

# 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  $\phi_1$  do
  | if  $\phi_2$  then
    |   foo;
    | else
    |   bar;
    | end
  | end
end
```

```
while  $\psi_1$  do
  |   foo;
end
while  $\psi_2$  do
  |   bar;
  |   while  $\psi_3$  do
    |   foo;
    | end
  | end
end
```

# A tale of two programs

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

```
while  $\psi_1$  do
  | foo;
end
while  $\psi_2$  do
  | bar;
  | while  $\psi_3$  do
    |   foo;
    | end
  | end
end
```

EVERYBODY STAND BACK.

# A tale of two programs

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

} ( $\text{foo} + \text{bar}$ )<sup>\*</sup>

```
while  $\psi_1$  do  
  | foo;  
end  
while  $\psi_2$  do  
  | bar;  
  | while  $\psi_3$  do  
  |   | foo;  
  |   end  
end
```

I KNOW REGULAR  
EXPRESSIONS.

# A tale of two programs

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

$(foo + bar)^*$

```
while  $\psi_1$  do  
  | foo;  
end  
while  $\psi_2$  do  
  | bar;  
  | while  $\psi_3$  do  
  |   | foo;  
  |   end  
end
```

$foo^* \cdot (bar \cdot foo^*)^*$

# A tale of two programs

We can prove this using KA:

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

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

$$e + e \equiv_{\text{KA}} e$$

$$e \cdot 1 \equiv_{\text{KA}} e$$

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



# 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.,  $e \equiv_{\text{KA}} f \Leftrightarrow \mathcal{L}(e) = \mathcal{L}(f)$ .

# The lay of the land

KA is well-understood:

Theorem (Salomaa 1966; Kozen 1994)

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

