# Concurrent Kleene Algebra

Tobias Kappé

University College London

BCTCS 2018

# What is Kleene Algebra?

*Kleene Algebra describes program behavior*

# What is Kleene Algebra?

*Kleene Algebra describes program behavior*

regular expressions

# What is Kleene Algebra?

order, repetition

*Kleene Algebra describes program behavior*

regular expressions

# A tale of two programs

```
while φ1 do
    if φ2 then
        foo;
    else
        bar;
    end
end
```

```
while ψ1 do
    foo;
end
while ψ2 do
    bar;
    while ψ3 do
        foo;
    end
end
```

# A tale of two programs

```
while φ1 do
    if φ2 then
        foo;
    else
        bar;
    end
end
```


```
while ψ1 do
    foo;
end
while ψ2 do
    bar;
    while ψ3 do
        foo;
    end
end
```

# A tale of two programs

**while** $\phi_1$ **do**
    **if** $\phi_2$ **then**
        foo;
    **else**
        bar;
    **end**
**end**

$\Big\}$ $(\text{foo} + \text{bar})^*$


**while** $\psi_1$ **do**
    foo;
**end**
**while** $\psi_2$ **do**
    bar;
    **while** $\psi_3$ **do**
        foo;
    **end**
**end**

# A tale of two programs

```
while φ1 do
    if φ2 then
        foo;
    else
        bar;
    end
end
```

$(\text{foo} + \text{bar})^*$

```
while ψ1 do
    foo;
end
while ψ2 do
    bar;
    while ψ3 do
        foo;
    end
end
```

$\text{foo}^* \cdot (\text{bar} \cdot \text{foo}^*)^*$

# A tale of two programs

We can prove this using KA:

$$(\mathsf{foo} + \mathsf{bar})^* \equiv_{\mathsf{KA}} \mathsf{foo}^* \cdot (\mathsf{bar} \cdot \mathsf{foo}^*)^*$$

where $\equiv_{\mathsf{KA}}$ is generated by axioms such as (among others)

$$e + e \equiv_{\mathsf{KA}} e \qquad e \cdot 1 \equiv_{\mathsf{KA}} e \qquad e^* \equiv_{\mathsf{KA}} 1 + e \cdot e^*$$

# The lay of the land

KA is well-understood:

# The lay of the land

KA is well-understood:

**Theorem (Salomaa 1966; Kozen 1994)**

KA *axiomatizes regex-equivalence, i.e.,* $e \equiv_{\mathsf{KA}} f \Leftrightarrow \mathcal{L}(e) = \mathcal{L}(f)$.

# The lay of the land

KA is well-understood:

**Theorem (Salomaa 1966; Kozen 1994)**

KA *axiomatizes regex-equivalence, i.e.,* $e \equiv_{\mathsf{KA}} f \Leftrightarrow \mathcal{L}(e) = \mathcal{L}(f)$.

**Theorem (Kleene 1956; Brzozowski 1964)**

*Every regex is equivalent to some finite automaton, and vice versa.*

# Towards concurrency

| Thread 1 | Thread 2 |
|---|---|
| $a$ | $c$ |
| $b$ | $d$ |

How do we model concurrent composition in KA?

# Towards concurrency

| Thread 1 | Thread 2 |
|---|---|
| $a$ | $c$ |
| $b$ | $d$ |

$$\underbrace{\qquad\qquad}_{a \cdot b \cdot c \cdot d + a \cdot c \cdot b \cdot d + \cdots ?}$$

Interleaving is insufficient!

# Towards concurrency

| Thread 1 | Thread 2 |
| --- | --- |
| $a$ | $c$ |
| $b$ | $d$ |

$$\underbrace{\qquad\qquad}_{(a \cdot b) \parallel (c \cdot d)}$$

Concurrent KA[a] adds *parallel composition* ($\parallel$)

- expressions grow linearly with the program
- interleaving still possible: $(e \parallel f) \cdot (g \parallel h) \leqq_{\mathsf{CKA}} (e \cdot g) \parallel (f \cdot h)$.

[a]Hoare et al. 2009.

