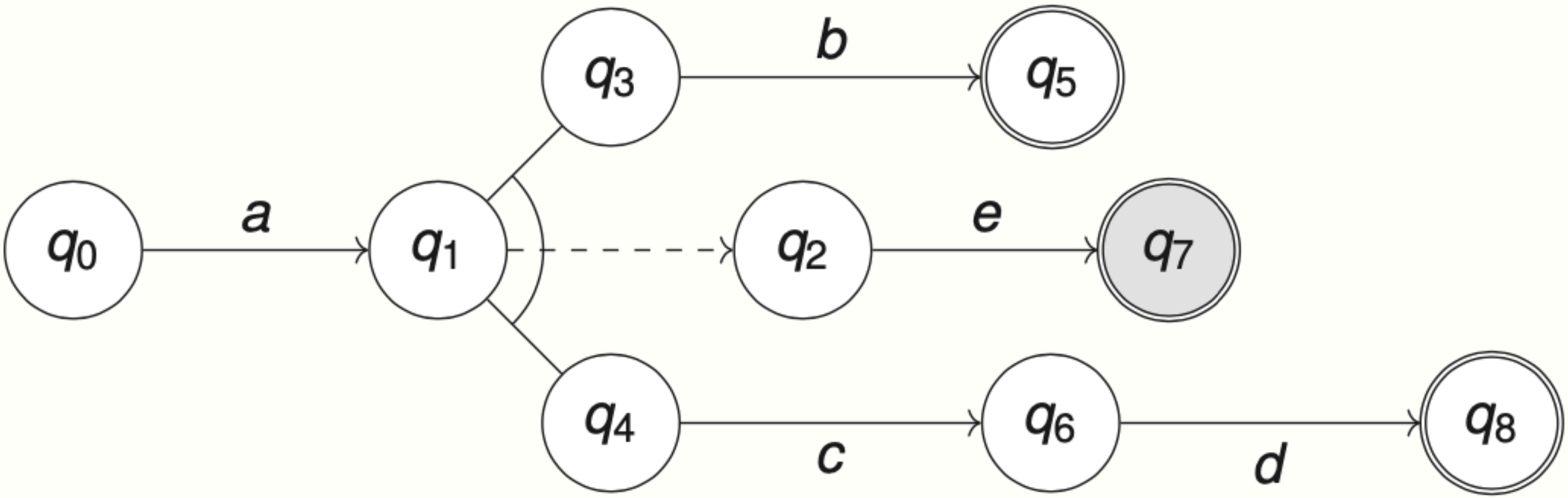
Decision Procedure

Concurrent Kleene Algebra: free model and completeness (ESOP 2018)

Tobias Kappé, Paul Brunet, Alexandra Silva, and Fabio Zanasi

Kleene Theorem

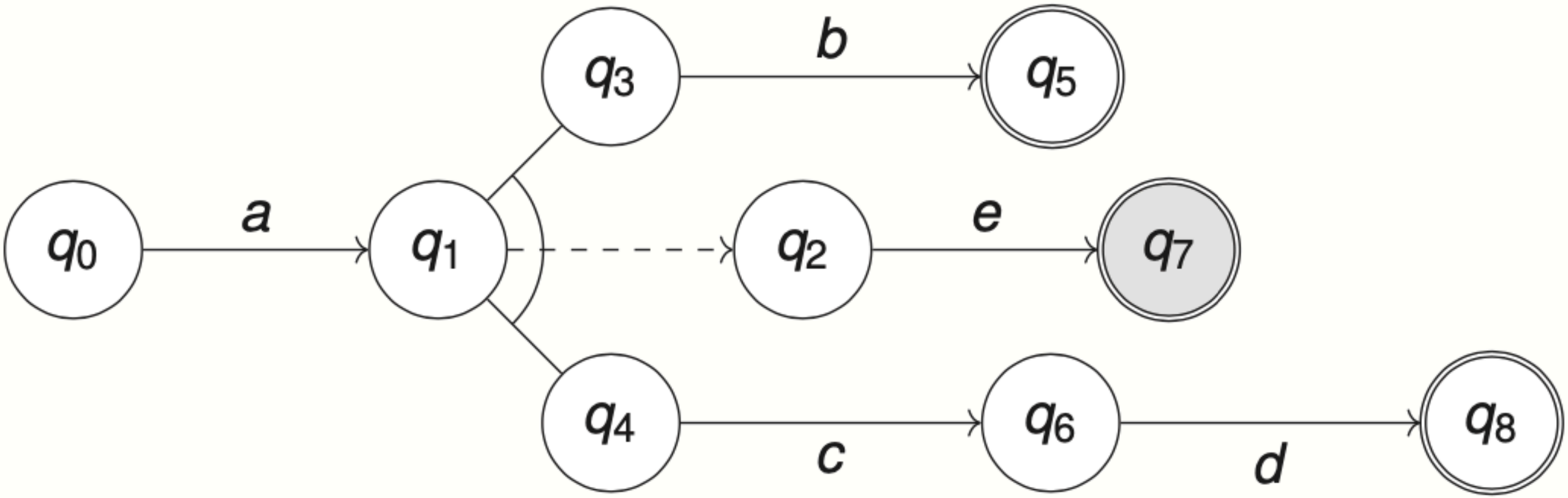
Fork Automata (Gischer'88)