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 |
|----------|----------|
| $a$      | $c$      |
| $b$      | $d$      |

$a \cdot b \cdot c \cdot d + a \cdot c \cdot b \cdot d + \dots ?$

Interleaving is insufficient!

# Towards concurrency

| Thread 1                            | Thread 2 |
|-------------------------------------|----------|
| $a$                                 | $c$      |
| $b$                                 | $d$      |
| $(a \cdot b) \parallel (c \cdot d)$ |          |

Concurrent KA<sup>a</sup> adds *parallel composition* ( $\parallel$ )

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

---

<sup>a</sup>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.,  $e \equiv_{\text{CKA}} f \Leftrightarrow \mathcal{L}_{\parallel}(e) = \mathcal{L}_{\parallel}(f)$ ?*

# Questions begged

Enquiring minds want to know:

## Question

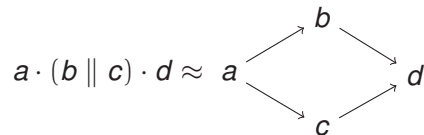
*Does CKA axiomatize “concurrent regex” equivalence, i.e.,  $e \equiv_{\text{CKA}} f \Leftrightarrow \mathcal{L}_{\parallel}(e) = \mathcal{L}_{\parallel}(f)$ ?*

## Question

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

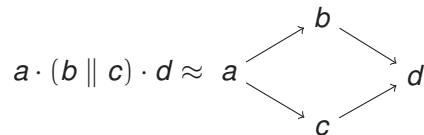
## Interlude: *partially ordered multisets*

A *pomset* is a “word with parallelism”



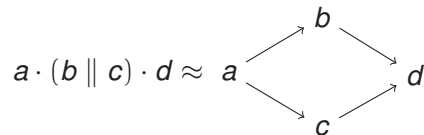
## Interlude: *partially ordered multisets*

A *pomset* is a “word with parallelism”

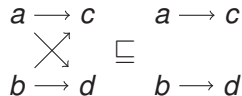


## Interlude: *partially ordered multisets*

A *pomset* is a “word with parallelism”

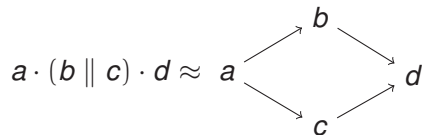


Pomset subsumption:



## Interlude: *partially ordered multisets*

A *pomset* is a “word with parallelism”



