Recent developments

in

concurrent Kleene algebra

IRIS Seminar - London

December 2020

Paul Brunet
University College London

Concurrent Kleene Algebra

Concurrent Kleene Algebra

C.A.R. Tony Hoare[1], Bernhard Möller[2], Georg Struth[3], and Ian Wehrman[4]

[1] Microsoft Research, Cambridge, UK
[2] Universität Augsburg, Germany
[3] University of Sheffield, UK
[4] University of Texas at Austin, USA

2009

CKA is introduced.

Concurrent Kleene Algebra

On Locality and the Exchange Law for Concurrent Processes

C.A.R. Hoare[1], Akbar Hussain[2], Bernhard Möller[3], Peter W. O'Hearn[2], Rasmus Lerchedahl Petersen[2], and Georg Struth[4]

[1] Microsoft Research Cambridge
[2] Queen Mary University of London
[3] Universität Augsburg
[4] University of Sheffield

2009

CKA is introduced.

2011

Models of CKA are introduced, and the relationship with separation logic is established.

Concurrent Kleene Algebra

Completeness Theorems for Bi-Kleene Algebras and Series-Parallel Rational Pomset Languages

Michael R. Laurence and Georg Struth

Department of Computer Science, University of Sheffield, UK
{m.laurence,g.struth}@sheffield.ac.uk

Concurrent Kleene Algebra with Tests

Peter Jipsen

Chapman University, Orange, California 92866, USA
jipsen@chapman.edu

- **2009**: CKA is introduced.
- **2011**: Models of CKA are introduced, and the relationship with separation logic is established.
- **2014**: First completeness theorem (without the exchange law), CKA with tests is introduced.

Concurrent Kleene Algebra

Concurrent Kleene algebra with tests and branching automata

Peter Jipsen*, M. Andrew Moshier

Chapman University, Orange, CA 92866, USA

2009 — CKA is introduced.

2011 — Models of CKA are introduced, and the relationship with separation logic is established.

2014 — First completeness theorem (without the exchange law), CKA with tests is introduced.

2016 — Second paper on CKAT, correcting some mistakes from the first one.

Concurrent Kleene Algebra

On Decidability of Concurrent Kleene Algebra*†

Paul Brunet[1], Damien Pous[2], and Georg Struth[3]

1 Univ. Lyon, CNRS, ENS de Lyon, UCB Lyon 1, LIP, France
2 Univ. Lyon, CNRS, ENS de Lyon, UCB Lyon 1, LIP, France
3 Department of Computer Science, The University of Sheffield, UK

2009 2011 2014 2016 2017

- 2009: CKA is introduced.
- 2011: Models of CKA are introduced, and the relationship with separation logic is established.
- 2014: First completeness theorem (without the exchange law), CKA with tests is introduced.
- 2016: Second paper on CKAT, correcting some mistakes from the first one.
- 2017: Decidability of CKA.

Concurrent Kleene Algebra

Concurrent Kleene Algebra: Free Model and Completeness

Tobias Kappé(✉), Paul Brunet, Alexandra Silva, and Fabio Zanasi

University College London, London, UK
tkappe@cs.ucl.ac.uk

- 2009: CKA is introduced.
- 2011: Models of CKA are introduced, and the relationship with separation logic is established.
- 2014: First completeness theorem (without the exchange law), CKA with tests is introduced.
- 2016: Second paper on CKAT, correcting some mistakes from the first one.
- 2017: Decidability of CKA.
- 2018: Completeness of CKA.