$$a(b \parallel cd)e$$

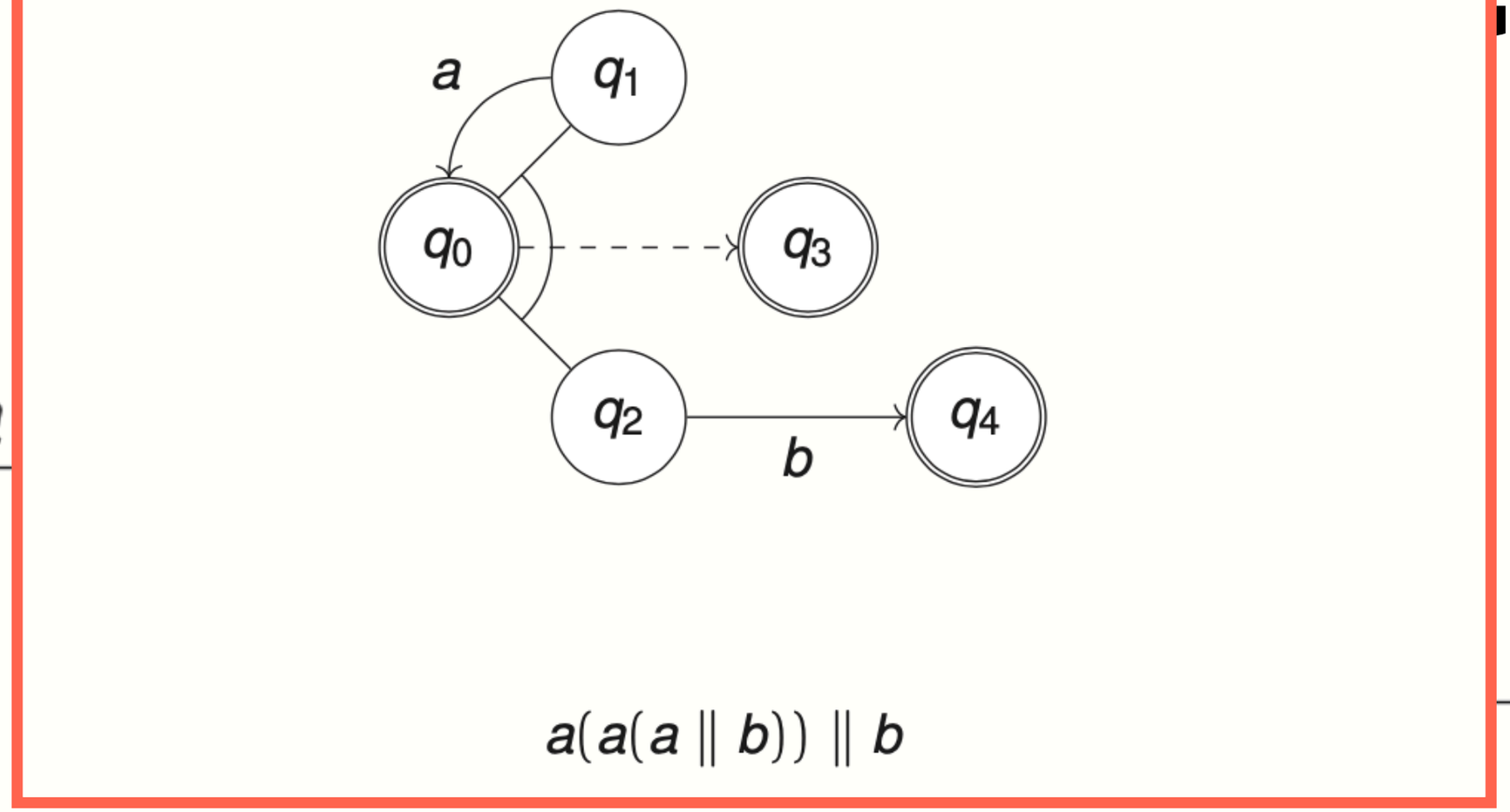
Kleene Theorem

Fork Automata (Gischer'88)



Kleene Theorem

ischer'88)



Brzozowski-like construction

Automata

Expressions

Requires restriction

CKA ✓

CKAT

CKA

next

GoNeCo

CNetKAT — soon

CKAT — next

CKA 

CKAT [Jipsen'14]

$$e, f ::= \underbrace{p \mid 0 \mid 1 \mid a \in \Sigma \mid e + f \mid e \cdot f \mid e^*}_{\text{KAT terms}} \mid e \parallel f$$

$$p, q ::= \bot \mid \top \mid t \in T \mid p \vee q \mid p \wedge q \mid \overline{p}$$

CKAT [Jipsen'14]

CKA terms

$$e, f ::= p \mid 0 \mid 1 \mid a \in \Sigma \mid e + f \mid e \cdot f \mid e^* \mid e \parallel f$$

KAT terms

$$p, q ::= \bot \mid \top \mid t \in T \mid p \vee q \mid p \wedge q \mid \overline{p}$$

