An algebraic framework to reason about concurrency

Alexandra Silva

UCL ENGINEERING
Change the world

UCL

Credits

GoNeCo

http://coneco-project.org

Program verification

```
var perc = 99.0, wmin = 1920, hmin = 1080, w, h, w1, h1, ratio;
var FromDoc = open ( File ("D:\FromMacro.psd"));
var IntoDoc = open ( File ("D:\IntoMacro.psd"));

app.preferences.rulerUnits = Units.PIXELS;
 w = FromDoc.width.value;
 h = FromDoc.height.value;
   ratio = h/w;
app.activeDocument = FromDoc;
activeDocument.activeLayer = activeDocument.layers[0];

var shapeRef =
 [ [ Math.floor ((w-1920)/2), Math.floor ((h-1080)/2) ],
   [ Math.floor ((w-1920)/2)+1920, Math.floor ((h-1080)/2) ],
   [ Math.floor ((w-1920)/2)+1920, Math.floor ((h-1080)/2)+1080 ],
   [ Math.floor ((w-1920)/2), Math.floor ((h-1080)/2)+1080 ] ];

app.activeDocument.selection.select ( shapeRef,SelectionType.REPLACE );
app.activeDocument.selection.copy ();
   app.activeDocument = IntoDoc;
activeDocument.activeLayer = activeDocument.layers[0];
 IntoDoc.paste ();

while (1) {
 if ( (w < wmin) || (h < hmin) ) break;
   app.activeDocument = FromDoc;
   activeDocument.activeLayer = activeDocument.layers[0];

app.activeDocument.activeLayer.copy ();
app.activeDocument = betweenDoc;
betweenDoc.paste ();
w1 = w;
h1 = h;
w = w * perc / 100;
h = w * ratio;
}
```

Program verification

```
var perc = 99.0, wmin = 1920, hmin = 1080, w, h, w1, h1, ratio;
var FromDoc = open ( File ("D:\FromMacro.psd"));
var IntoDoc = open ( File ("D:\IntoMacro.psd"));

app.preferences.rulerUnits = Units.PIXELS;
 w = FromDoc.width.value;
 h = FromDoc.height.value;
   ratio = h/w;
app.activeDocument = FromDoc;
activeDocument.activeLayer = activeDocument.layers[0];

var shapeRef =
 [ [ Math.floor ((w-1920)/2), Math.floor ((h-1080)/2) ],
   [ Math.floor ((w-1920)/2)+1920, Math.floor ((h-1080)/2) ],
   [ Math.floor ((w-1920)/2)+1920, Math.floor ((h-1080)/2)+1080 ],
   [ Math.floor ((w-1920)/2), Math.floor ((h-1080)/2)+1080 ] ];

app.activeDocument.selection.select ( shapeRef,SelectionType.REPLACE );
app.activeDocument.selection.copy ();
   app.activeDocument = IntoDoc;
activeDocument.activeLayer = activeDocument.layers[0];
 IntoDoc.paste ();

while (1) {
 if ( (w < wmin) || (h < hmin) ) break;
   app.activeDocument = FromDoc;
   activeDocument.activeLayer = activeDocument.layers[0

app.activeDocument.activeLayer.copy ();
app.activeDocument = betweenDoc;
betweenDoc.paste ();
w1 = w;
h1 = h;
w = w * perc / 100;
h = w * ratio;
}
```

Program verification

```
var perc = 99.0, wmin = 1920, hmin = 1080, w, h, w1, h1, ratio;
var FromDoc = open ( File ("D:\FromMacro.psd"));
var IntoDoc = open ( File ("D:\IntoMacro.psd"));

app.preferences.rulerUnits = Units.PIXELS;
 w = FromDoc.width.value;
 h = FromDoc.height.value;
   ratio = h/w;
app.activeDocument = FromDoc;
activeDocument.activeLayer = activeDocument.layers[0];

var shapeRef =
 [ [ Math.floor ((w-1920)/2), Math.floor ((h-1080)/2) ],
   [ Math.floor ((w-1920)/2)+1920, Math.floor ((h-1080)/2) ],
   [ Math.floor ((w-1920)/2)+1920, Math.floor ((h-1080)/2)+1080 ],
   [ Math.floor ((w-1920)/2), Math.floor ((h-1080)/2)+1080 ] ];

app.activeDocument.selection.select ( shapeRef,SelectionType.REPLACE );
app.activeDocument.selection.copy ();
   app.activeDocument = IntoDoc;
activeDocument.activeLayer = activeDocument.layers[0];
 IntoDoc.paste ();

while (1) {
 if ( (w < wmin) || (h < hmin) ) break;
   app.activeDocument = FromDoc;
   activeDocument.activeLayer = activeDocument.layers[0

app.activeDocument.activeLayer.copy ();
app.activeDocument = betweenDoc;
betweenDoc.paste ();
w1 = w;
h1 = h;
w = w * perc / 100;
h = w * ratio;
}
```

Where can things go wrong?

Language semantics

Compiler translation

Hardware

Program verification

```
var perc = 99.0, wmin = 1920, hmin = 1080, w, h, w1, h1, ratio;
var FromDoc = open ( File ("D:\FromMacro.psd"));
var IntoDoc = open ( File ("D:\IntoMacro.psd"));

app.preferences.rulerUnits = Units.PIXELS;
 w = FromDoc.width.value;
 h = FromDoc.height.value;
   ratio = h/w;
app.activeDocument = FromDoc;
activeDocument.activeLayer = activeDocument.layers[0];

var shapeRef =
 [ [ Math.floor ((w-1920)/2), Math.floor ((h-1080)/2) ],
   [ Math.floor ((w-1920)/2)+1920, Math.floor ((h-1080)/2) ],
   [ Math.floor ((w-1920)/2)+1920, Math.floor ((h-1080)/2)+1080 ],
   [ Math.floor ((w-1920)/2), Math.floor ((h-1080)/2)+1080 ] ];

app.activeDocument.selection.select ( shapeRef,SelectionType.REPLACE );
app.activeDocument.selection.copy ();
   app.activeDocument = IntoDoc;
activeDocument.activeLayer = activeDocument.layers[0];
 IntoDoc.paste ();

while (1) {
 if ( (w < wmin) || (h < hmin) ) break;
   app.activeDocument = FromDoc;
   activeDocument.activeLayer = activeDocument.layers[0

app.activeDocument.activeLayer.copy ();
app.activeDocument = betweenDoc;
betweenDoc.paste ();
w1 = w;
h1 = h;
w = w * perc / 100;
h = w * ratio;
}
```

Where can things go wrong?

Language semantics

Compiler translation

Hardware

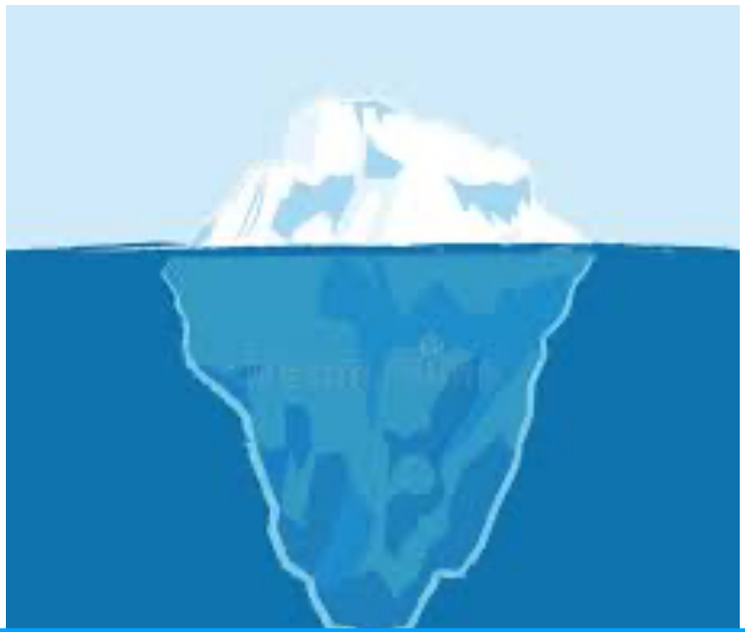
Verification is hard!

Program verification

Language semantics

Compiler translation

Hardware



Verification is hard!

Program verification

distributed computation

Language semantics

Compiler translation

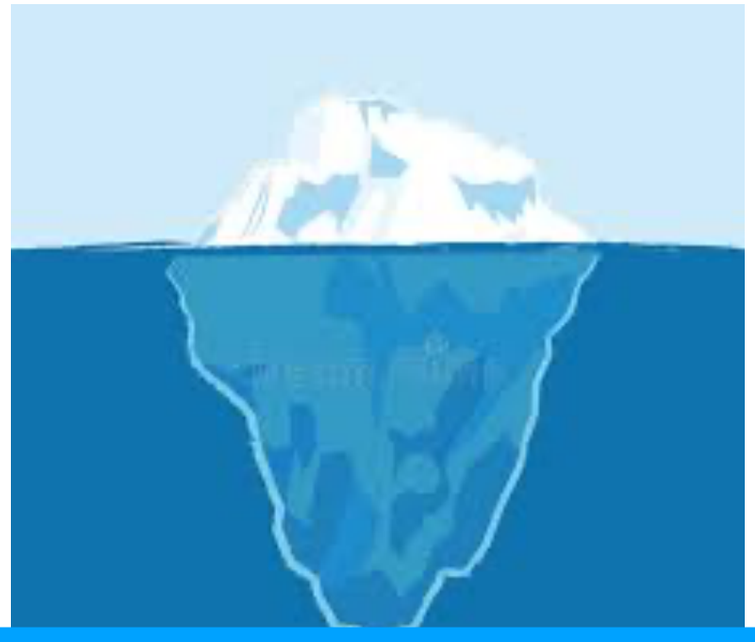
Hardware

concurrent communication

Verification is hard!

Program verification

```
var perc = 99.0, wmin = 1920, hmin = 1080, w, h, w1, h1, ratio;
var FromDoc = open ( File ("D:\FromMacro.psd"));
var IntoDoc = open ( File ("D:\IntoMacro.psd"));

app.preferences.rulerUnits = Units.PIXELS;
 w = FromDoc.width.value;
 h = FromDoc.height.value;
   ratio = h/w;
app.activeDocument = FromDoc;
activeDocument.activeLayer = activeDocument.layers[0];

var shapeRef =
 [ [ Math.floor ((w-1920)/2), Math.floor ((h-1080)/2) ],
   [ Math.floor ((w-1920)/2)+1920, Math.floor ((h-1080)/2) ],
   [ Math.floor ((w-1920)/2)+1920, Math.floor ((h-1080)/2)+1080 ],
   [ Math.floor ((w-1920)/2), Math.floor ((h-1080)/2)+1080 ] ];

app.activeDocument.selection.select ( shapeRef,SelectionType.REPLACE );
app.activeDocument.selection.copy ();
   app.activeDocument = IntoDoc;
activeDocument.activeLayer = activeDocument.layers[0];
 IntoDoc.paste ();

while (1) {
 if ( (w < wmin) || (h < hmin) ) break;
   app.activeDocument = FromDoc;
   activeDocument.activeLayer = activeDocument.layers[0

app.activeDocument.activeLayer.copy ();
app.activeDocument = betweenDoc;
betweenDoc.paste ();
w1 = w;
h1 = h;
w = w * perc / 100;
h = w * ratio;
}
```

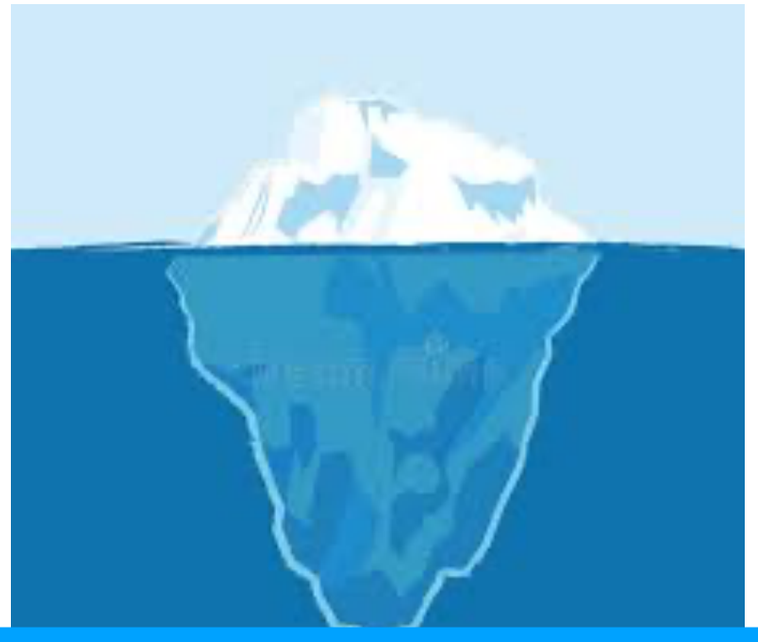


Verification is hard!