# Questions begged

Enquiring minds want to know:

# Questions begged

Enquiring minds want to know:

## Question

*Does* CKA *axiomatize "concurrent regex" equivalence, i.e.,* $e \equiv_{\mathsf{CKA}} f \Leftrightarrow \mathcal{L}_{\parallel}(e) = \mathcal{L}_{\parallel}(f)$*?*

# Questions begged

Enquiring minds want to know:

**Question**

*Does* CKA *axiomatize "concurrent regex" equivalence, i.e.,* $e \equiv_{\mathsf{CKA}} f \Leftrightarrow \mathcal{L}_{\|}(e) = \mathcal{L}_{\|}(f)$*?*

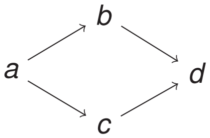
**Question**

*Is there an automaton model that corresponds to concurrent regular expressions?*

# Interlude: *partially ordered multisets*

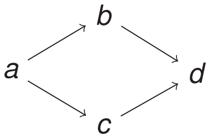
A *pomset* is a “word with parallelism”

$$a \cdot (b \parallel c) \cdot d \approx$$



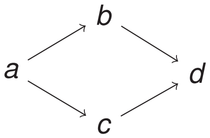
# Interlude: *partially ordered multisets*

A *pomset* is a "word with parallelism"

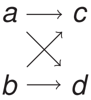
$a \cdot (b \parallel c) \cdot d \approx$ 

# Interlude: *partially ordered multisets*

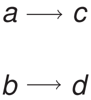
A *pomset* is a “word with parallelism”

$a \cdot (b \parallel c) \cdot d \approx$ 


Pomset subsumption:

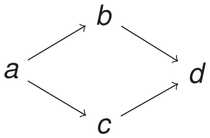



$\sqsubseteq$



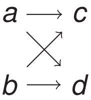
A *pomset* is a “word with parallelism”

$$a \cdot (b \parallel c) \cdot d \approx$$

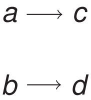



Pomset subsumption:

$$(a \parallel b) \cdot (c \parallel d) \approx$$




$$\sqsubseteq$$




$$\approx (a \cdot c) \parallel (b \cdot d)$$

# Interlude: *partially ordered multisets*

Composition lifts to pomset languages:

- $\mathcal{U} \cdot \mathcal{V} = \{U \cdot V : U \in \mathcal{U}, V \in \mathcal{V}\}$
- $\mathcal{U} \parallel \mathcal{V} = \{U \parallel V : U \in \mathcal{U}, V \in \mathcal{V}\}$

# Interlude: *partially ordered multisets*

Composition lifts to pomset languages:

- $\mathcal{U} \cdot \mathcal{V} = \{U \cdot V : U \in \mathcal{U}, V \in \mathcal{V}\}$
- $\mathcal{U} \parallel \mathcal{V} = \{U \parallel V : U \in \mathcal{U}, V \in \mathcal{V}\}$

Kleene star: $\mathcal{U}^* = \bigcup_{n<\omega} \mathcal{U}^n$

# Interlude: *partially ordered multisets*

Composition lifts to pomset languages:

- $\mathcal{U} \cdot \mathcal{V} = \{U \cdot V : U \in \mathcal{U}, V \in \mathcal{V}\}$
- $\mathcal{U} \parallel \mathcal{V} = \{U \parallel V : U \in \mathcal{U}, V \in \mathcal{V}\}$

Kleene star: $\mathcal{U}^* = \bigcup_{n<\omega} \mathcal{U}^n$

Closure: $\mathcal{U}\downarrow = \{U' \in \mathrm{Pom}_\Sigma : U' \sqsubseteq U \in \mathcal{U}\}$.

# Axiomatization

CKA *semantics* is given by $\llbracket - \rrbracket_{\mathsf{CKA}} : \mathcal{T} \to 2^{\mathsf{Pom}_\Sigma}$.