The problem with CKAT

$$
\begin{aligned}
p \cdot e \cdot \overline{p} &\leqq (p \parallel e) \cdot \overline{p} \\
&\equiv (p \parallel e) \cdot (\overline{p} \parallel 1) \\
&\leqq (p \cdot \overline{p}) \parallel (e \cdot 1) \\
&\equiv (p \cdot \overline{p}) \parallel e
\end{aligned}
$$

The problem with CKAT

$$
\begin{aligned}
p \cdot e \cdot \overline{p} &\leqq (p \parallel e) \cdot \overline{p} \\
&\equiv (p \parallel e) \cdot (\overline{p} \parallel 1) \\
&\leqq (p \cdot \overline{p}) \parallel (e \cdot 1) \\
&\equiv (p \cdot \overline{p}) \parallel e \\
&\equiv (p \wedge \overline{p}) \parallel e \\
&\equiv \bot \parallel e \\
&\equiv 0 \parallel e \\
&\equiv 0
\end{aligned}
$$

The problem with CKAT

$$
\begin{aligned}
p \cdot e \cdot \overline{p} &\leqq (p \parallel e) \cdot \overline{p} \\
&\equiv (p \parallel e) \cdot (\overline{p} \parallel 1) \\
&\leqq (p \cdot \overline{p}) \parallel (e \cdot 1) \\
&\equiv (p \cdot \overline{p}) \parallel e \\
&\equiv (p \wedge \overline{p}) \parallel e \\
&\equiv \bot \parallel e \\
&\equiv 0 \parallel e \\
&\equiv 0
\end{aligned}
$$

The problem with CKAT

$$
\begin{aligned}
p \cdot e \cdot \overline{p} &\leqq (p \parallel e) \cdot \overline{p} \\
&\equiv (p \parallel e) \cdot (\overline{p} \parallel 1) \\
&\leqq (p \cdot \overline{p}) \parallel (e \cdot 1) \\
&\equiv (p \cdot \overline{p}) \parallel e \\
&\equiv (p \wedge \overline{p}) \parallel e \\
&\equiv \bot \parallel e \\
&\equiv 0 \parallel e \\
&\equiv 0
\end{aligned}
$$

The problem with CKAT

$$
\begin{aligned}
p \cdot e \cdot \overline{p} &\leqq (p \parallel e) \cdot \overline{p} \\
&\equiv (p \parallel e) \cdot (\overline{p} \parallel 1) \\
&\leqq (p \cdot \overline{p}) \parallel (e \cdot 1) \\
&\equiv (p \cdot \overline{p}) \parallel e \\
&\equiv (p \wedge \overline{p}) \parallel e \\
&\equiv \bot \parallel e \\
&\equiv 0 \parallel e \\
&\equiv 0
\end{aligned}
$$

Every test is an invariant of every program!

Crossroads

Non-congruence?

Drop $e \parallel 0 \equiv 0$?

Drop exchange law?

Crossroads

See also [Bergstra and Ponse 2011].

Crossroads

See also [Bergstra and Ponse 2011].

Crossroads

See also [Bergstra and Ponse 2011].