Program verification

```
var perc = 99.0, wmin = 1920, hmin = 1080, w, h, w1, h1, ratio;
var FromDoc = open ( File ("D:\FromMacro.psd"));
var IntoDoc = open ( File ("D:\IntoMacro.psd"));

app.preferences.rulerUnits = Units.PIXELS;
 w = FromDoc.width.value;
 h = FromDoc.height.value;
   ratio = h/w;
app.activeDocument = FromDoc;
activeDocument.activeLayer = activeDocument.layers[0];

var shapeRef =
 [ [ Math.floor ((w-1920)/2), Math.floor ((h-1080)/2) ],
   [ Math.floor ((w-1920)/2)+1920, Math.floor ((h-1080)/2) ],
   [ Math.floor ((w-1920)/2)+1920, Math.floor ((h-1080)/2)+1080 ],
   [ Math.floor ((w-1920)/2), Math.floor ((h-1080)/2)+1080 ] ];

app.activeDocument.selection.select ( shapeRef,SelectionType.REPLACE );
app.activeDocument.selection.copy ();
   app.activeDocument = IntoDoc;
activeDocument.activeLayer = activeDocument.layers[0];
 IntoDoc.paste ();

while (1) {
 if ( (w < wmin) || (h < hmin) ) break;
   app.activeDocument = FromDoc;
   activeDocument.activeLayer = activeDocument.layers[0

app.activeDocument.activeLayer.copy ();
app.activeDocument = betweenDoc;
betweenDoc.paste ();
w1 = w;
h1 = h;
w = w * perc / 100;
h = w * ratio;
}
```

Abstraction



Verification is hard!

Program verification

```
var perc = 99.0, wmin = 1920, hmin = 10
var FromDoc = open ( File ("D:\FromMacr
var IntoDoc = open ( File ("D:\IntoMacr

app.preferences.rulerUnits = Units.PIXELS;
 w = FromDoc.width.value;
 h = FromDoc.height.value;
   ratio = h/w;
app.activeDocument = FromDoc;
activeDocument.activeLayer = activeDocumen

var shapeRef =
 [ [ Math.floor ((w-1920)/2), Math.floor (
   [ Math.floor ((w-1920)/2)+1920, Math.fl
   [ Math.floor ((w-1920)/2)+1920, Math.fl
   [ Math.floor ((w-1920)/2), Math.floor (

app.activeDocument.selection.select ( shap
app.activeDocument.selection.copy ();
   app.activeDocument = IntoDoc;
activeDocument.activeLayer = activeDocument.layers[0];
 IntoDoc.paste ();

while (1) {
 if ( (w < wmin) || (h < hmin) ) break;
   app.activeDocument = FromDoc;
   activeDocument.activeLayer = activeDocument.layers[0

app.activeDocument.activeLayer.copy ();
app.activeDocument = betweenDoc;
betweenDoc.paste ();
w1 = w;
h1 = h;
w = w * perc / 100;
h = w * ratio;
}
```

cluster data values

restrict to function calls

Abstraction

control flow

property slicing

Verification is hard!

Example

Write a program that prints *n>0 space separated* ☺

Example

Write a program that prints *n>0 space separated* ☺

```
i := 1
while i < n do
    print ☺
    print ␣
    i := i + 1
end
print ☺
```

```
i := 1
print ☺
while i < n do
    print ␣
    print ☺
    i := i + 1
end
```

Example

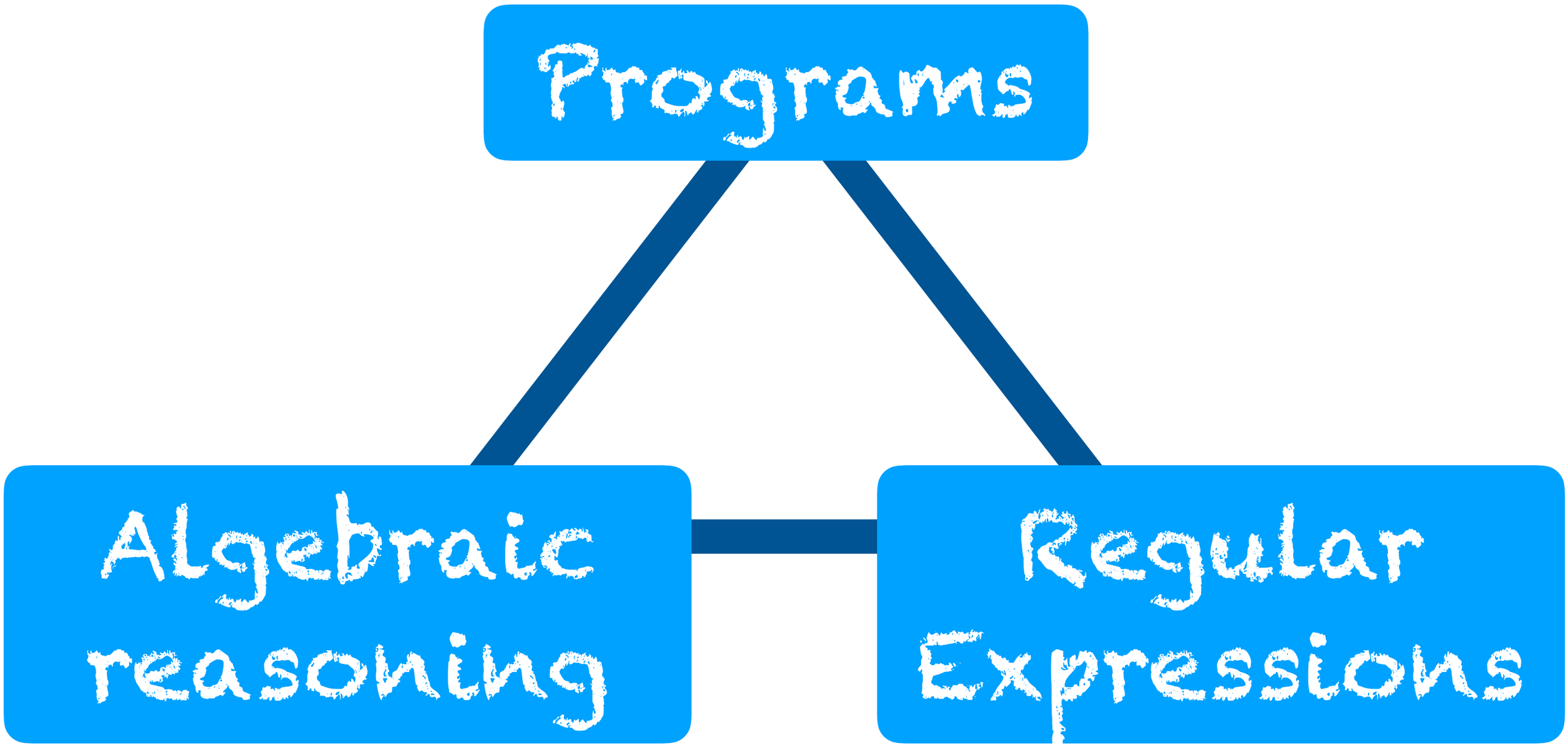
Write a program that prints *n>0 space separated* ☺

```
i := 1
while i < n do
    print ☺
    print ␣
    i := i + 1
end
print ☺
```

Equivalent?

```
i := 1
print ☺
while i < n do
    print ␣
    print ☺
    i := i + 1
end
```

(Co)Algebraic techniques



Example revisited

```
i := 1
while i < n do
    print ☺
    print ␣
    i := i + 1
end
print ☺
```

$(☺ \cdot ␣)^* \cdot ☺$

```
i := 1
print ☺
while i < n do
    print ␣
    print ☺
    i := i + 1
end
```

$☺ \cdot (␣ \cdot ☺)^*$

Example revisited

```
i := 1
while i < n do
    print ☺
    print ␣
    i := i + 1
end
print ☺
```

$(☺ \cdot ␣)^* \cdot ☺$

```
i := 1
print ☺
while i < n do
    print ␣
    print ☺
    i := i + 1
end
```

$☺ \cdot (␣ \cdot ☺)^*$

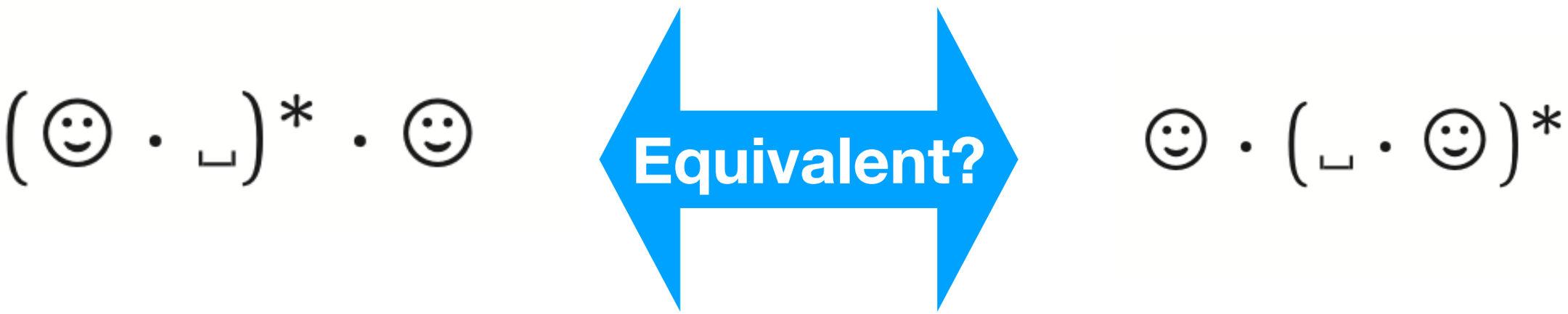
Abstraction

Example revisited

(☺ · ␣)* · ☺

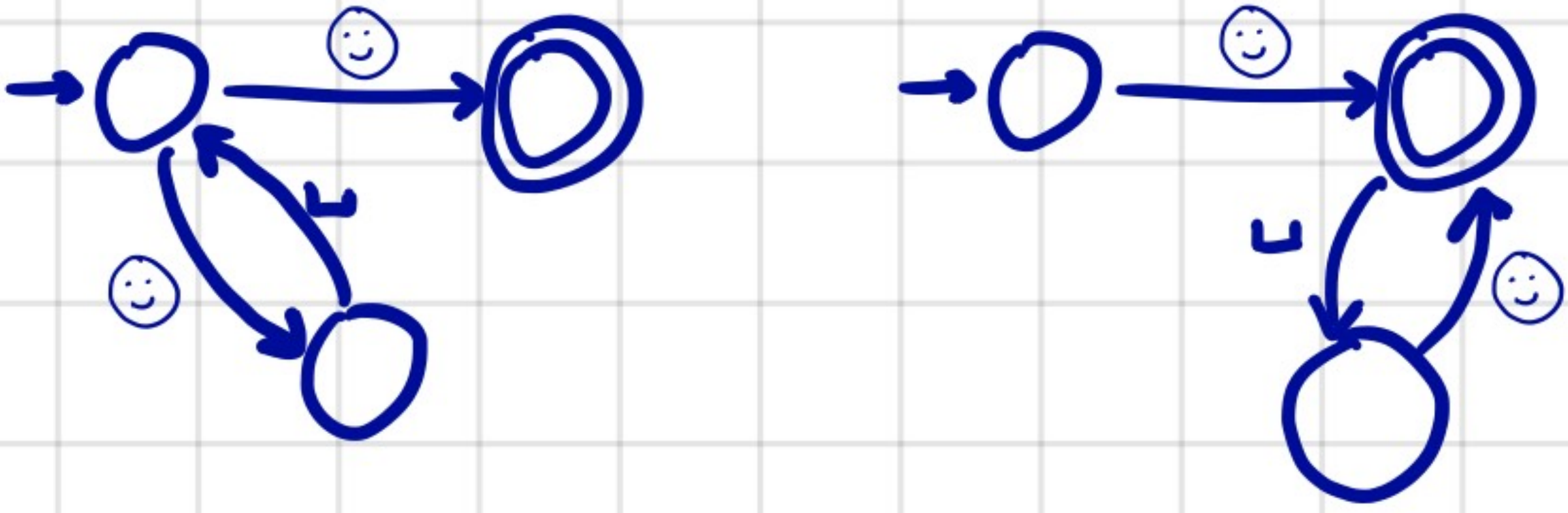
☺ · (␣ · ☺)*

Example revisited



Example revisited

(☺ · ␣)* · ☺ **Equivalent?** ☺ · (␣ · ☺)*



Concurrent programs

initial state: $x = 0, y = 0$	
T0	T1
`a: x ← 1;`	`c: y ← 1;`
`b: r1 ← y;`	`d: r2 ← x;`
assert: $0{:}r1 = 0 \wedge 1{:}r2 = 0$	

Concurrent programs

initial state: $x = 0, y = 0$	
T0	T1
a: $x \leftarrow 1$;	c: $y \leftarrow 1$;
b: $r1 \leftarrow y$;	d: $r2 \leftarrow x$;
assert: $0{:}r1 = 0 \wedge 1{:}r2 = 0$	

Can we reason about these programs (co)algebraically?