$$\llbracket 0 \rrbracket_{\mathsf{CKA}} = \emptyset \qquad \llbracket e + f \rrbracket_{\mathsf{CKA}} = \llbracket e \rrbracket_{\mathsf{CKA}} \cup \llbracket f \rrbracket_{\mathsf{CKA}} \qquad \llbracket e^* \rrbracket_{\mathsf{CKA}} = \llbracket e \rrbracket_{\mathsf{CKA}}^{*}\downarrow$$

$$\llbracket 1 \rrbracket_{\mathsf{CKA}} = \{1\} \qquad \llbracket e \cdot f \rrbracket_{\mathsf{CKA}} = \llbracket e \rrbracket_{\mathsf{CKA}} \cdot \llbracket f \rrbracket_{\mathsf{CKA}}$$

$$\llbracket a \rrbracket_{\mathsf{CKA}} = \{a\} \qquad \llbracket e \parallel f \rrbracket_{\mathsf{CKA}} = (\llbracket e \rrbracket_{\mathsf{CKA}} \parallel \llbracket f \rrbracket_{\mathsf{CKA}})\downarrow$$

CKA *axioms* is given by axioms of KA, plus

$$e \parallel f \equiv_{\mathsf{CKA}} f \parallel e \qquad 0 \parallel f \equiv_{\mathsf{CKA}} 0 \qquad 1 \parallel f \equiv_{\mathsf{CKA}} f$$

$$(e + f) \parallel g \equiv_{\mathsf{CKA}} e \parallel g + f \parallel g \qquad e \parallel (f \parallel g) = (e \parallel f) \parallel g$$

$$(e \parallel f) \cdot (g \parallel h) \leqq_{\mathsf{CKA}} (e \cdot g) \parallel (f \cdot h)$$