Concurrent Kleene Algebra

Concurrent Kleene Algebra with Observations: from Hypotheses to Completeness

Tobias Kappé (✉), Paul Brunet, Alexandra Silva,
Jana Wagemaker, and Fabio Zanasi

University College London, London, United Kingdom; tkappe@cs.ucl.ac.uk

Pomsets with Boxes: Protection, Separation, and Locality in Concurrent Kleene Algebra

Paul Brunet
University College London, UK
paul.brunet-zamansky.fr
paul@brunet-zamansky.fr

David Pym
University College London, UK
www.cantab.net/users/david.pym/
d.pym@ucl.ac.uk

Partially Observable Concurrent Kleene Algebra

Jana Wagemaker
Radboud University, Nijmegen
j.wagemaker@cs.ru.nl

Paul Brunet
University College London

Simon Docherty
University College London

Tobias Kappé
University College London

Jurriaan Rot
Radboud University, Nijmegen and University College London

Alexandra Silva
University College London

2016 2017 2018 2020

CKA is

eorem
e law),
intro-

Decidability of CKA.

Completeness of CKA.

CKA with observations, partially observable CKA, CKA with boxes.

Second paper on CKAT, correcting some mistakes from the first one.

Kleene algebra: the algebra of regular expressions

$$e, f \in E_A ::= 0 \mid 1 \mid a \mid e \cdot f \mid e + f \mid e^\star$$

Interpretation: regular languages

$$[\![\cdot]\!] : E_A \to \mathcal{P}(A^\star)$$

example:

$$[\![a \cdot ((a+b) \cdot a)^\star]\!] = \left\{\begin{array}{l}\text{words of odd length over the} \\ \text{alphabet } \{a, b\} \text{ such that every} \\ \text{other letter is an } a\end{array}\right\}$$

$$= \{a, aaa, aba, aaaaa, abaaa, aaaba, ababa, \ldots\}$$

Kleene algebra: the algebra of regular expressions

The axioms of KA

$$
\begin{array}{ccc}
e+e=e & e+f=f+e & e+(f+g)=(e+f)+g \\
e+0=0 & e\cdot 1=e=1\cdot e & e\cdot(f\cdot g)=(e\cdot f)\cdot g \\
e\cdot 0=0=0\cdot e & e\cdot(f+g)=e\cdot f+e\cdot g & (e+f)\cdot g=e\cdot g+f\cdot g
\end{array}
$$

$$e^\star = 1 + e\cdot e^\star \qquad e\cdot f\leq f \Rightarrow e^\star\cdot f\leq f$$

Theorem

$$KA \vdash e = f \Leftrightarrow [\![e]\!] = [\![f]\!].$$

Kozen, "A completeness theorem for Kleene algebras and the algebra of regular events", LiCS '90

KAT: the algebra of imperative programs

Syntax

$$e, f \in E_{A \cup B_T} ::= 0 \mid 1 \mid a \in A \mid t \in B_T \mid e \cdot f \mid e + f \mid e^\star$$

$$t, t_1, t_2 \in B_T ::= \top \mid \bot \mid \alpha \in T \mid t_1 \wedge t_2 \mid t_1 \vee t_2 \mid \neg t$$

Encodes a simple `While` language:

$$\texttt{if } b \texttt{ then } p \texttt{ else } q \mapsto b \cdot p + \neg b \cdot q \qquad \texttt{while } b \texttt{ do } p \mapsto (b \cdot p)^\star \cdot \neg b$$

KAT: the algebra of imperative programs

Syntax

abort execution

$$e, f \in E_{A \cup B_T} ::= 0 \mid 1 \mid a \in A \mid t \in B_T \mid e \cdot f \mid e + f \mid e^\star$$

$$t, t_1, t_2 \in B_T ::= \top \mid \bot \mid \alpha \in T \mid t_1 \wedge t_2 \mid t_1 \vee t_2 \mid \neg t$$

Encodes a simple `While` language:

$$\texttt{if } b \texttt{ then } p \texttt{ else } q \mapsto b \cdot p + \neg b \cdot q \qquad \texttt{while } b \texttt{ do } p \mapsto (b \cdot p)^\star \cdot \neg b$$

KAT: the algebra of imperative programs

skip

Syntax

abort execution

$$e, f \in E_{A\cup B_T} ::= 0 \mid 1 \mid a \in A \mid t \in B_T \mid e \cdot f \mid e + f \mid e^\star$$

$$t, t_1, t_2 \in B_T ::= \top \mid \bot \mid \alpha \in T \mid t_1 \wedge t_2 \mid t_1 \vee t_2 \mid \neg t$$