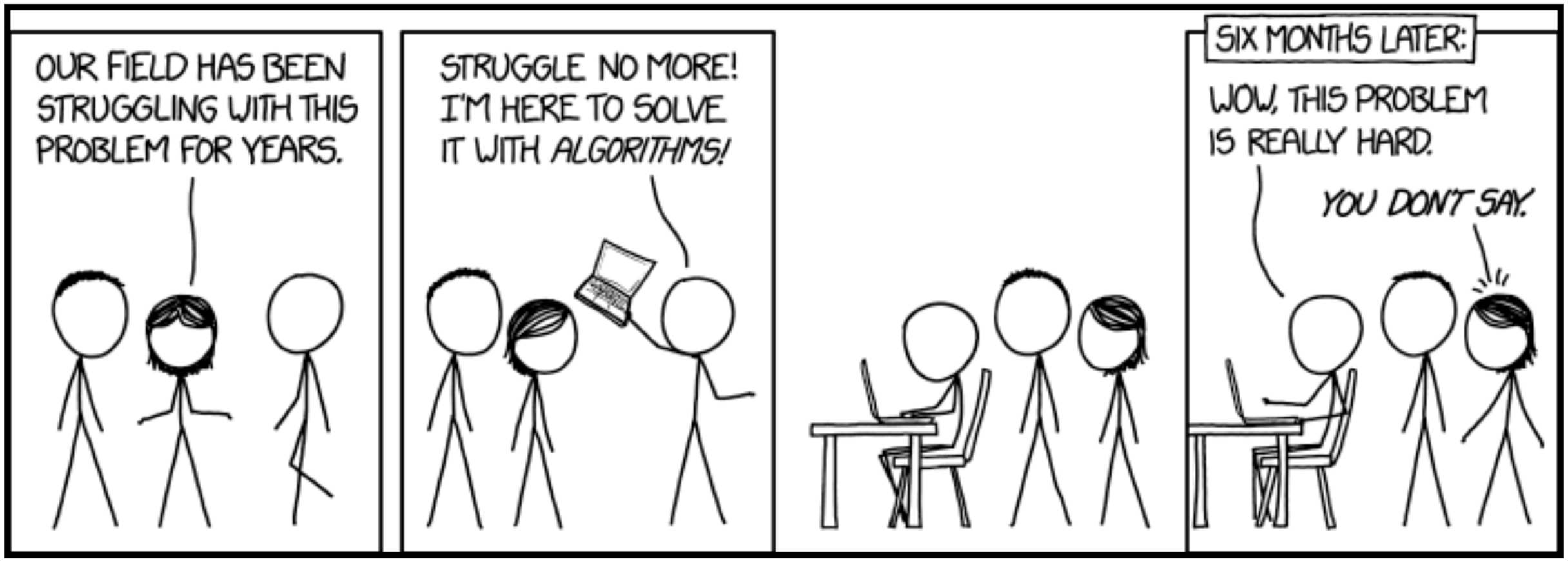
Concurrent programs

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

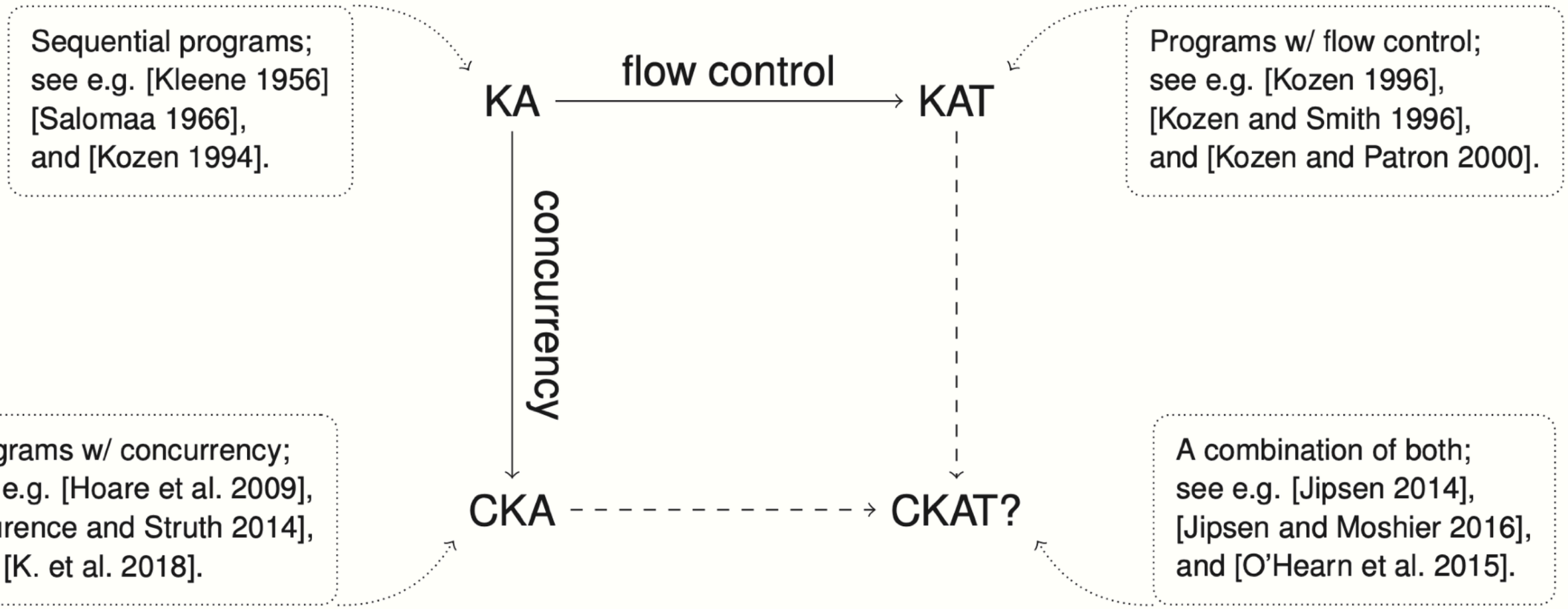
Can we reason about these programs (co)algebraically?

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

Concurrency

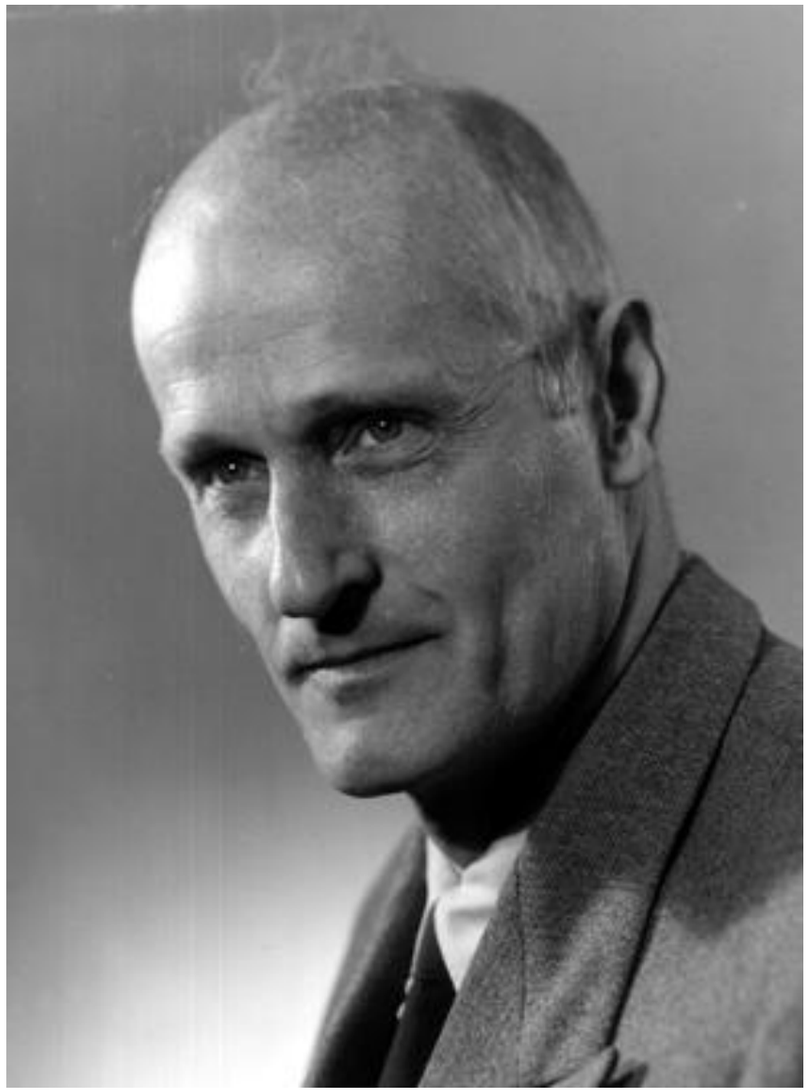


This talk


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…

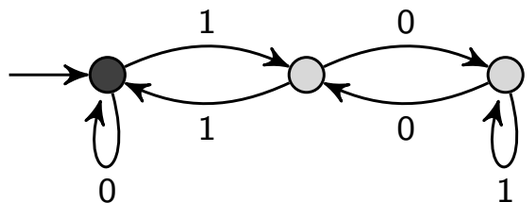
Kleene algebra - **KA**



Stephen Cole Kleene
(1909–1994)

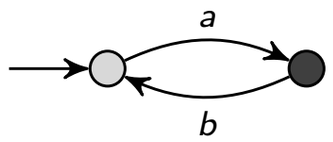
$(0 + 1(01^*0)^*1)^*$

{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**

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

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

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

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

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 1990)

The axioms for KA *are sound &* ***complete*** *for equivalence:*

$$e \equiv f \iff \mathcal{L}(e) = \mathcal{L}(f)$$

$\mathcal{L}(e)$ *is the regular language interpretation of* e*.*

Equivalence

- To check KA equivalence is to check regular language equivalence
- Via Kleene's theorem: checking DFA equivalence
- Sophisticated (near-linear) algorithms exist for automata equivalence
- Hopcroft-Karp, Bisimulation up-to HKC