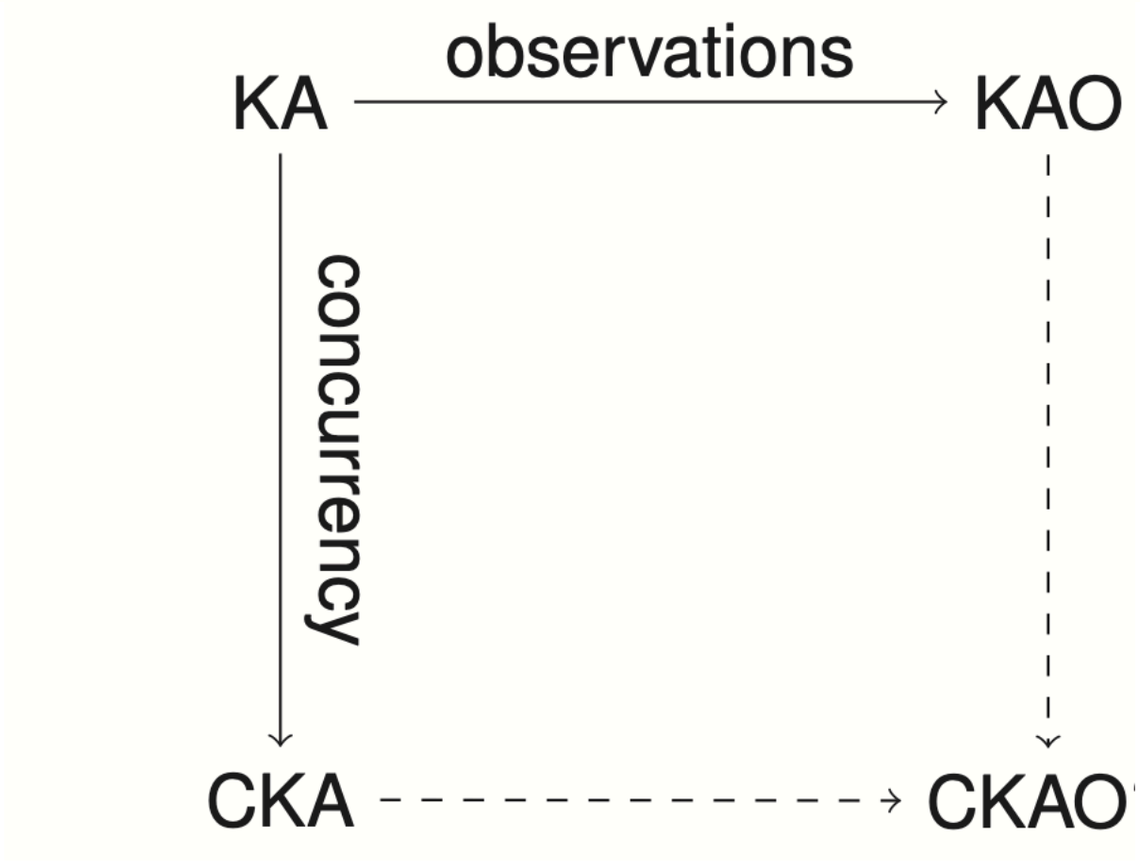
Crossroads

See also [Bergstra and Ponse 2011].