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

# Axiomatization

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

$$\llbracket 0 \rrbracket_{\text{CKA}} = \emptyset$$

$$\llbracket e + f \rrbracket_{\text{CKA}} = \llbracket e \rrbracket_{\text{CKA}} \cup \llbracket f \rrbracket_{\text{CKA}}$$

$$\llbracket e^* \rrbracket_{\text{CKA}} = \llbracket e \rrbracket_{\text{CKA}}^* \downarrow$$

$$\llbracket 1 \rrbracket_{\text{CKA}} = \{1\}$$

$$\llbracket e \cdot f \rrbracket_{\text{CKA}} = \llbracket e \rrbracket_{\text{CKA}} \cdot \llbracket f \rrbracket_{\text{CKA}}$$

$$\llbracket a \rrbracket_{\text{CKA}} = \{a\}$$

$$\llbracket e \parallel f \rrbracket_{\text{CKA}} = (\llbracket e \rrbracket_{\text{CKA}} \parallel \llbracket f \rrbracket_{\text{CKA}}) \downarrow$$

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

$$e \parallel f \equiv_{\text{CKA}} f \parallel e$$

$$0 \parallel f \equiv_{\text{CKA}} 0$$

$$1 \parallel f \equiv_{\text{CKA}} f$$

$$(e + f) \parallel g \equiv_{\text{CKA}} e \parallel g + f \parallel g$$

$$e \parallel (f \parallel g) = (e \parallel f) \parallel g$$

$$(e \parallel f) \cdot (g \parallel h) \leq_{\text{CKA}} (e \cdot g) \parallel (f \cdot h)$$

## Theorem (K. et al. 2018)

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

$$e \equiv_{\text{CKA}} f \Leftrightarrow \llbracket e \rrbracket_{\text{CKA}} = \llbracket f \rrbracket_{\text{CKA}}$$

## Theorem (K. et al. 2018)

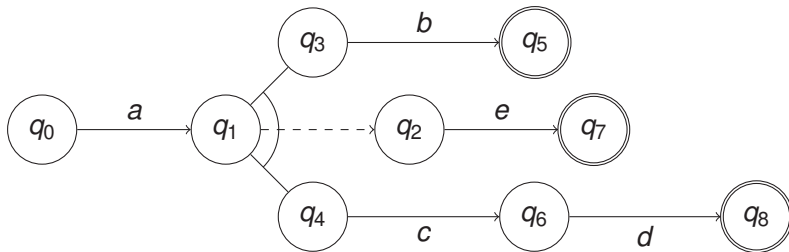
*The axioms for CKA are sound and complete for semantic equivalence:*

$$e \equiv_{\text{CKA}} f \Leftrightarrow \llbracket e \rrbracket_{\text{CKA}} = \llbracket f \rrbracket_{\text{CKA}}$$