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

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

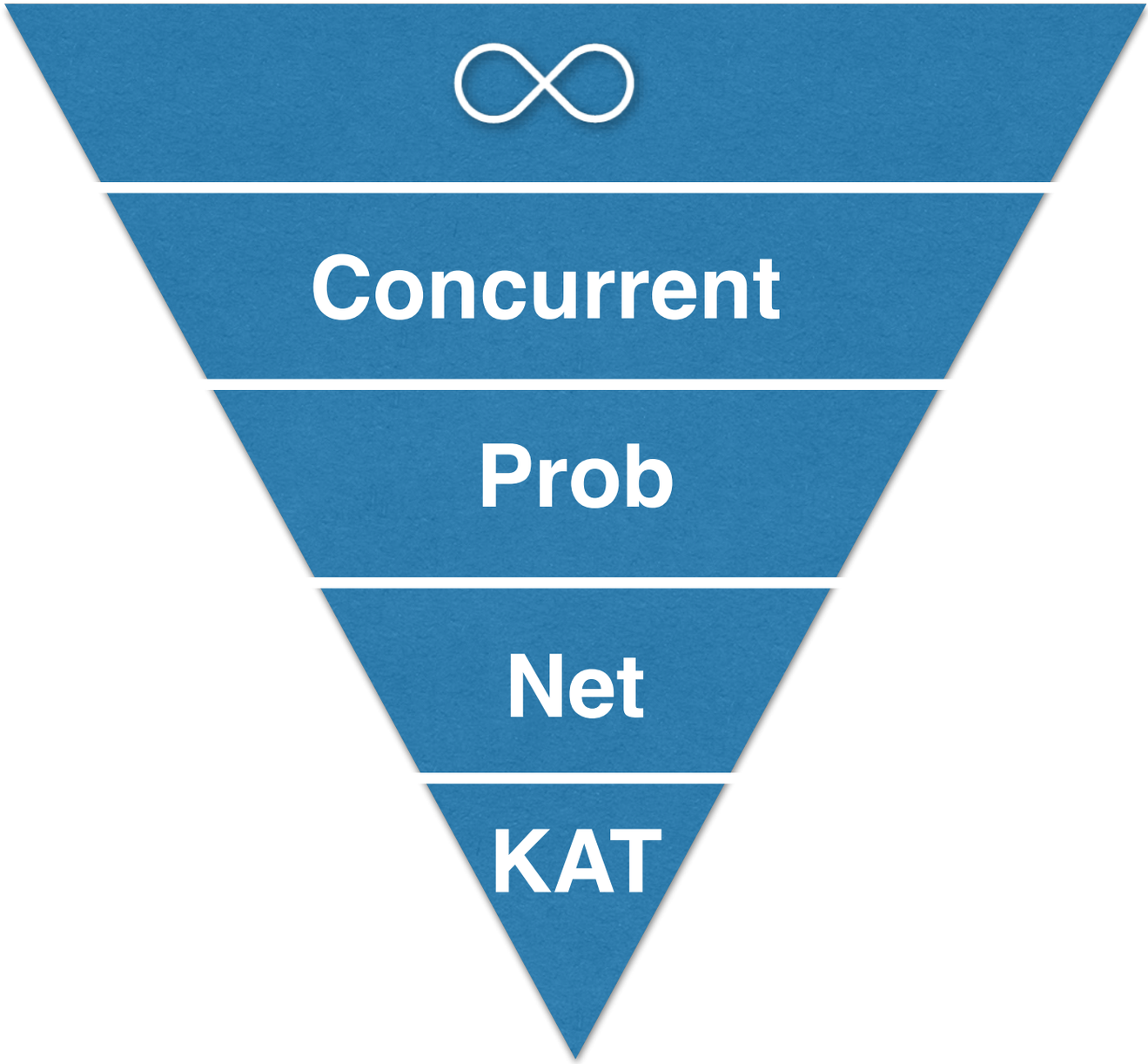
$$\{\alpha_{!b}, \alpha_b \mathsf{p} \alpha_{!b}, \alpha_b \mathsf{p} \alpha_b \mathsf{p} \alpha_{!b}, \ldots\}$$

valuations
variables

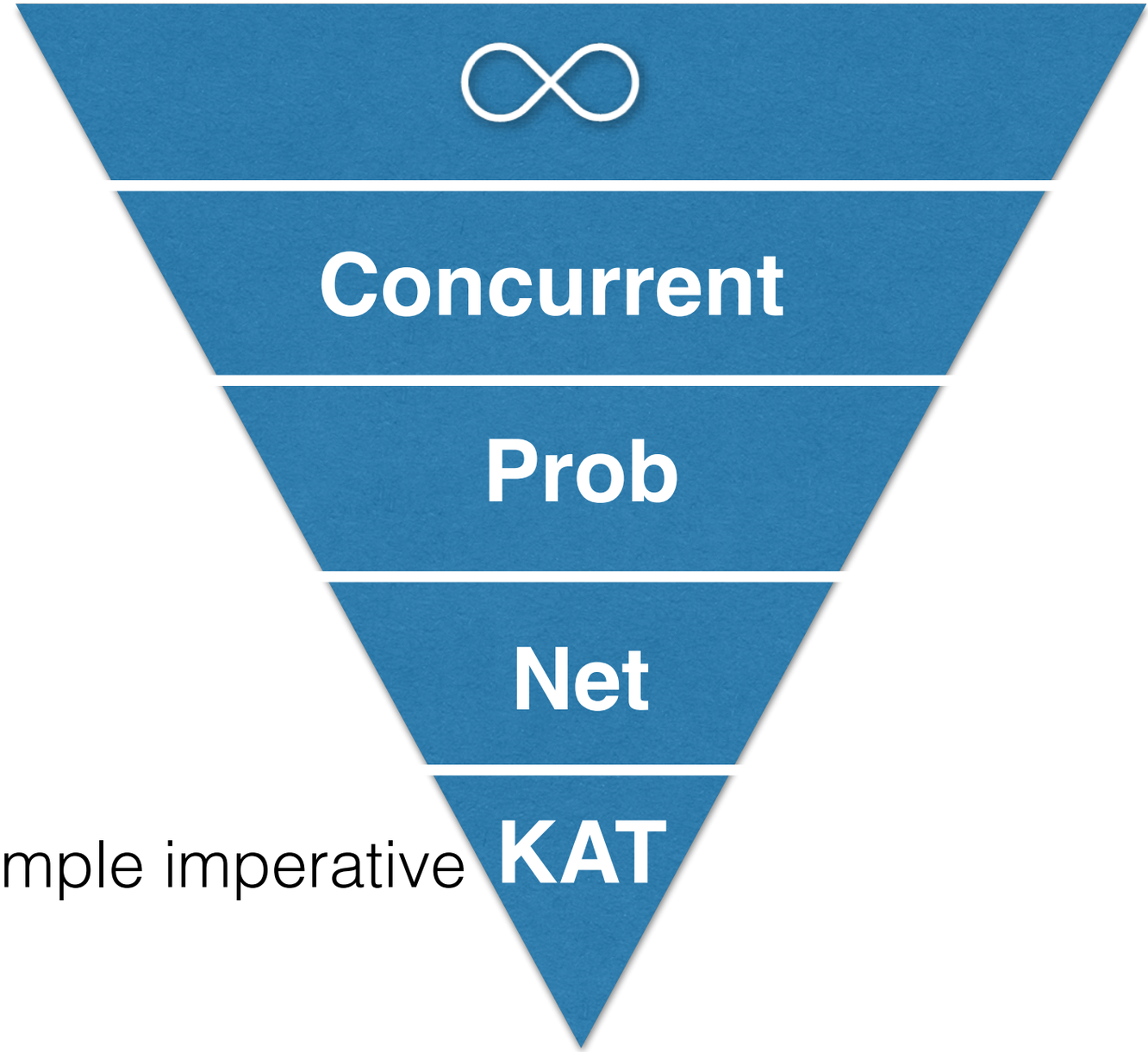
KAT

- Axiomatisation of equivalence
- Kleene-like theorem and algorithms for automata equivalence
- Used in verification of e.g. compiler optimisation