Encodes a simple `While` language:

$$\texttt{if } b \texttt{ then } p \texttt{ else } q \mapsto b \cdot p + \neg b \cdot q \qquad \texttt{while } b \texttt{ do } p \mapsto (b \cdot p)^\star \cdot \neg b$$

KAT: the algebra of imperative programs

skip

abort execution atomic action

Syntax

$$e, f \in E_{A \cup B_T} ::= 0 \mid 1 \mid a \in A \mid t \in B_T \mid e \cdot f \mid e + f \mid e^\star$$

$$t, t_1, t_2 \in B_T ::= \top \mid \bot \mid \alpha \in T \mid t_1 \wedge t_2 \mid t_1 \vee t_2 \mid \neg t$$

Encodes a simple `While` language:

$$\texttt{if } b \texttt{ then } p \texttt{ else } q \mapsto b \cdot p + \neg b \cdot q \qquad \texttt{while } b \texttt{ do } p \mapsto (b \cdot p)^\star \cdot \neg b$$

KAT: the algebra of imperative programs

Syntax

skip

test

abort execution

atomic action

$$e, f \in E_{A \cup B_T} ::= 0 \mid 1 \mid a \in A \mid t \in B_T \mid e \cdot f \mid e + f \mid e^\star$$

$$t, t_1, t_2 \in B_T ::= \top \mid \bot \mid \alpha \in T \mid t_1 \wedge t_2 \mid t_1 \vee t_2 \mid \neg t$$

atomic test

Encodes a simple `While` language:

$$\texttt{if } b \texttt{ then } p \texttt{ else } q \mapsto b \cdot p + \neg b \cdot q \qquad \texttt{while } b \texttt{ do } p \mapsto (b \cdot p)^\star \cdot \neg b$$

KAT: THE ALGEBRA OF IMPERATIVE PROGRAMS

Syntax

skip test

abort execution atomic action sequential composition

$$e, f \in E_{A \cup B_T} ::= 0 \mid 1 \mid a \in A \mid t \in B_T \mid e \cdot f \mid e + f \mid e^\star$$

$$t, t_1, t_2 \in B_T ::= \top \mid \bot \mid \alpha \in T \mid t_1 \wedge t_2 \mid t_1 \vee t_2 \mid \neg t$$

atomic test

Encodes a simple `While` language:

$\texttt{if } b \texttt{ then } p \texttt{ else } q \mapsto b \cdot p + \neg b \cdot q$ $\qquad$ $\texttt{while } b \texttt{ do } p \mapsto (b \cdot p)^\star \cdot \neg b$

KAT: THE ALGEBRA OF IMPERATIVE PROGRAMS

skip
test
non-deterministic choice

Syntax

abort execution
atomic action
sequential composition

$$e, f \in E_{A \cup B_T} ::= 0 \mid 1 \mid a \in A \mid t \in B_T \mid e \cdot f \mid e + f \mid e^\star$$

$$t, t_1, t_2 \in B_T ::= \top \mid \bot \mid \alpha \in T \mid t_1 \wedge t_2 \mid t_1 \vee t_2 \mid \neg t$$

atomic test

Encodes a simple `While` language:

$$\texttt{if } b \texttt{ then } p \texttt{ else } q \mapsto b \cdot p + \neg b \cdot q \qquad \texttt{while } b \texttt{ do } p \mapsto (b \cdot p)^\star \cdot \neg b$$