# Axiomatization

## Theorem (Kappé et al. 2018)

*The axioms for* CKA *are sound and complete for semantic equivalence:*

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

# Axiomatization

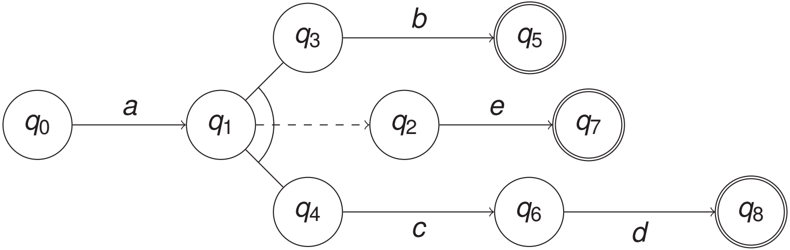
**Theorem (Kappé et al. 2018)**

*The axioms for* CKA *are sound and complete for semantic equivalence:*

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

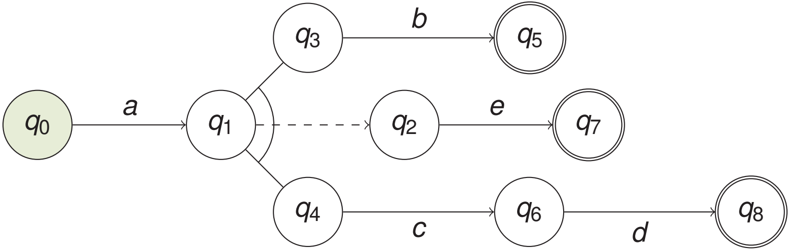
**Question**

*What happens when we add the "parallel Kleene star"?*

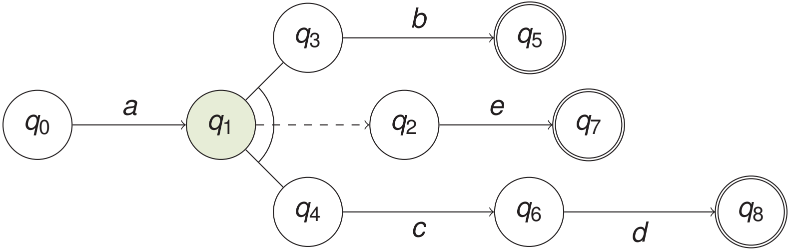


$q_0$
a
$q_1$
$q_3$
b
$q_5$
$q_2$
e
$q_7$
$q_4$
c
$q_6$
d
$q_8$

# Automata model

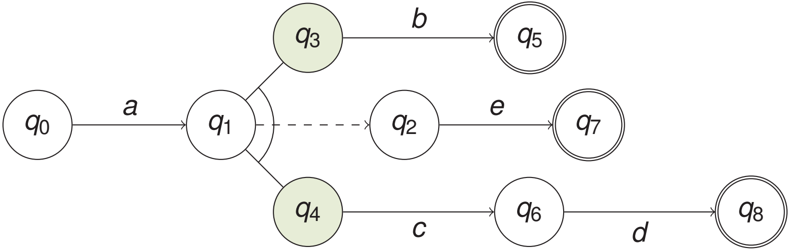


# Automata model



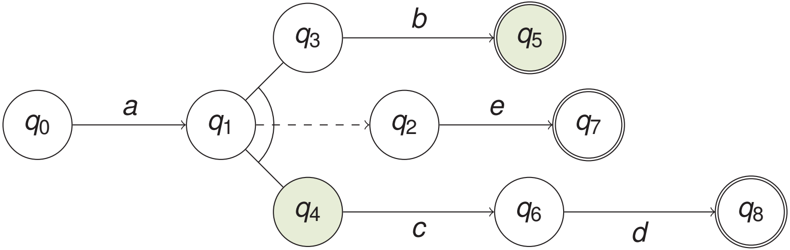

$a$

# Automata model



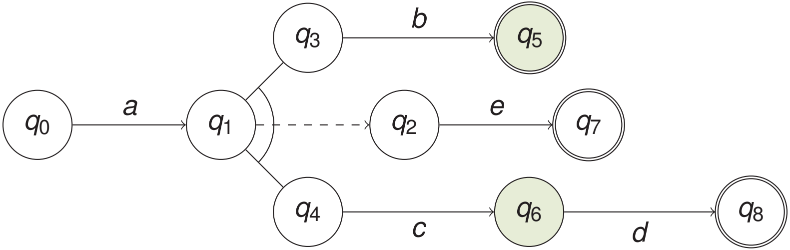

*a*

# Automata model



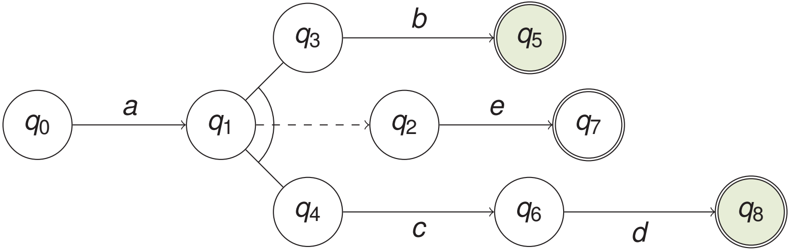

$a$

# Automata model



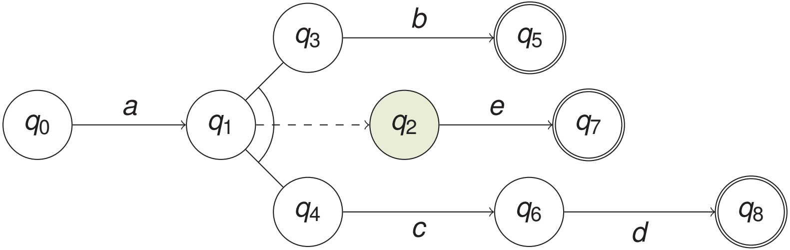

$a$

# Automata model



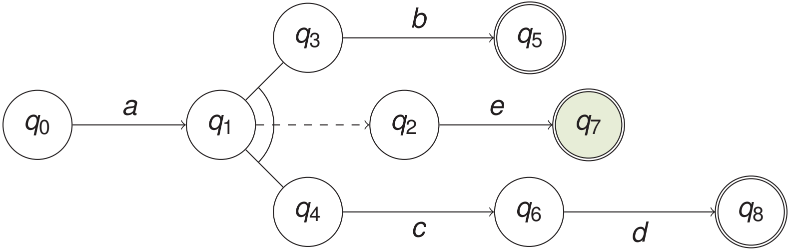

$a$

# Automata model



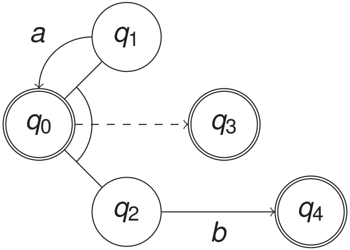

$$a \cdot (b \parallel c \cdot d)$$

# Automata model

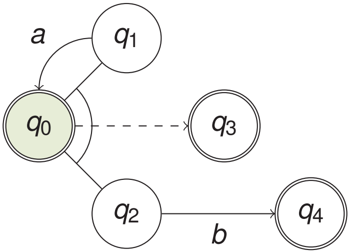



$$a \cdot (b \parallel c \cdot d) \cdot e$$

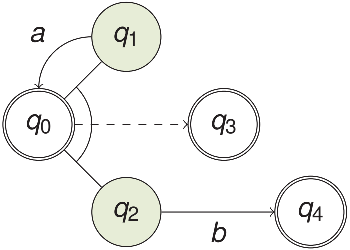
# Automata model



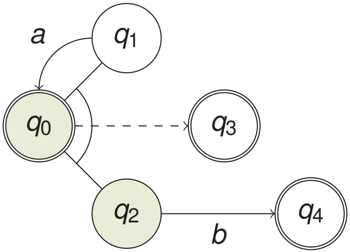
# Automata model



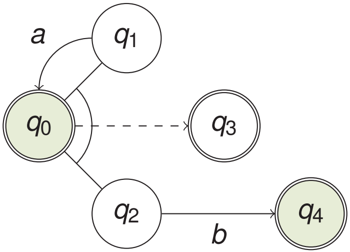