The KAT tower principle

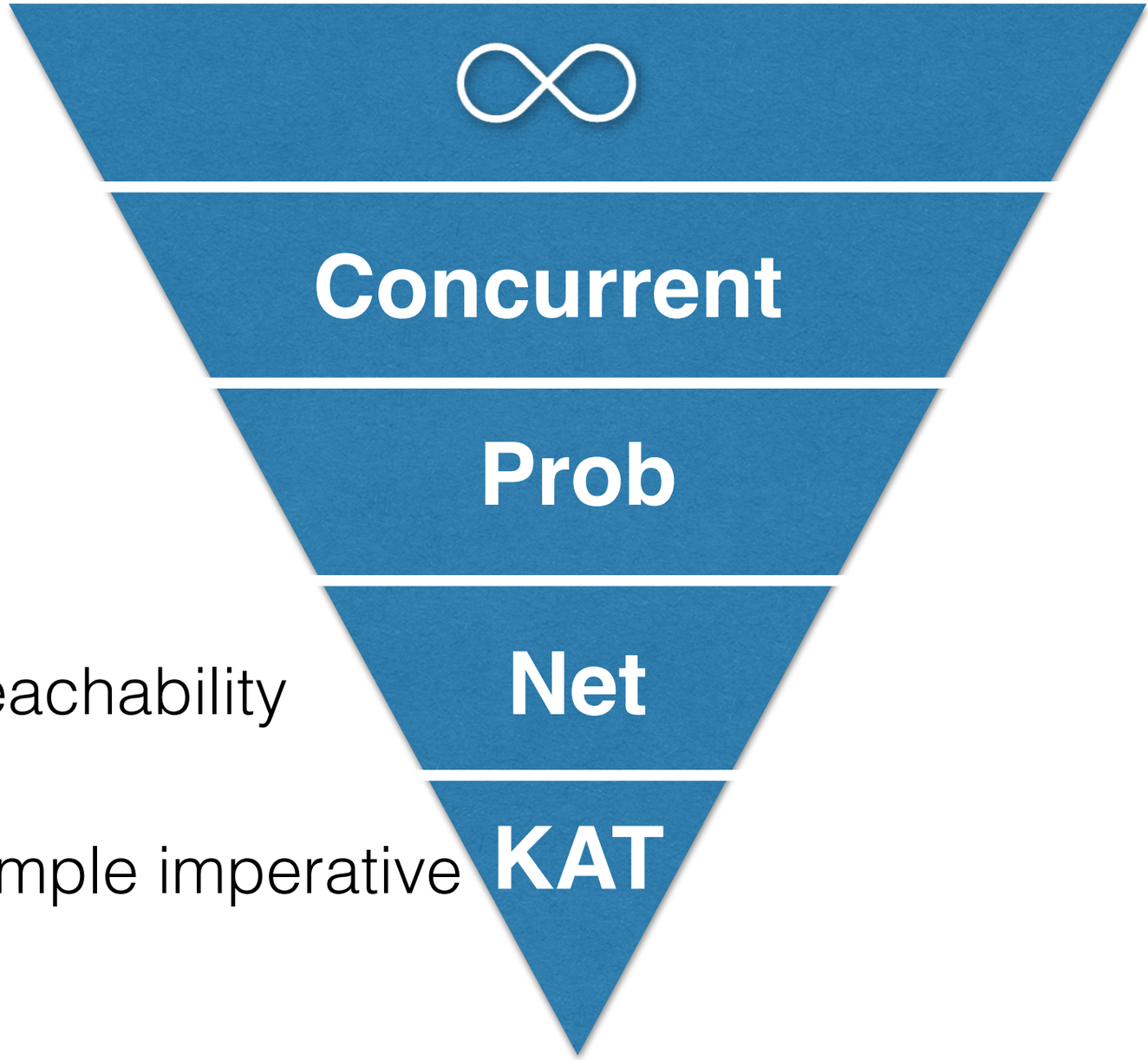


The KAT tower principle



simple imperative

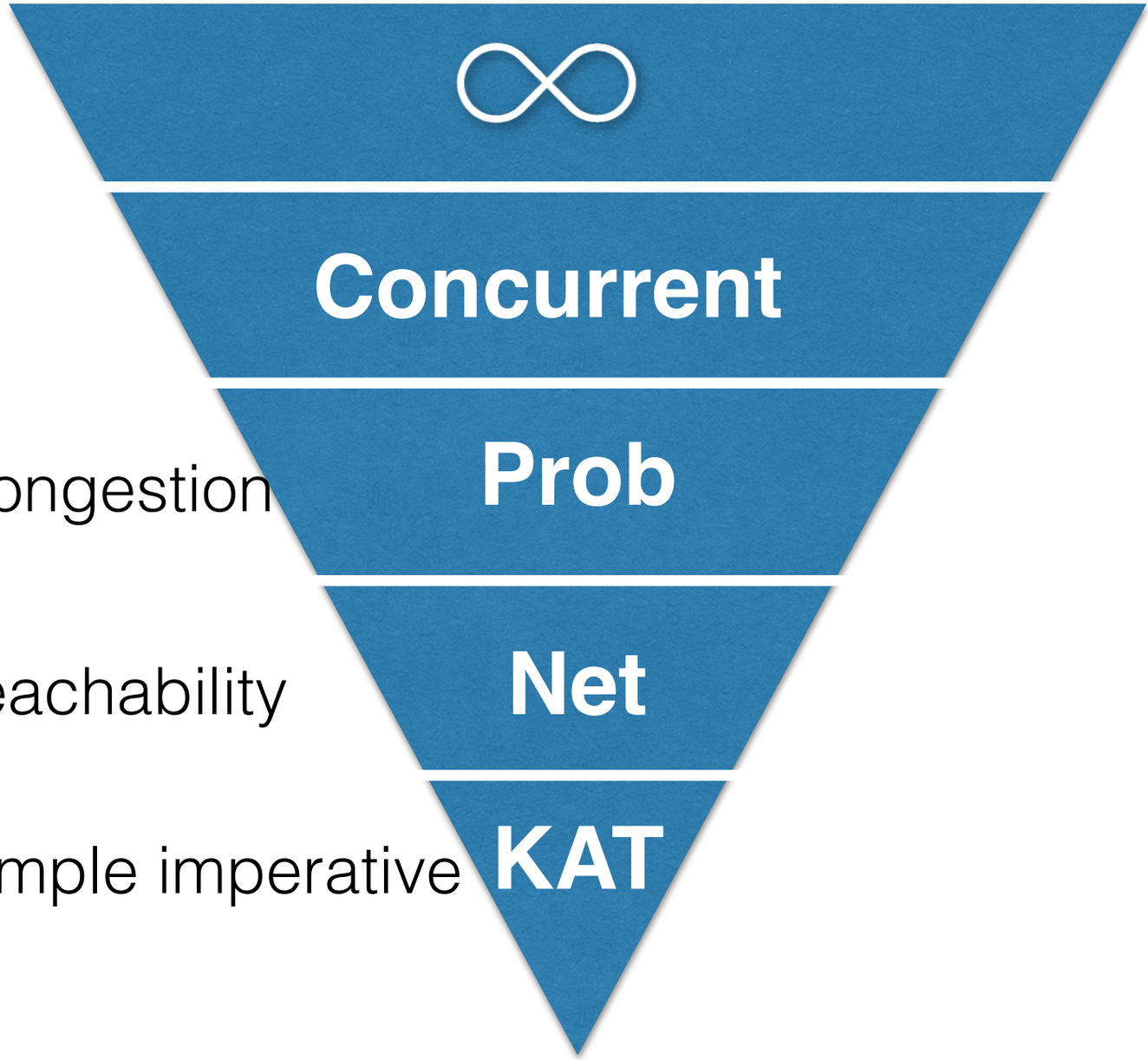
The KAT tower principle



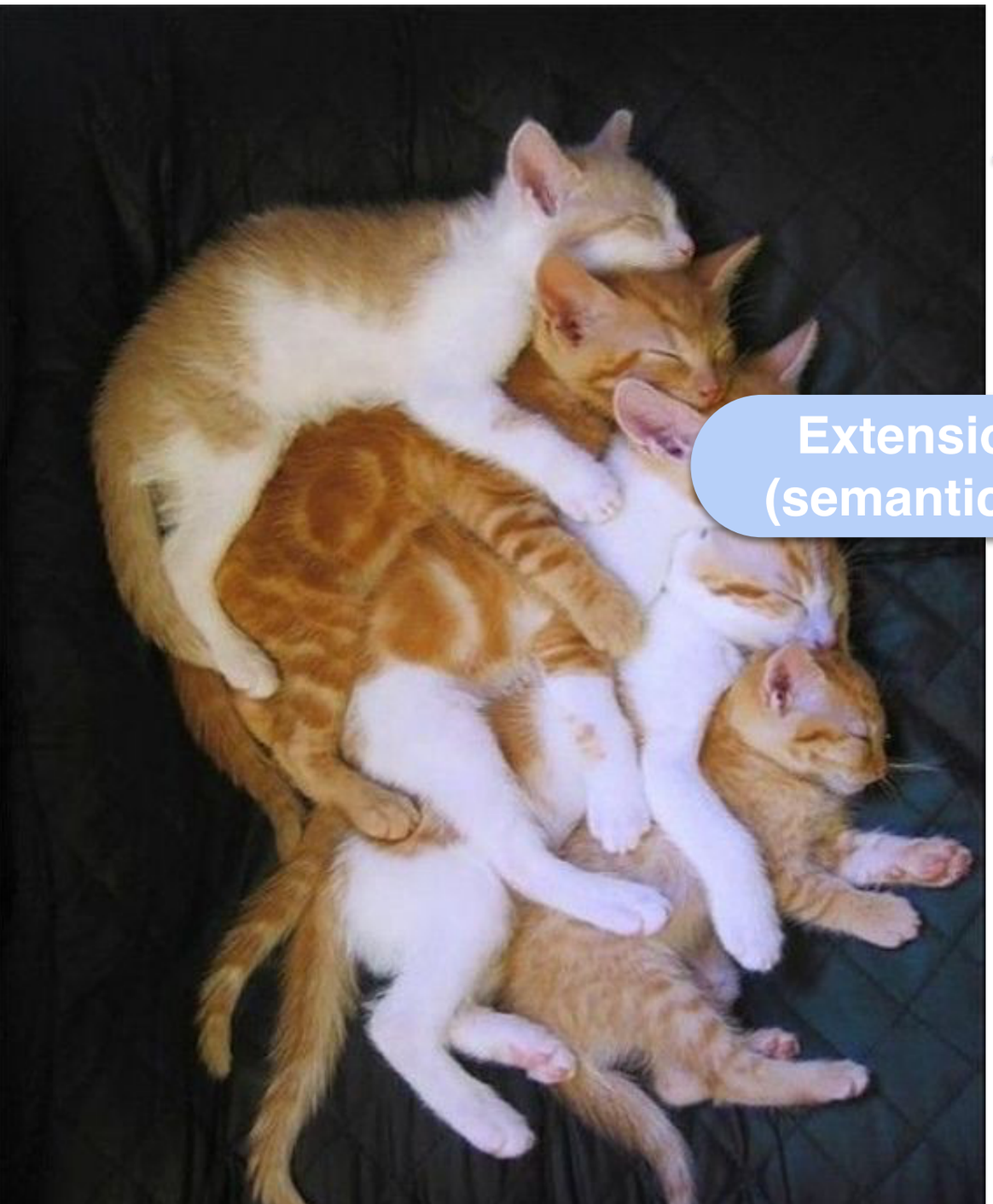
reachability

simple imperative

The KAT tower principle



The KAT tower principle



Extensions conservative
(semantics and algorithms)

∞

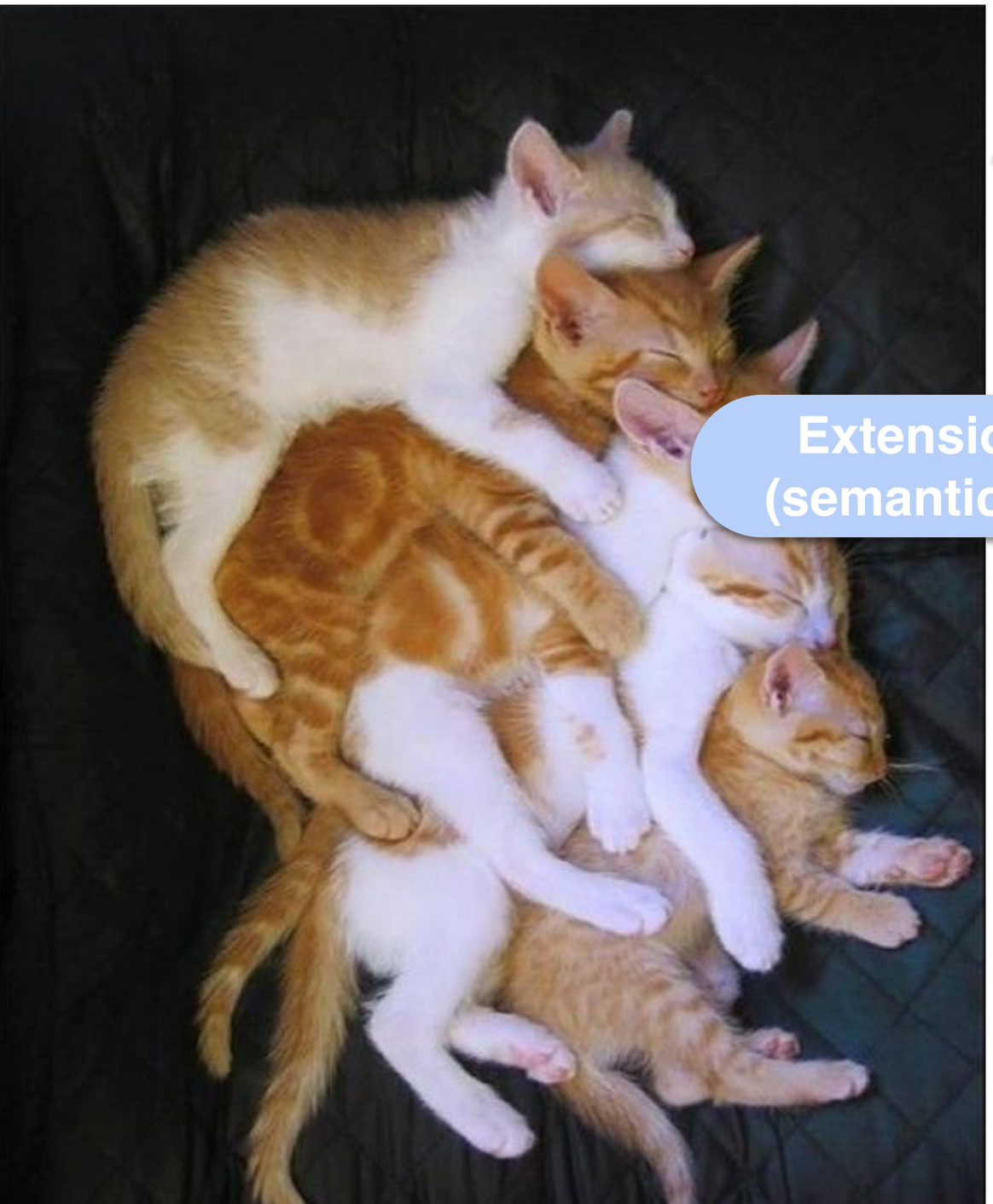
Concurrent

Prob

Net

KAT

The KAT tower principle



Extensions conservative
(semantics and algorithms)

∞

Concurrent

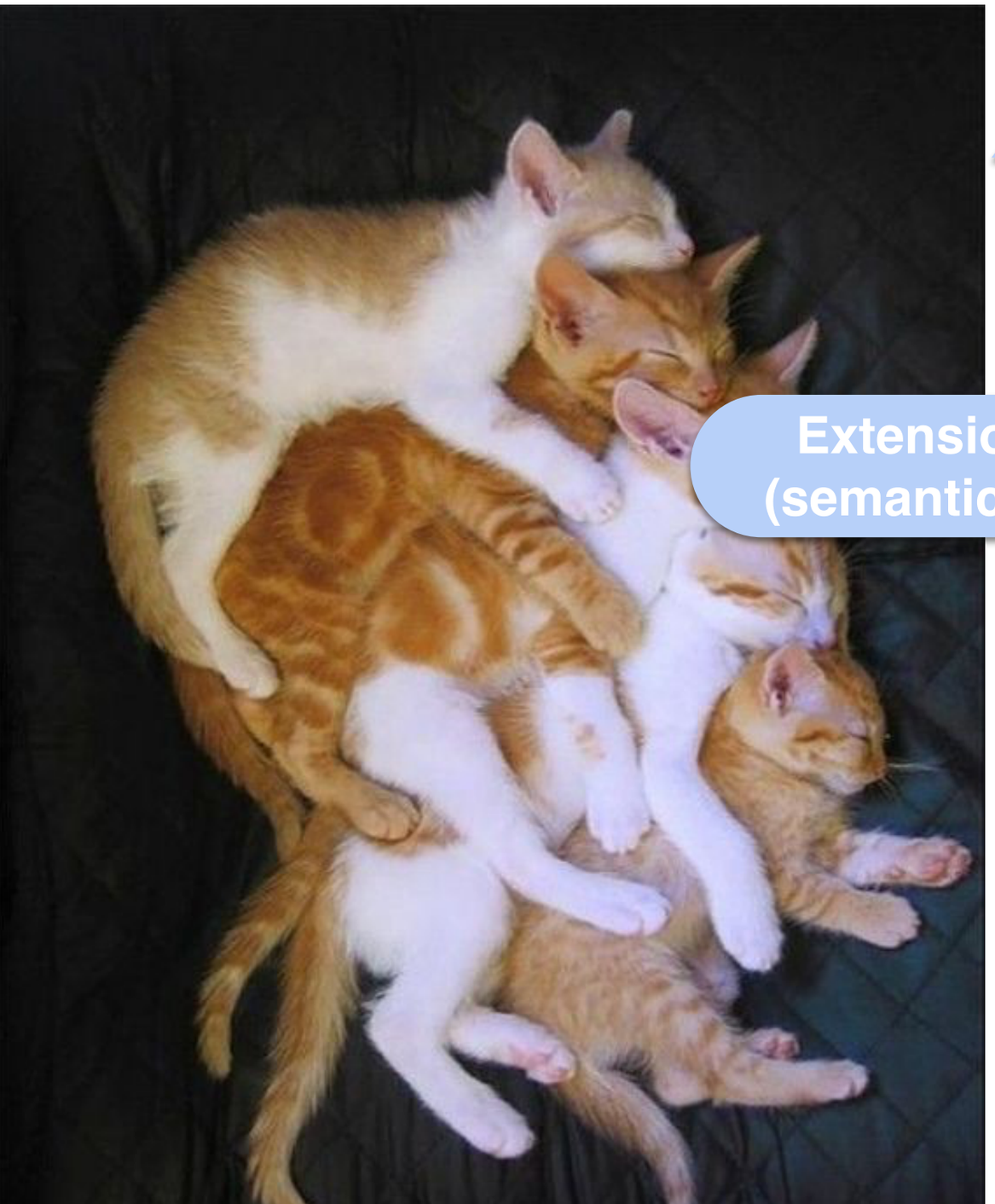
Prob

Net

KAT

Compositional Semantics

The KAT tower principle



Extensions conservative (semantics and algorithms)