A new algebra

$$p \wedge q \equiv p \cdot q \rightsquigarrow p \wedge q \leqq p \cdot q$$

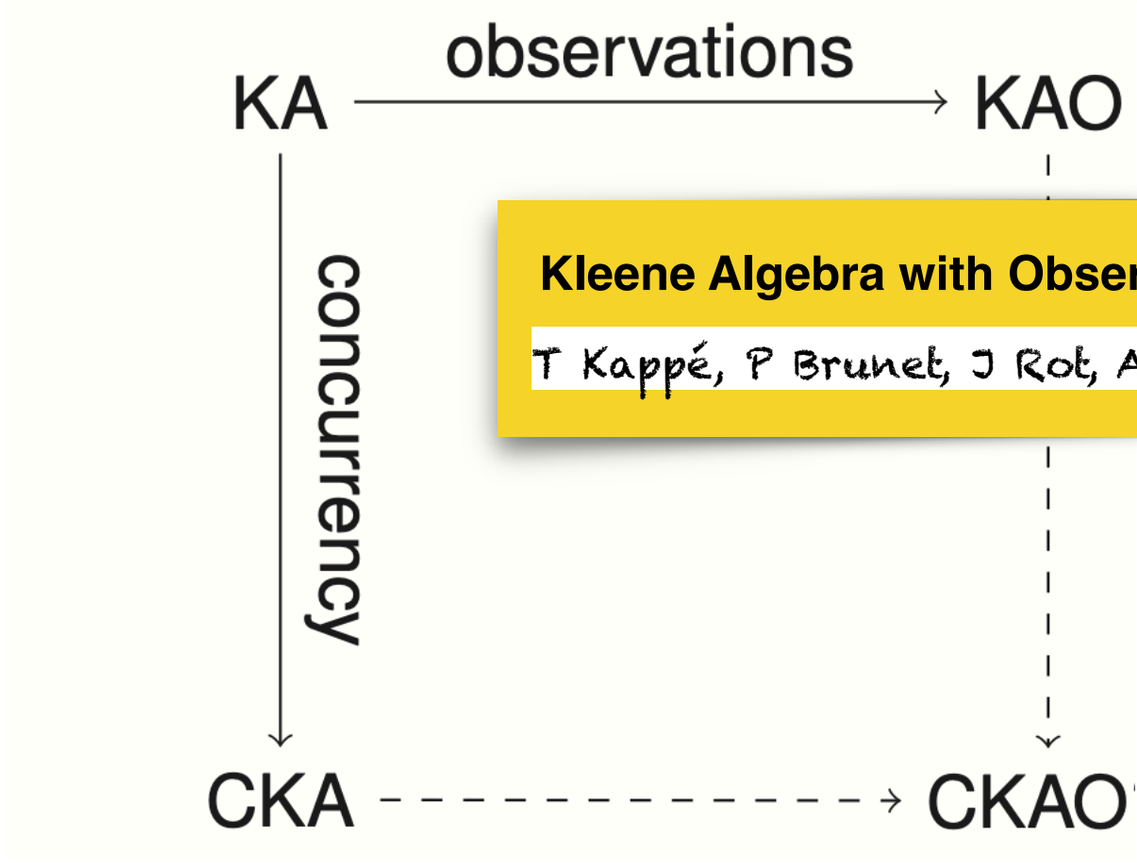
~~$1 \equiv \top$~~



A new algebra

$$p \wedge q \equiv p \cdot q \rightsquigarrow p \wedge q \leqq p \cdot q$$

~~$1 \equiv \top$~~



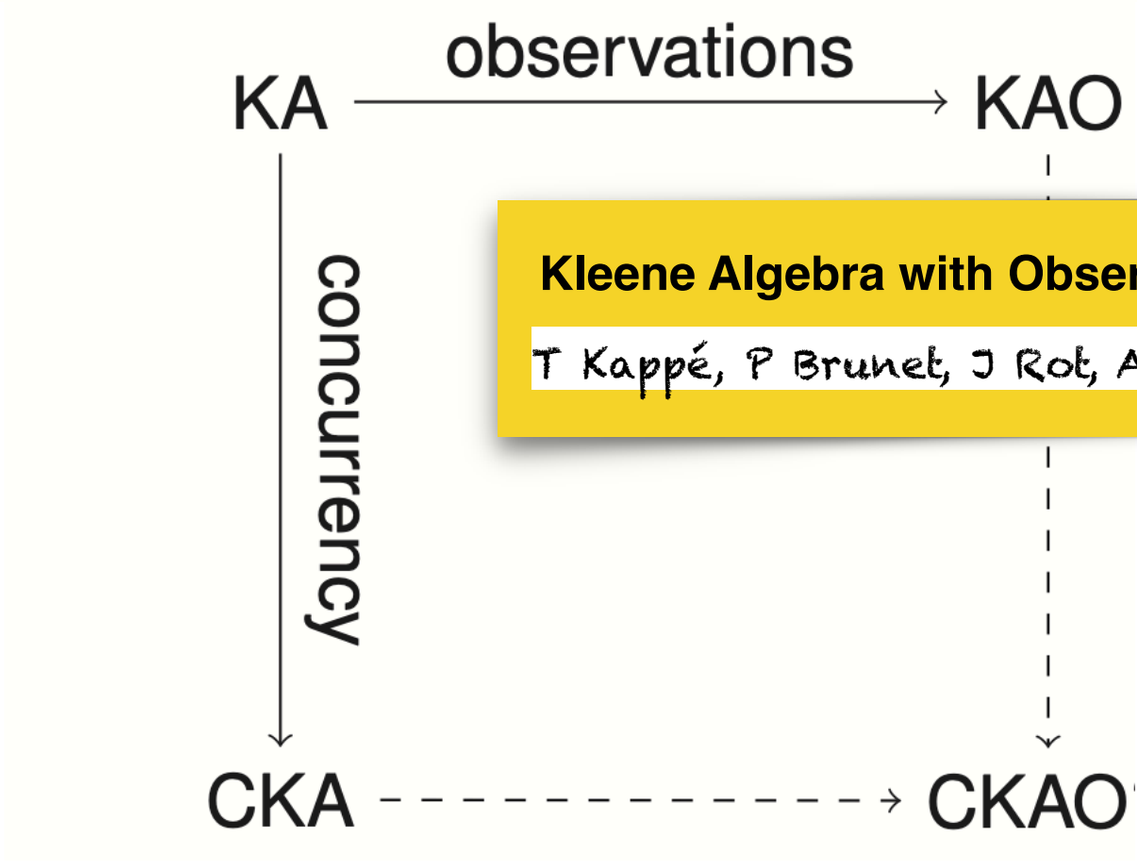
Kleene Algebra with Observations (CONCUR'19)

T Kappé, P Brunet, J Rot, A Silva, J Wagemaker, and F Zanasi

A new algebra

$$p \wedge q \equiv p \cdot q \rightsquigarrow p \wedge q \leqq p \cdot q$$

~~$1 \equiv \top$~~



Kleene Algebra with Observations (CONCUR'19)

T Kappé, P Brunet, J Rot, A Silva, J Wagemaker, and F Zanasi

Concurrent Kleene Algebra with Observations: from Hypotheses to Completeness (Fossacs'20)

T Kappé, P Brunet, A Silva, J Wagemaker, and F Zanasi

CKAO

initial state: x = 0, y = 0	
T0	T1
a: x ← 1;	c: y ← 1;
b: r1 ← y;	d: r2 ← x;
assert: 0:r1 = 0 ∧ 1:r2 = 0	

Can we reason about these programs (co)algebraically?

$$(x = 0 \wedge y = 0) \cdot (T0 \mid\mid T1) \leq \neg(r1 = 0 \wedge r2 = 0)$$

Axioms
KA+BA+CKA

Keep conjunction and sequential composition distinct

CKAO

initial state: x = 0, y = 0	
T0	T1
a: x ← 1;	c: y ← 1;
b: r1 ← y;	d: r2 ← x;
assert: 0:r1 = 0 ∧ 1:r2 = 0	

Can we reason about these programs (co)algebraically?