KAT: THE ALGEBRA OF IMPERATIVE PROGRAMS

skip test non-deterministic choice

Syntax

abort execution atomic action sequential composition non-deterministic loop

$$e, f \in E_{A\cup B_T} ::= 0 \mid 1 \mid a \in A \mid t \in B_T \mid e \cdot f \mid e + f \mid e^\star$$

$$t, t_1, t_2 \in B_T ::= \top \mid \bot \mid \alpha \in T \mid t_1 \wedge t_2 \mid t_1 \vee t_2 \mid \neg t$$

atomic test

Encodes a simple `While` language:

$$\texttt{if } b \texttt{ then } p \texttt{ else } q \mapsto b \cdot p + \neg b \cdot q \qquad \texttt{while } b \texttt{ do } p \mapsto (b \cdot p)^\star \cdot \neg b$$

KAT: the algebra of imperative programs

Syntax

$$e, f \in E_{A \cup B_T} ::= 0 \mid 1 \mid a \in A \mid t \in B_T \mid e \cdot f \mid e + f \mid e^\star$$

$$t, t_1, t_2 \in B_T ::= \top \mid \bot \mid \alpha \in T \mid t_1 \wedge t_2 \mid t_1 \vee t_2 \mid \neg t$$

Encodes a simple `While` language:

$$\texttt{if } b \texttt{ then } p \texttt{ else } q \mapsto b \cdot p + \neg b \cdot q \qquad \texttt{while } b \texttt{ do } p \mapsto (b \cdot p)^\star \cdot \neg b$$

Interpretation: languages of guarded strings

<u>guarded strings</u>: alternating sequences of states $\in 2^T$ & actions $\in A$.

$$\begin{array}{|c|c|}\hline \alpha_1 & 1 \\ \alpha_2 & 0 \\ \alpha_3 & 1 \\ \alpha_4 & 1 \\ \hline \end{array} \xrightarrow{\;a\;} \begin{array}{|c|c|}\hline \alpha_1 & 1 \\ \alpha_2 & 0 \\ \alpha_3 & 0 \\ \alpha_4 & 1 \\ \hline \end{array} \xrightarrow{\;b\;} \begin{array}{|c|c|}\hline \alpha_1 & 0 \\ \alpha_2 & 0 \\ \alpha_3 & 1 \\ \alpha_4 & 0 \\ \hline \end{array} \xrightarrow{\;c\;} \begin{array}{|c|c|}\hline \alpha_1 & 1 \\ \alpha_2 & 1 \\ \alpha_3 & 1 \\ \alpha_4 & 0 \\ \hline \end{array}$$

KAT: the algebra of imperative programs

The axioms of KAT:

- The axioms of KA.
- For tests, the axioms of boolean algebra.
- The following "glue" axioms:

$$t_1 \vee t_2 = t_1 + t_2 \qquad t_1 \wedge t_2 = t_1 \cdot t_2 \qquad \top = 1 \qquad \bot = 0$$

Theorem

$$KAT \vdash e = f \Leftrightarrow [\![e]\!] = [\![f]\!].$$

Kozen & Smith, "Kleene algebra with tests: Completeness and decidability", CSL '96

KAT: the algebra of imperative programs

The axioms of KAT:

- ☞ The axioms of KA.
- ☞ For tests, the axioms of boolean algebra.
- ☞ The following "glue" axioms:

$$t_1 \vee t_2 = t_1 + t_2 \qquad t_1 \wedge t_2 = t_1 \cdot t_2 \qquad \top = 1 \qquad \bot = 0$$

Theorem

$$KAT \vdash e = f \Leftrightarrow [\![e]\!] = [\![f]\!].$$

Kozen & Smith, "Kleene algebra with tests: Completeness and decidability", CSL '96

Subsumes Hoare logic:

$$\begin{aligned} \{b\}\, p\, \{c\} &\Leftrightarrow b \cdot p \leq p \cdot c \\ &\Leftrightarrow b \cdot p = b \cdot p \cdot c \\ &\Leftrightarrow b \cdot p \cdot \neg c = 0 \end{aligned}$$

KAT: the algebra of imperative programs

The axioms of KAT:

- The axioms of KA.
- For tests, the axioms of boolean algebra.
- The following "glue" axioms:

$$t_1 \vee t_2 = t_1 + t_2 \qquad t_1 \wedge t_2 = t_1 \cdot t_2 \qquad \top = 1 \qquad \bot = 0$$