# Automata model



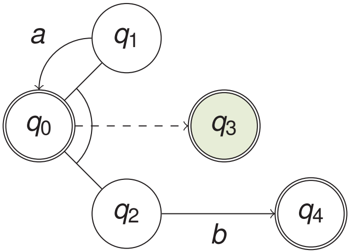
# Automata model



# Automata model

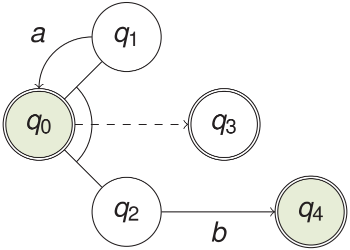


# Automata model

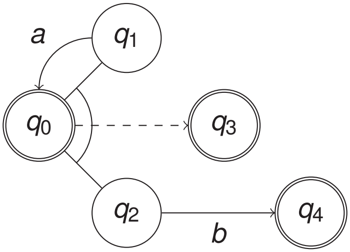



$a \parallel b$

# Automata model

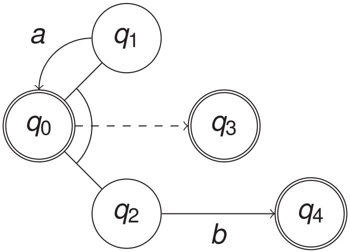


# Automata model



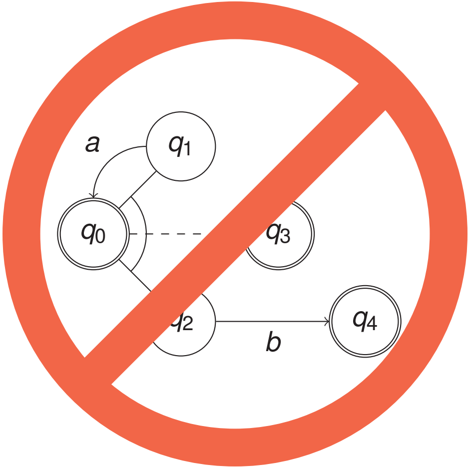

$$a \cdot (a \parallel b) \parallel b$$

# Automata model




$$a \cdot (a \cdot (a \parallel b)) \parallel b$$

# Automata model



# Automata model

**Theorem (K. et al. 2017)**

*The following are equivalent:*

i. $\mathfrak{U}$ *is described by a concurrent regex*
ii. $\mathfrak{U}$ *is recognized by a fork-acyclic pomset automaton.*

# Further work

**Question**

KA *can be described coalgebraically; what about* CKA*?*

# Further work

**Question**

KA *can be described coalgebraically; what about* CKA*?*

**Question**

*Is equivalence of pomset-automata (tractably) decidable?*

# Further work

**Question**

KA *can be described coalgebraically; what about* CKA*?*

**Question**

*Is equivalence of pomset-automata (tractably) decidable?*

**Question**

*NetKAT can be used to describe network policy. Can we add concurrency?*

Thank you for your attention

GoNeCo

Code: `https://doi.org/10.5281/zenodo.926651`.

Illustrations adapted from `https://xkcd.com/208/` (CC-BY-NC)