∞

Concurrent

Prob

Net

KAT

Compositional Semantics

Scalable verification

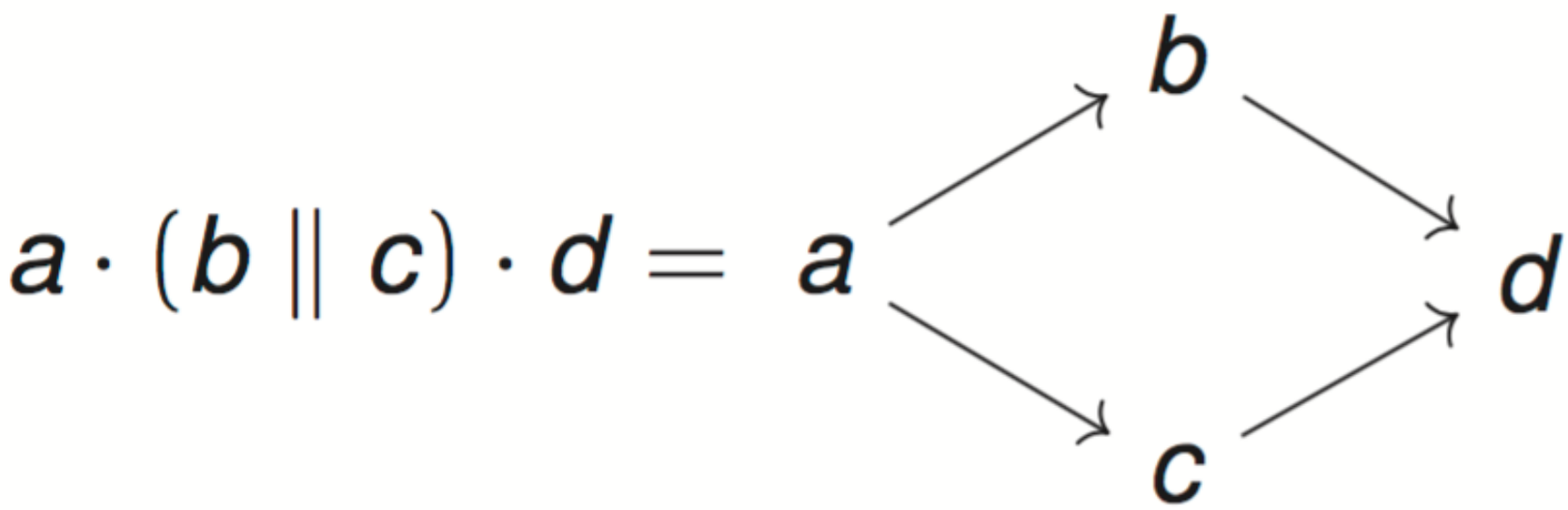
Concurrent Kleene Algebra
(Hoare, Moeller, Struth, Wehrman'09)

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

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)

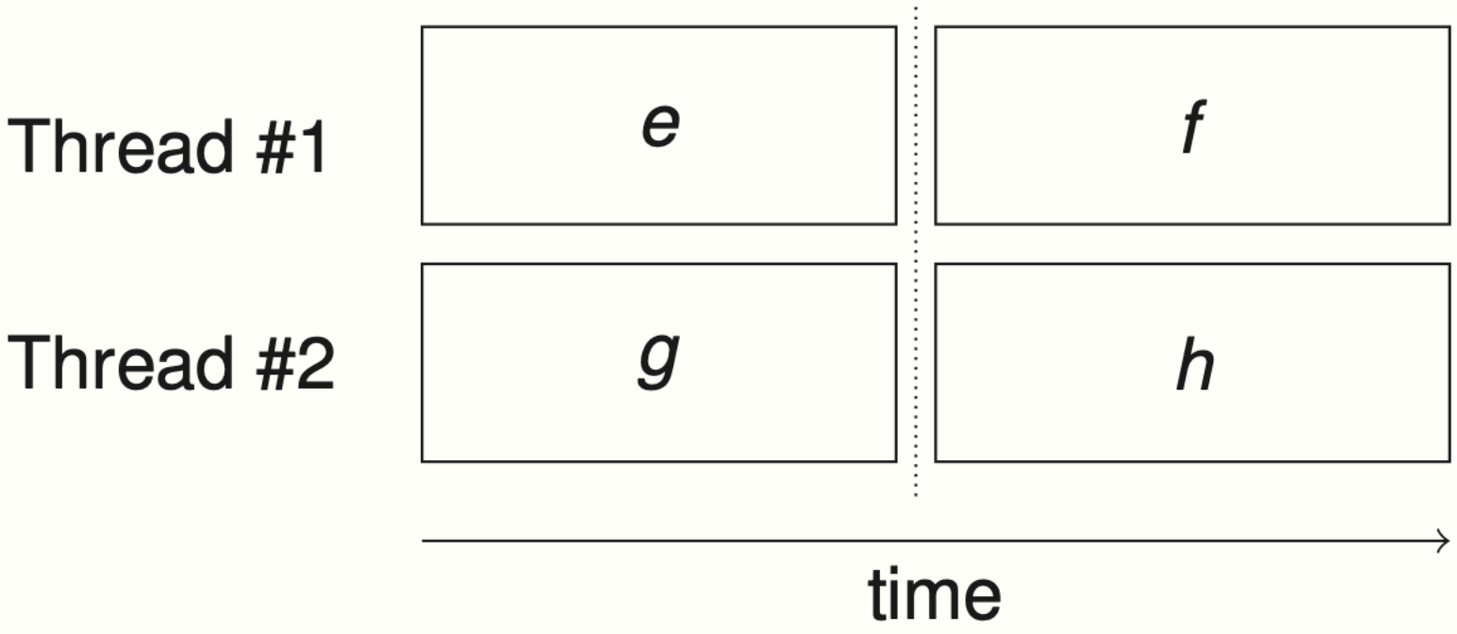
$$e \parallel f \equiv f \parallel e \qquad e \parallel (f \parallel g) \equiv (e \parallel f) \parallel g$$

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

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

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

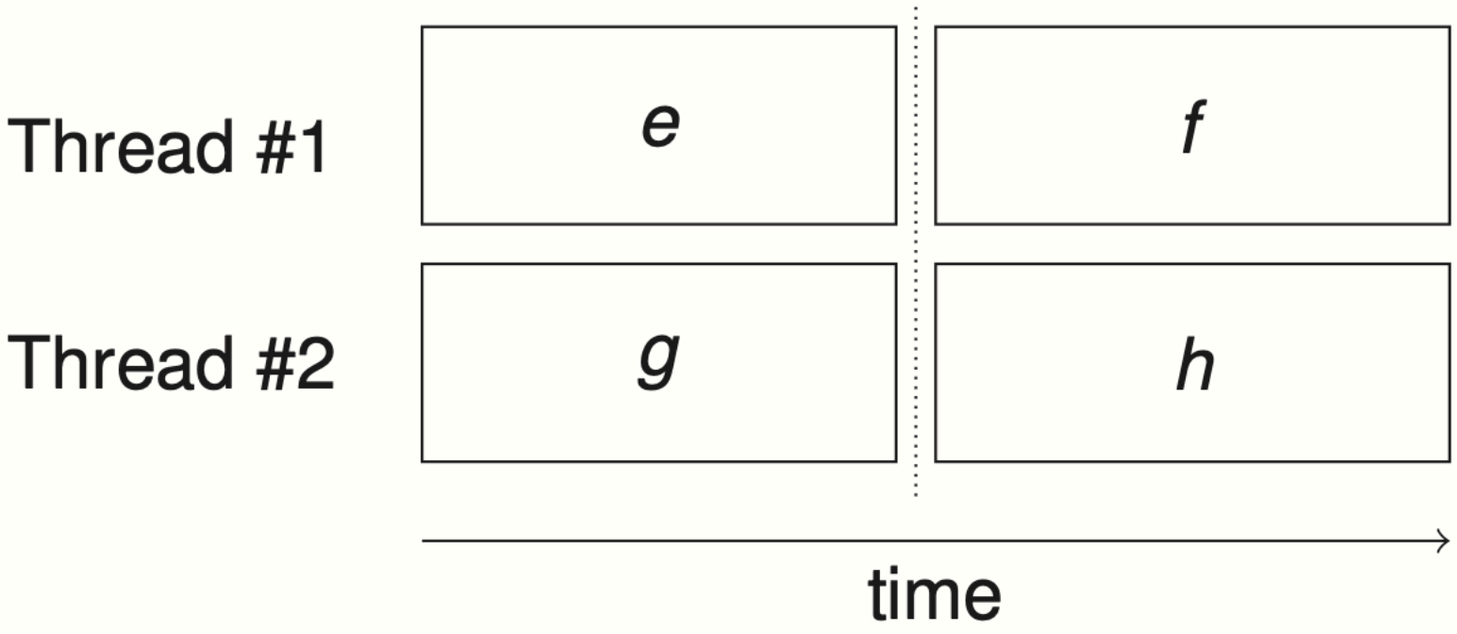
$$e \parallel 1 \equiv e \qquad e \parallel 0 \equiv 0 \qquad e \parallel (f + g) \equiv e \parallel f + e \parallel g$$



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

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

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



Equationally: $(e \parallel g) \cdot (f \parallel h) \leqq (e \cdot f) \parallel (g \cdot h)$.

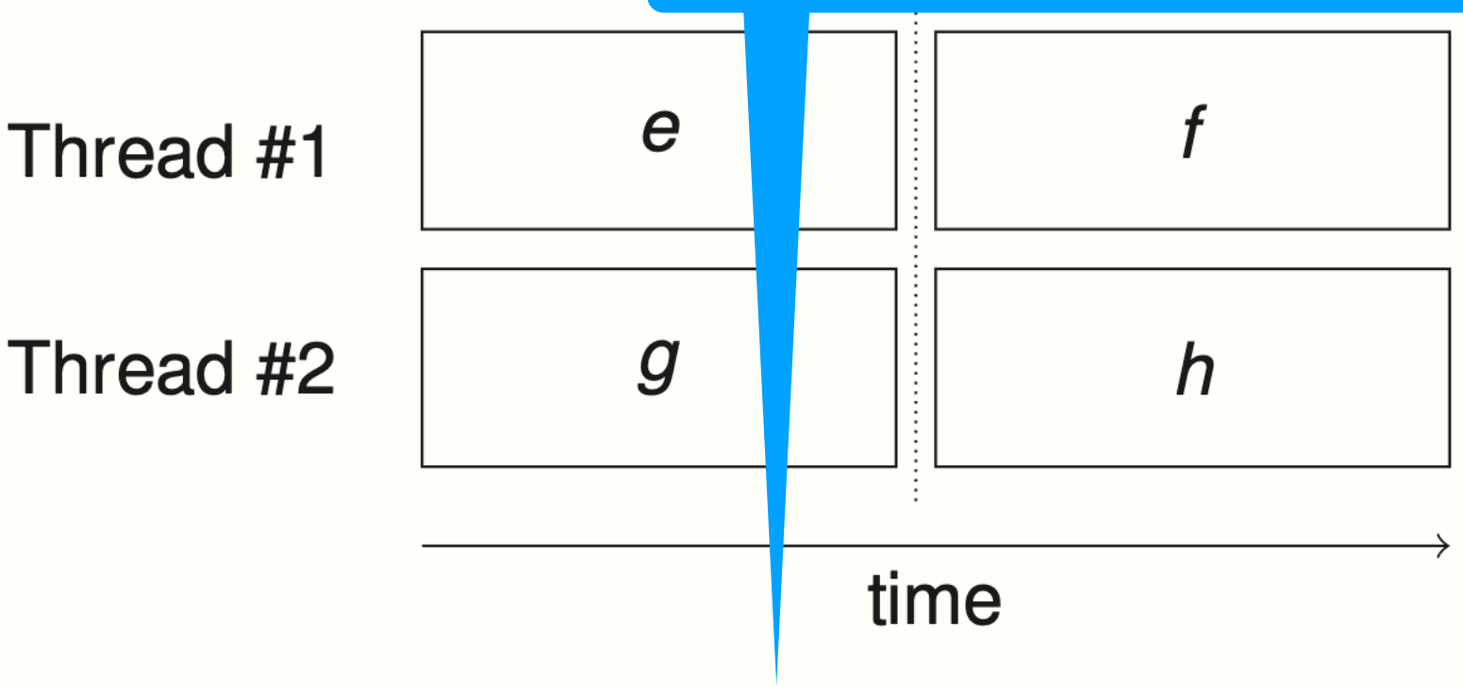
Concurrent Kleene Algebra
(Hoare, Moeller, Struth, Wehrman'09)

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

$$e \parallel 1 \equiv e \qquad e \parallel 0 \equiv$$

Exchange law interleaving as special case: e.f+f.e<=e‖f

$\parallel f + e \parallel g$



Equationally: $(e \parallel g) \cdot (f \parallel h) \leqq (e \cdot f) \parallel (g \cdot h)$.