$$(x = 0 \wedge y = 0) \cdot (T0 \mid\mid T1) \leq \neg(r1 = 0 \wedge r2 = 0)$$

Axioms
KA+BA+CKA

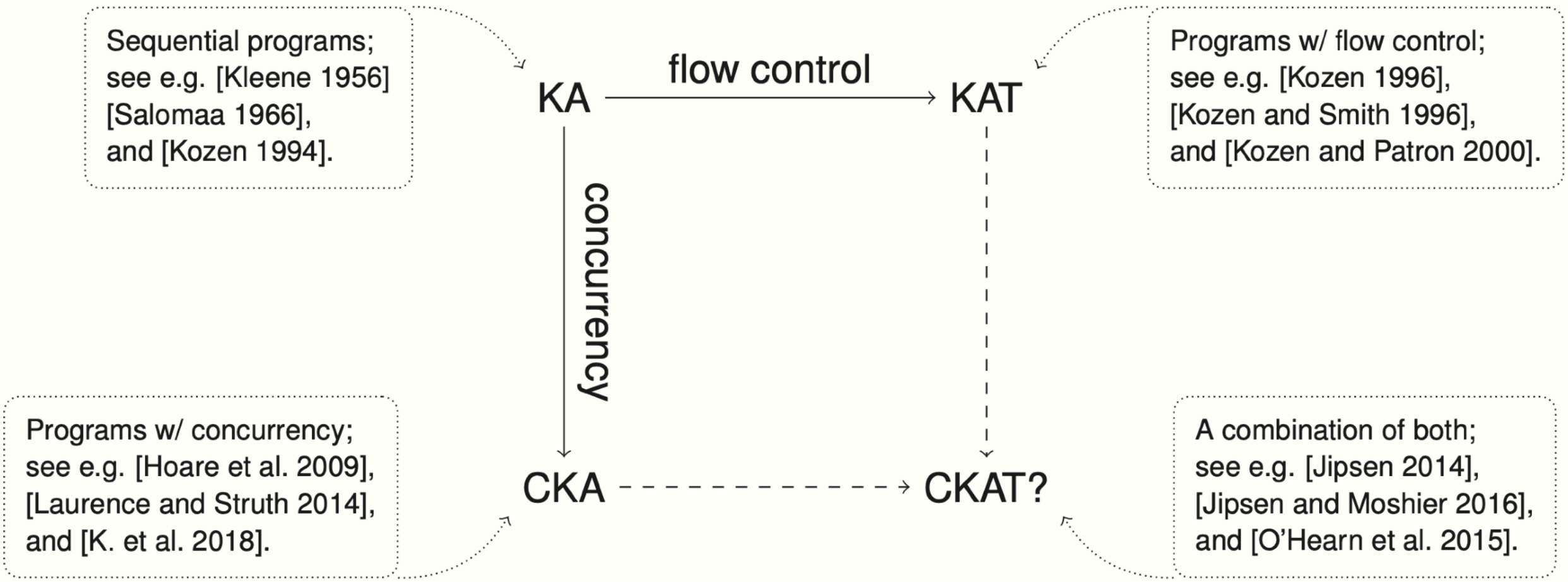
Keep conjunction and sequential composition distinct

Theorem 5.6 (Soundness and Completeness CKAO). *Let* $e, f \in \mathcal{T}_{\mathsf{CKAO}}$. *The following holds:*

$$e \equiv_{\mathsf{CKAO}} f \Leftrightarrow [\![e]\!]_{\mathsf{CKAO}} = [\![f]\!]_{\mathsf{CKAO}}.$$

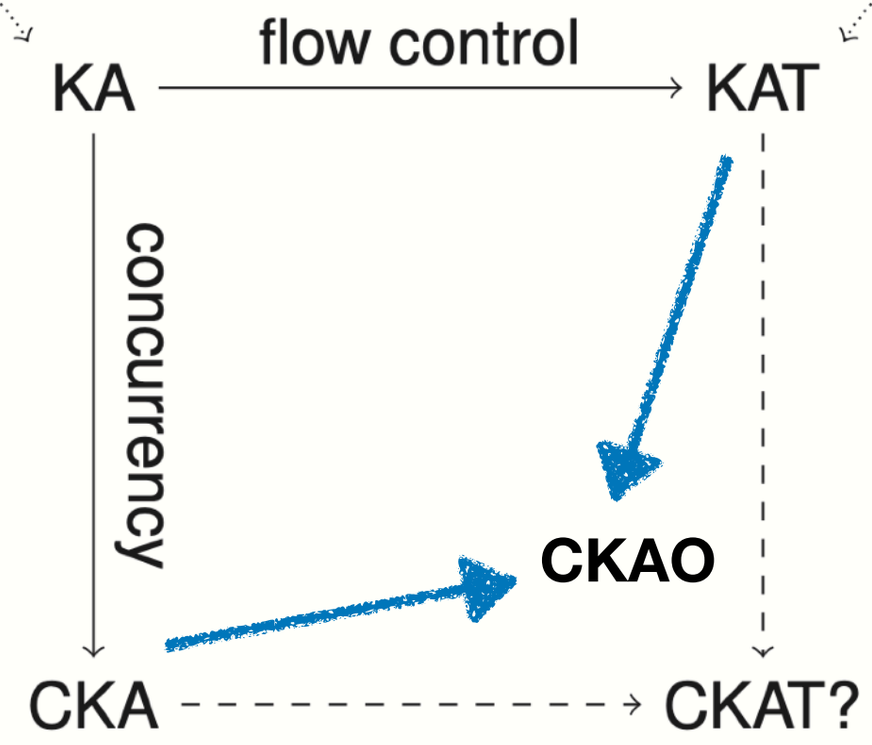
Moreover, given $e, f \in \mathcal{T}$, *it is decidable whether* $[\![e]\!]_{\mathsf{CKAO}} = [\![f]\!]_{\mathsf{CKAO}}$.

Conclusions



Conclusions

Sequential programs;
see e.g. [Kleene 1956]
[Salomaa 1966],
and [Kozen 1994].