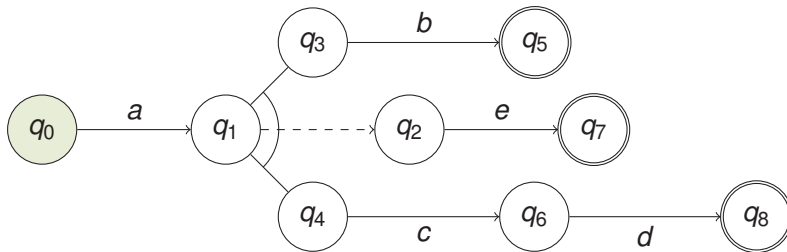
## Question

*What happens when we add the “parallel Kleene star”?*

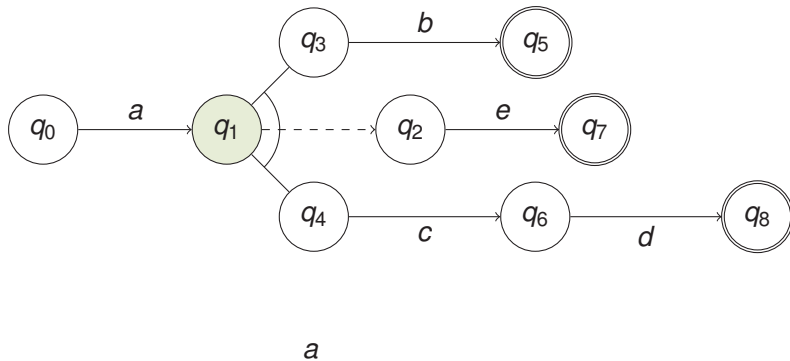
# Automata model



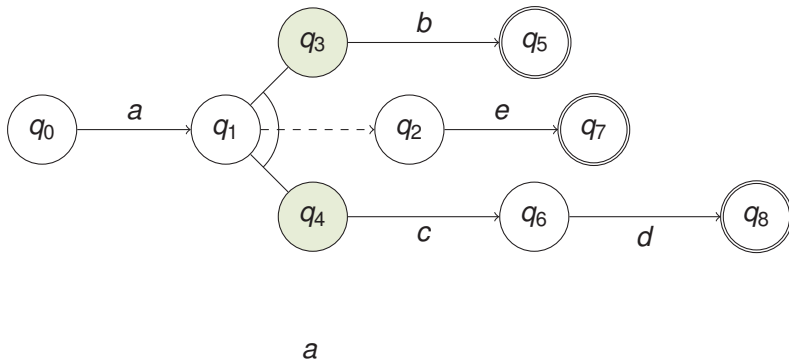
# Automata model



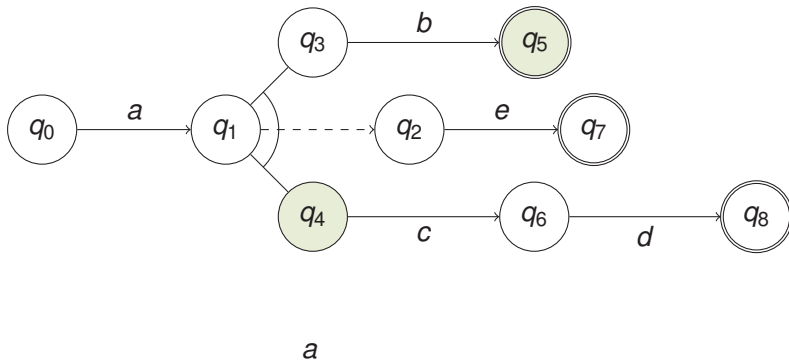
# Automata model



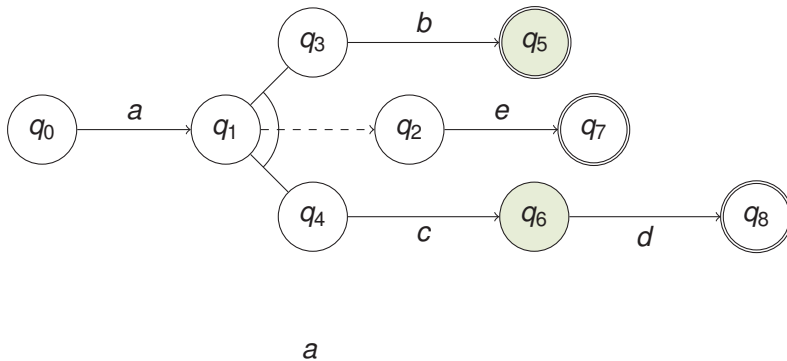
# Automata model



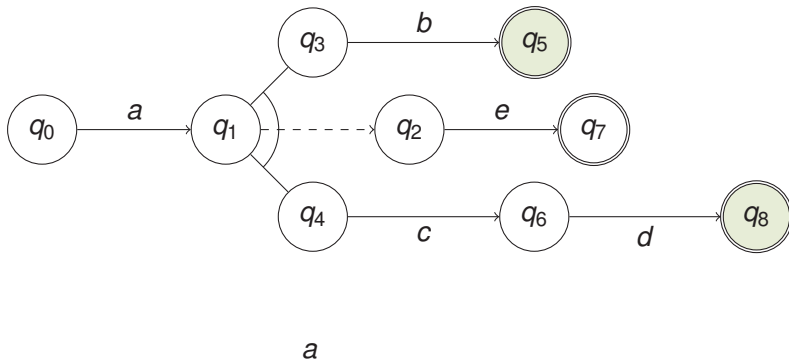
# Automata model



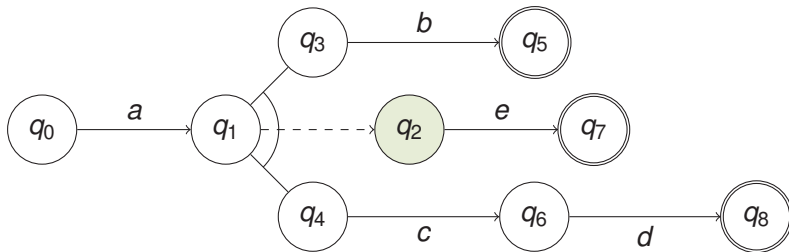
# Automata model



# Automata model

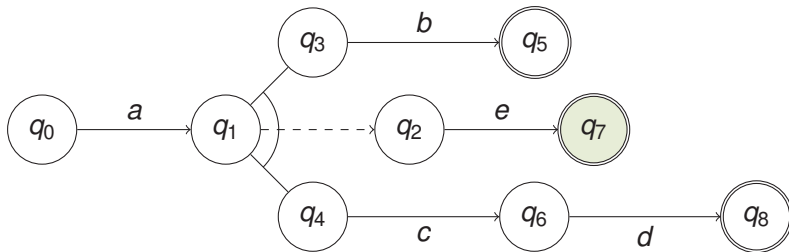


# Automata model



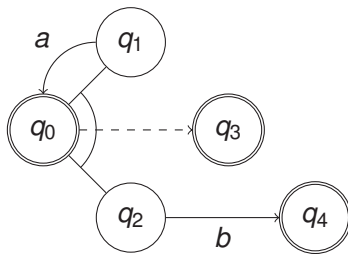
$$a \cdot (b \parallel c \cdot d)$$

# Automata model

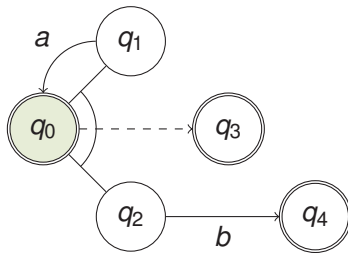


$$a \cdot (b \parallel c \cdot d) \cdot e$$

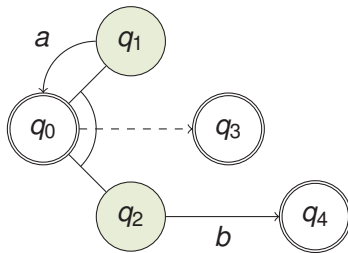
# Automata model



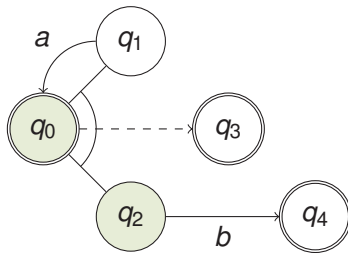
# Automata model



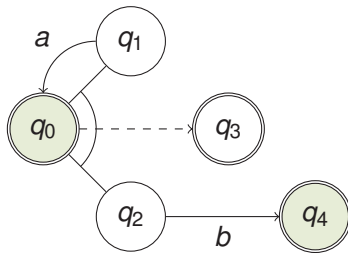
# Automata model



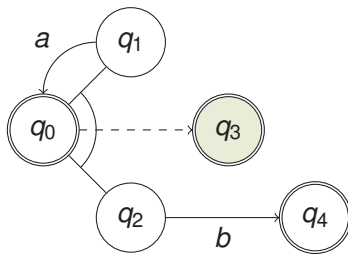
# Automata model



# Automata model

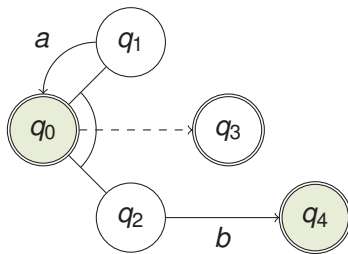


# Automata model

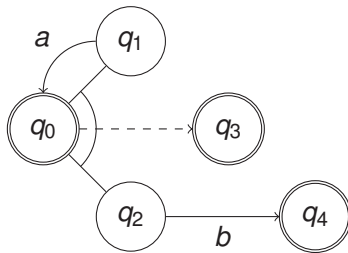


$a \parallel b$

# Automata model

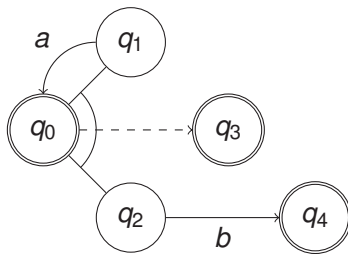


# Automata model



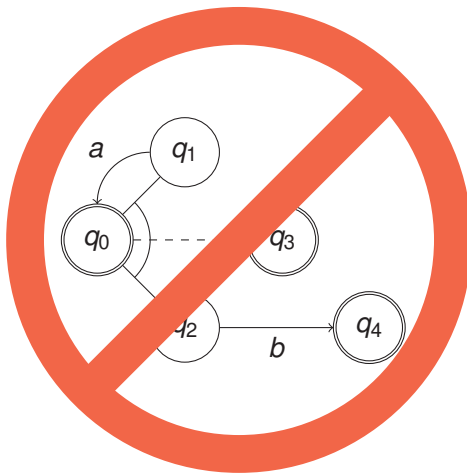
$$a \cdot (a \parallel b) \parallel b$$

# Automata model



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

# Automata model



## Theorem (K. et al. 2017)

*The following are equivalent:*

- i  $\mathcal{U}$  is described by a concurrent regex
- ii  $\mathcal{U}$  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



Code: <https://doi.org/10.5281/zenodo.926651>.

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