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

GoNeCo

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].

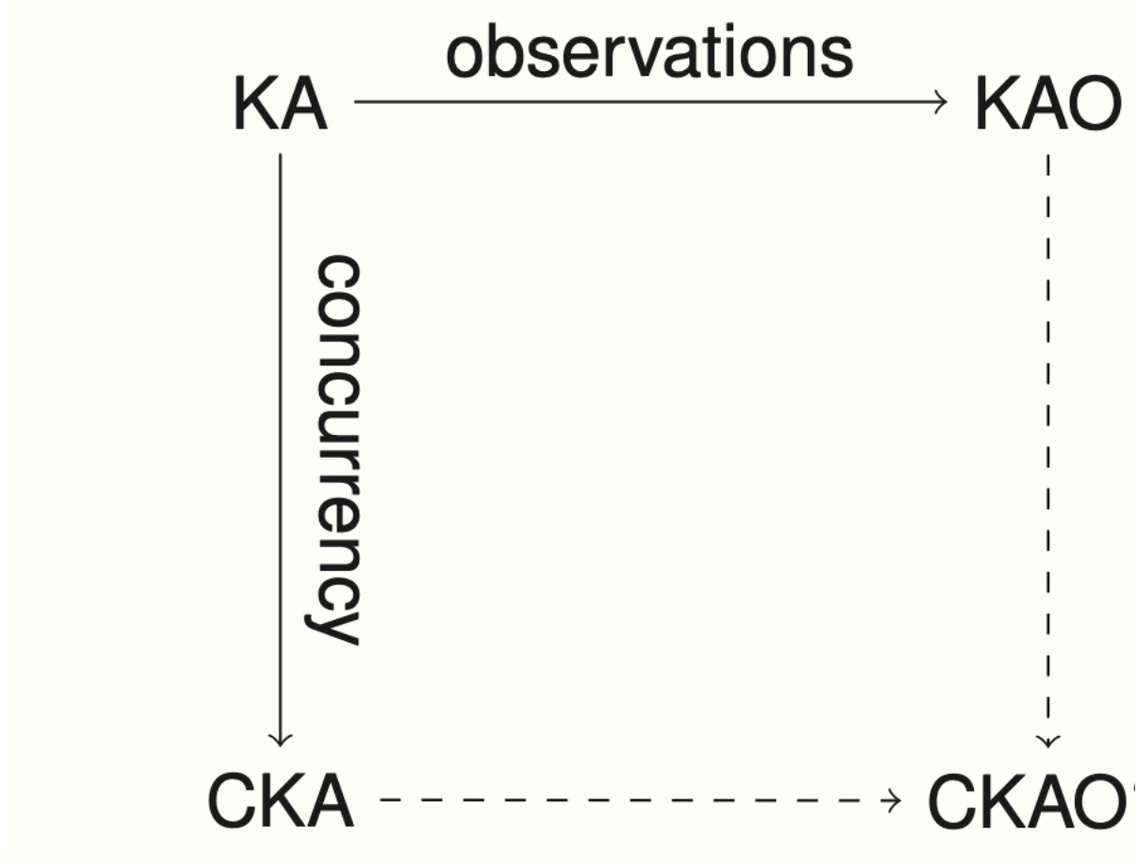
Crossroads

See also [Bergstra and Ponse 2011].

A new algebra

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

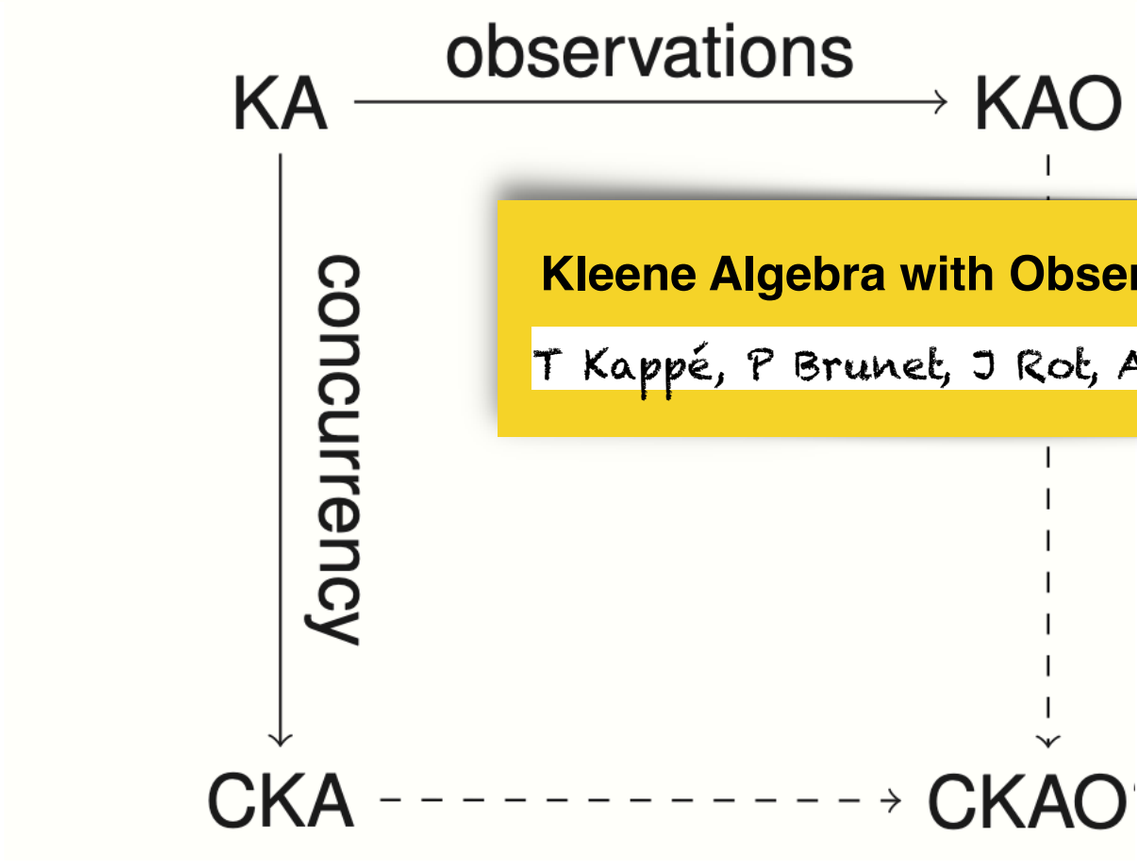
~~$1 \equiv \top$~~



A new algebra

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

~~$1 \equiv \top$~~



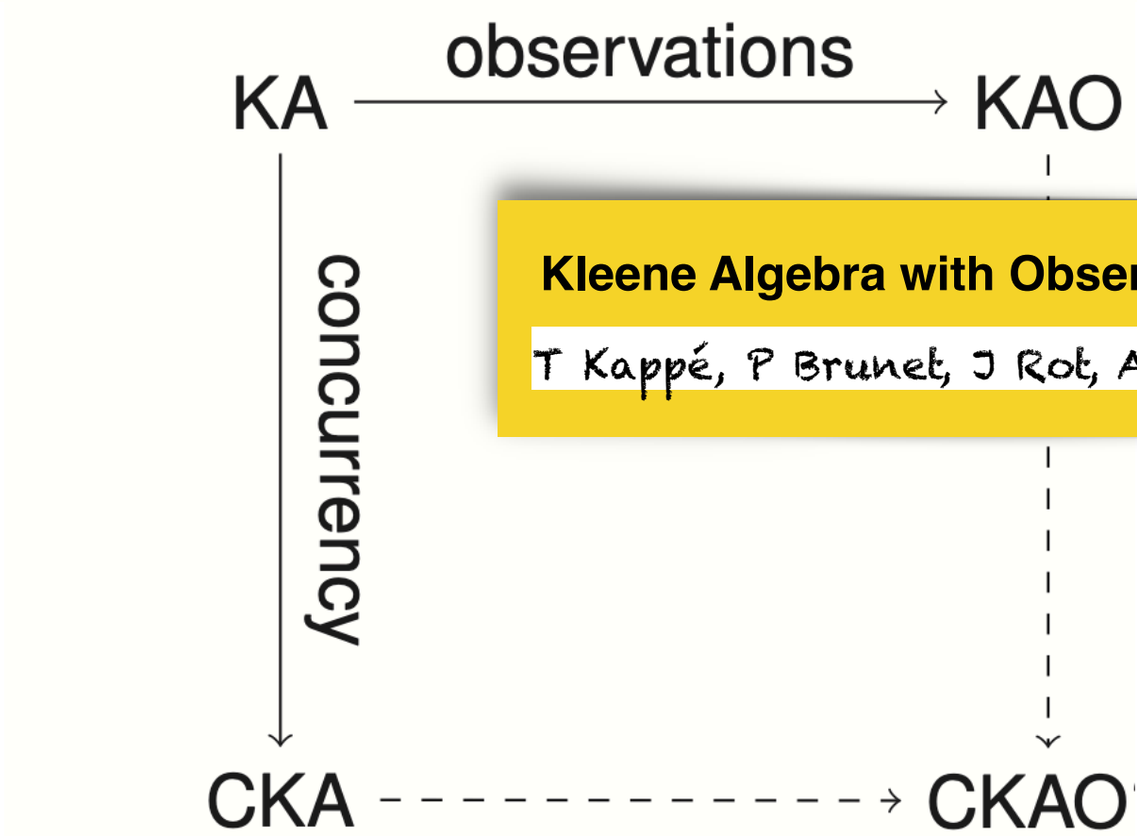
Kleene Algebra with Observations (CONCUR'19)

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?