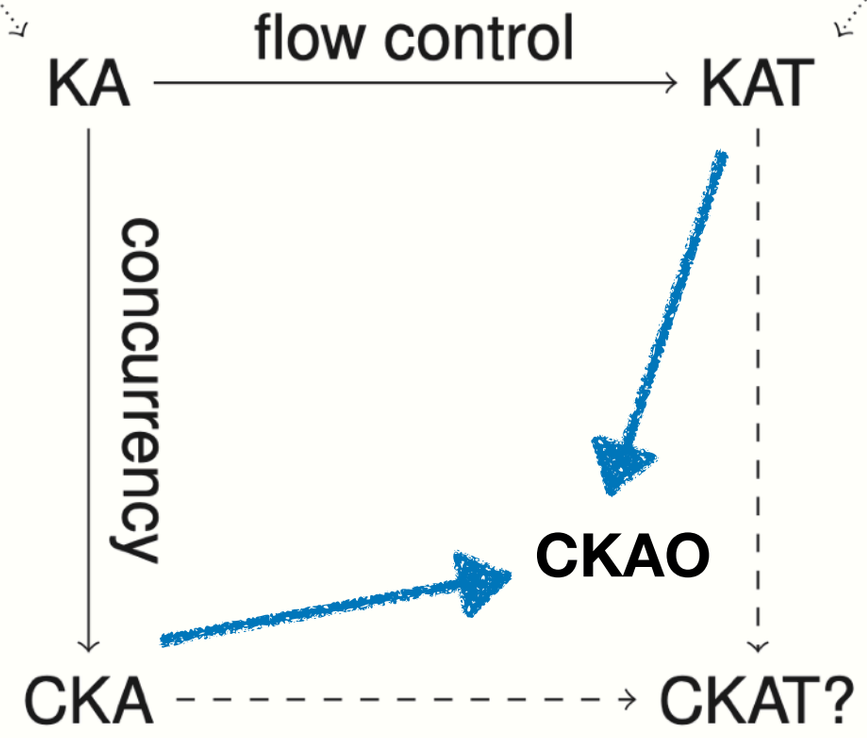
A (co)algebraic framework to analyse programs with concurrency and control flow

Programs w/ concurrency;
see e.g. [Hoare et al. 2009],
[Laurence and Struth 2014],
and [K. et al. 2018].

A combination of both;
see e.g. [Jipsen 2014],
[Jipsen and Moshier 2016],
and [O'Hearn et al. 2015].

Conclusions

Sequential programs;
see e.g. [Kleene 1956]
[Salomaa 1966],
and [Kozen 1994].



A (co)algebraic framework to analyse programs with concurrency and control flow

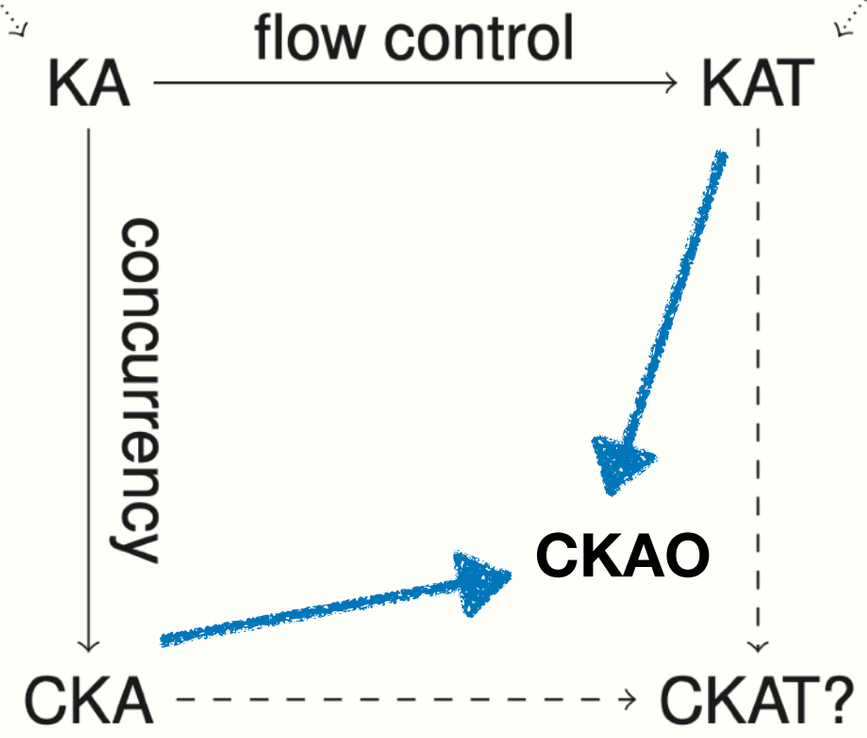
Sound and complete axiomatization
Decidability

Programs w/ concurrency;
see e.g. [Hoare et al. 2009],
[Laurence and Struth 2014],
and [K. et al. 2018].

[Jipsen and Moshier 2016],
and [O'Hearn et al. 2015].

Conclusions

Sequential programs;
see e.g. [Kleene 1956]
[Salomaa 1966],
and [Kozen 1994].



Programs w/ concurrency;
see e.g. [Hoare et al. 2009],
[Laurence and Struth 2014],
and [K. et al. 2018].