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

Axioms
KA+BA+CKA

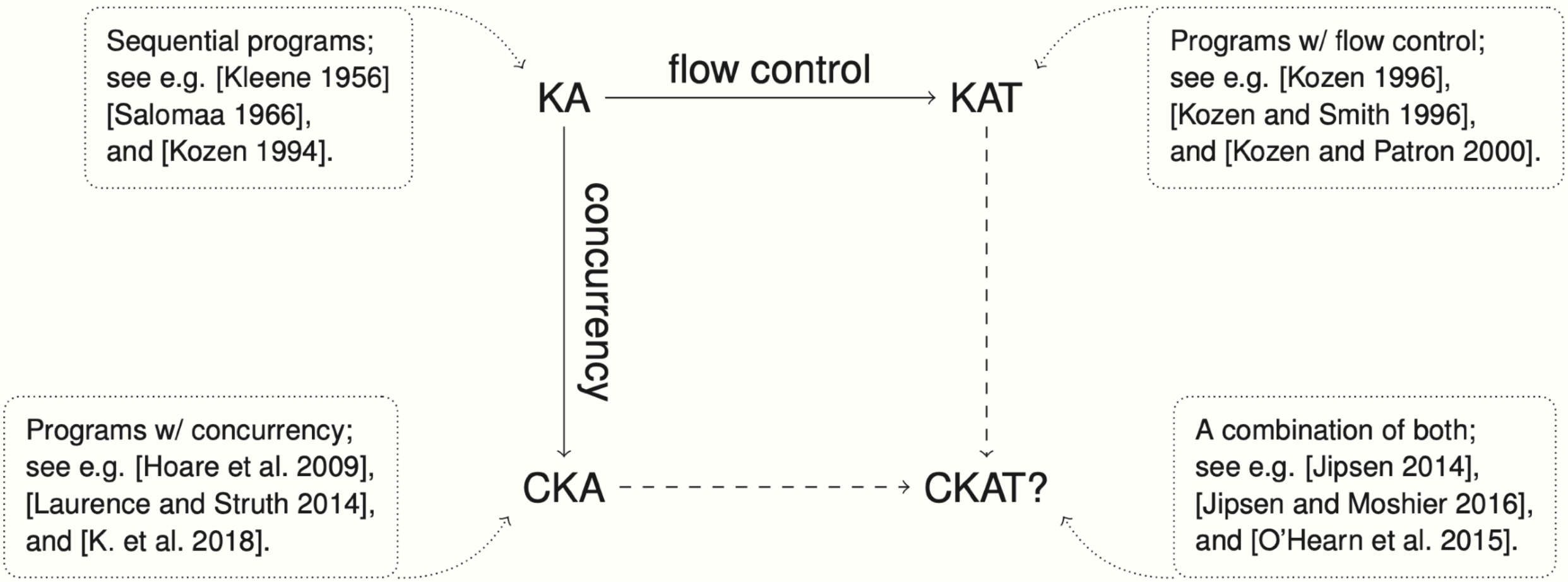
Keep conjunction and sequential composition distinct

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

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

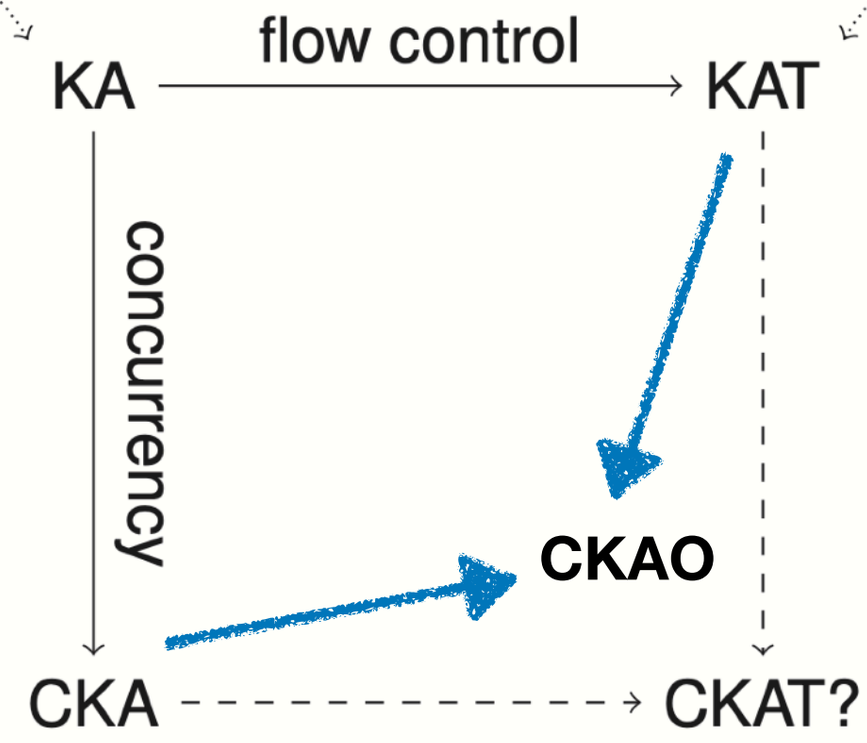
Moreover, given $e, f \in \mathcal{T}$, *it is decidable whether* $[\![e]\!]_{\mathsf{CKAO}} = [\![f]\!]_{\mathsf{CKAO}}$.

Conclusions



Conclusions

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



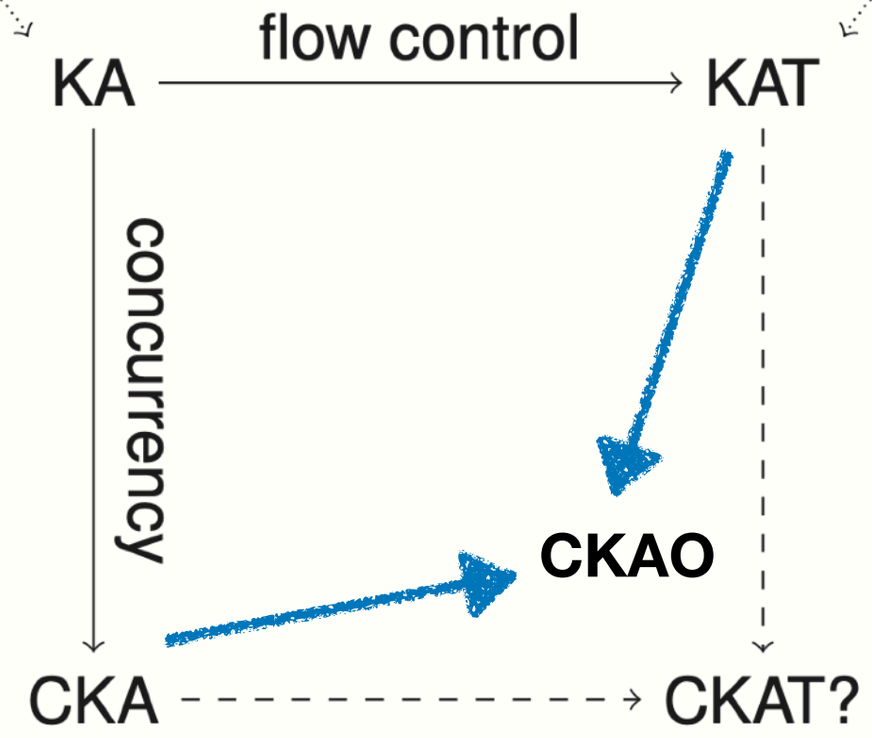
A (co)algebraic framework to analyse programs with concurrency and control flow

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

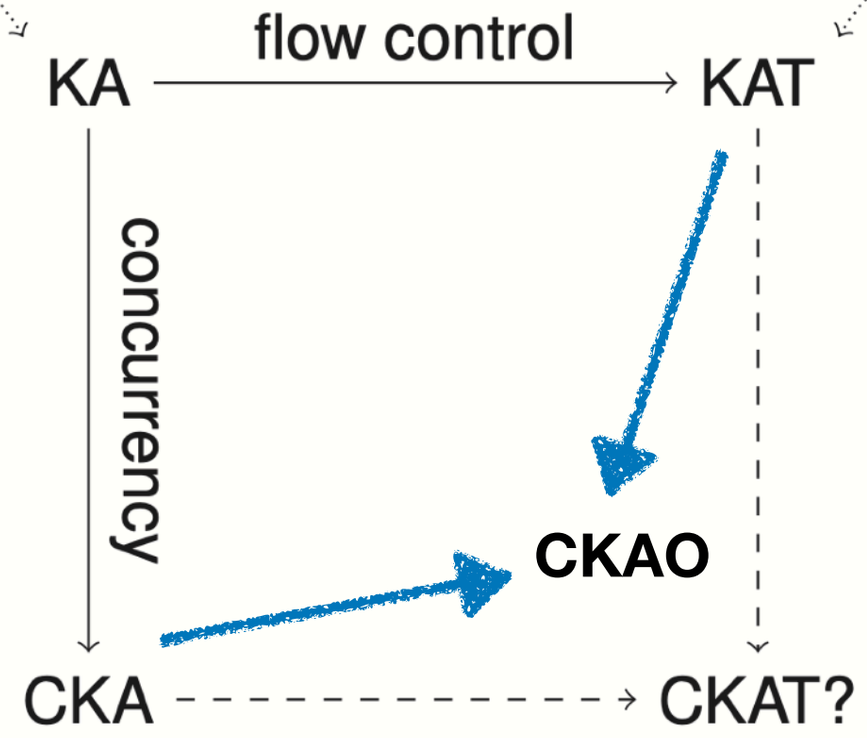
Sound and complete axiomatization
Decidability

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].

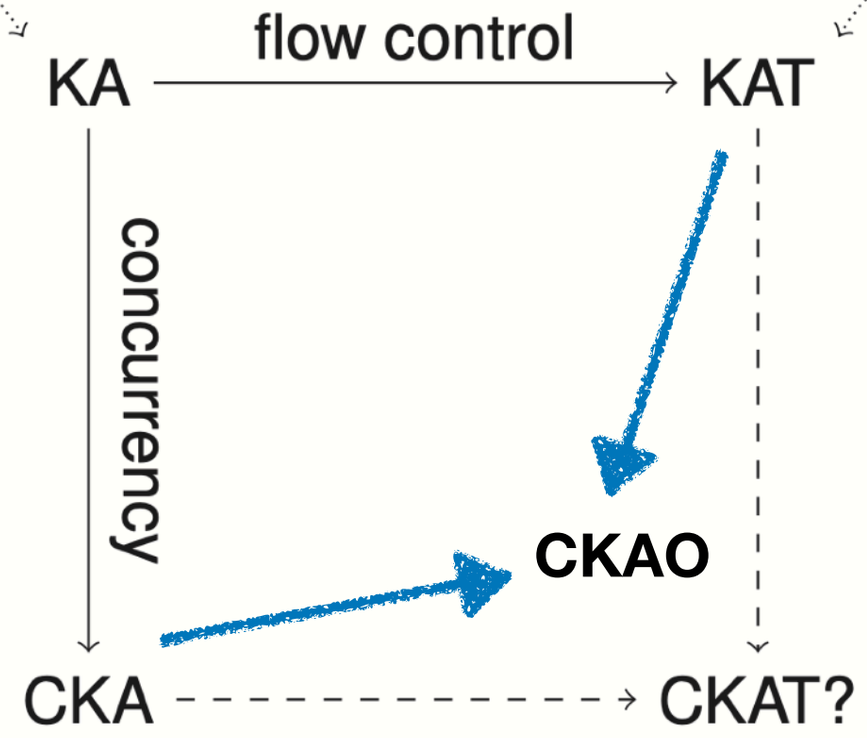
A (co)algebraic framework to analyse programs with concurrency and control flow

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

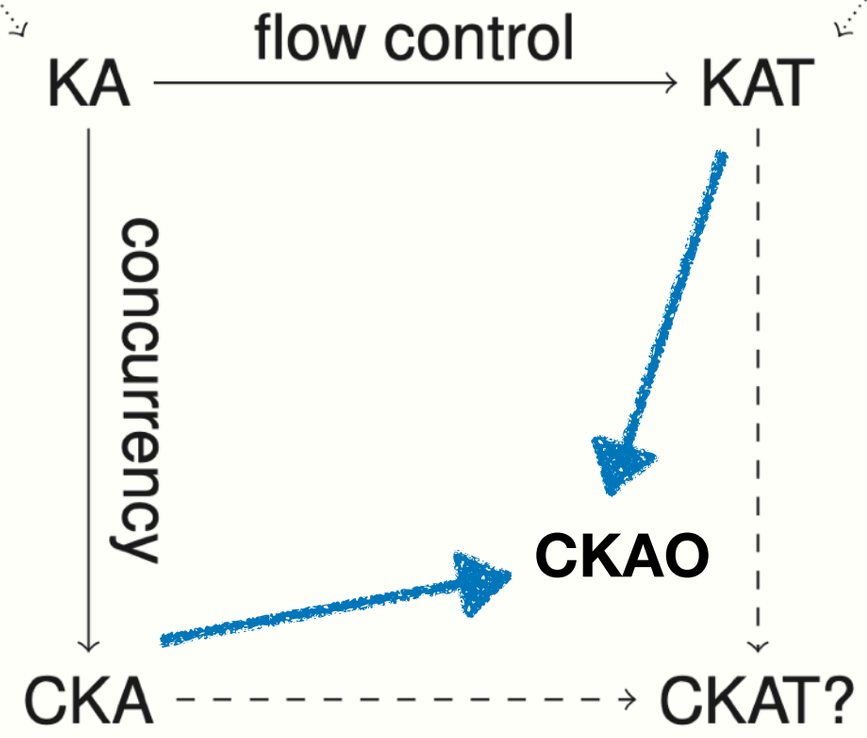
Sound and complete axiomatization
Decidability

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?