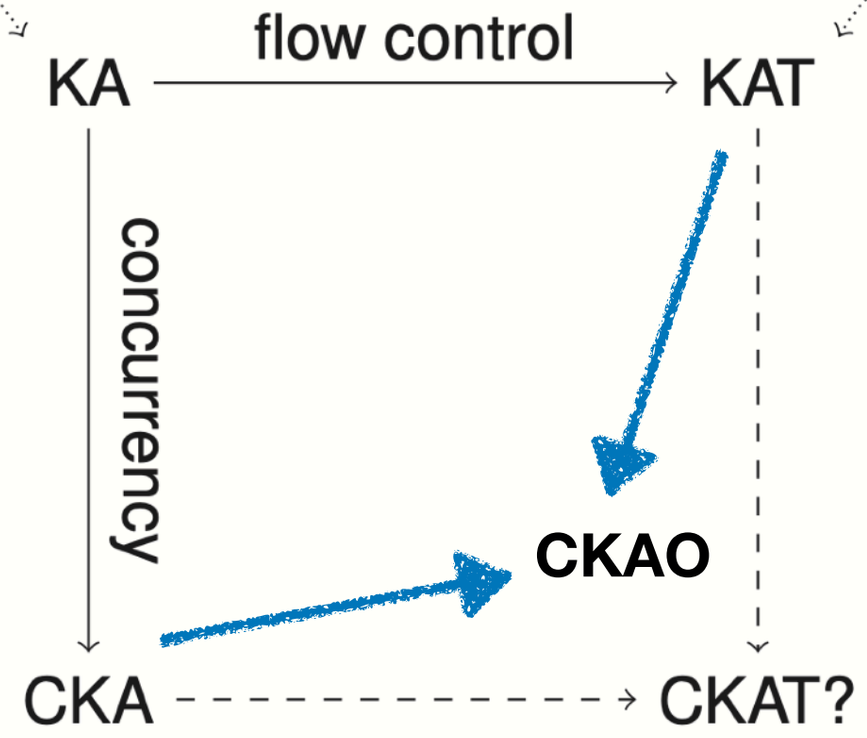
A (co)algebraic framework to analyse programs with concurrency and control flow

Sound and complete axiomatization
Decidability

Currently: case study with partial function memory model

Conclusions

Sequential programs;
see e.g. [Kleene 1956]
[Salomaa 1966],
and [Kozen 1994].



Programs w/ concurrency;
see e.g. [Hoare et al. 2009],
[Laurence and Struth 2014],
and [K. et al. 2018].

A (co)algebraic framework to analyse programs with concurrency and control flow

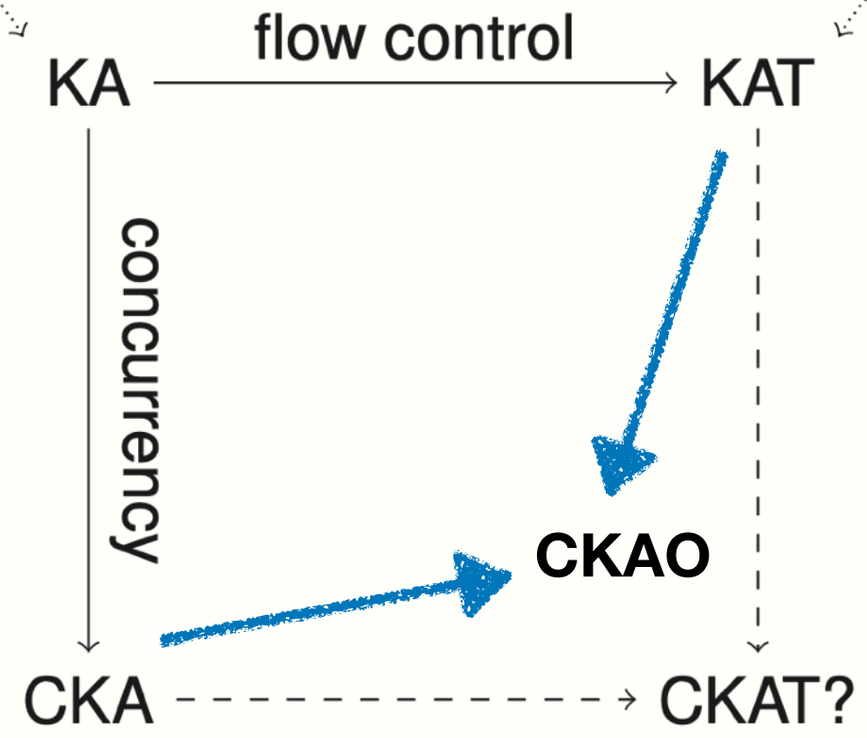
Sound and complete axiomatization
Decidability

Currently: case study with partial function memory model

Soon: Network application

Conclusions

Sequential programs;
see e.g. [Kleene 1956]
[Salomaa 1966],
and [Kozen 1994].



Programs w/ concurrency;
see e.g. [Hoare et al. 2009],
[Laurence and Struth 2014],
and [K. et al. 2018].

A (co)algebraic framework to analyse programs with concurrency and control flow

Sound and complete axiomatization
Decidability

Currently: case study with partial function memory model

Soon: Network application

Questions?

Credits

GoNeCo

http://coneco-project.org