Theorem

$$KAT \vdash e = f \Leftrightarrow [\![e]\!] = [\![f]\!].$$

Kozen & Smith, "Kleene algebra with tests: Completeness and decidability", CSL '96

Subsumes Hoare logic:

$$\begin{aligned} \{b\}\, p \,\{c\} &\Leftrightarrow b \cdot p \leq p \cdot c \\ &\Leftrightarrow b \cdot p = b \cdot p \cdot c \\ &\Leftrightarrow b \cdot p \cdot \neg c = 0 \end{aligned}$$

Can we do the same for concurrent programs?

Outline

Recent developments in CKA

I. Concurrent Kleene Algebra

II. CKA with observations

III. Partially observable CKA

IV. CKA with boxes

V. Ongoing and future work

Outline

Recent developments in CKA

I. Concurrent Kleene Algebra

II. CKA with observations

III. Partially observable CKA

IV. CKA with boxes

V. Ongoing and future work

bi-Kleene Algebra

$$e, f ::= 1 \mid 0 \mid x \mid e \cdot f \mid e \parallel f \mid e + f \mid e^\star \mid e^!$$

Definition

A bi-Kleene algebra is a structure $\langle A, 0, 1, \cdot, \parallel, +, \star, ! \rangle$ such that:

- ☞ $\langle A, 0, 1, \cdot, +, \star \rangle$ is a KA
- ☞ $\langle A, 0, 1, \parallel, +, ! \rangle$ is a commutative KA.

bi-Kleene Algebra

$$e, f ::= 1 \mid 0 \mid x \mid e \cdot f \mid e \parallel f \mid e + f \mid e^\star \mid e^!$$

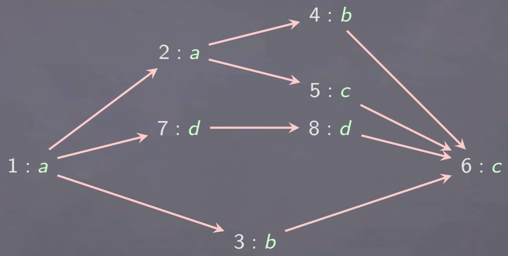
Definition

A bi-Kleene algebra is a structure $\langle A, 0, 1, \cdot, \parallel, +, \star, ! \rangle$ such that:

- ☞ $\langle A, 0, 1, \cdot, +, \star \rangle$ is a KA
- ☞ $\langle A, 0, 1, \parallel, +, ! \rangle$ is a commutative KA.

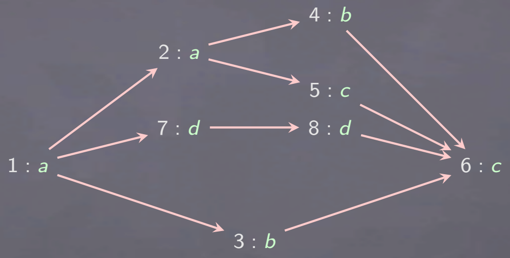
What is the free bi-KA?

Pomsets: concurrent traces



Pomsets: concurrent traces

A is some alphabet of actions.



finite set of events

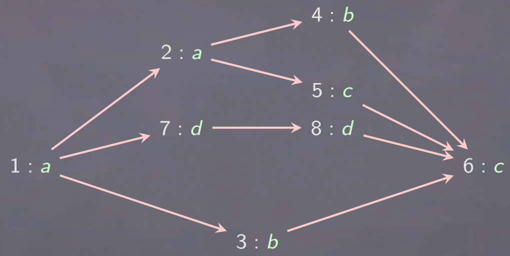
labeling function
$: E_P \to A$

$$P = \langle E_P, \leqslant_P, \lambda_P \rangle$$

partial order
$\subseteq E_P \times E_P$

Pomsets: concurrent traces

A is some alphabet of actions.



finite set of events

labeling function $: E_P \to A$

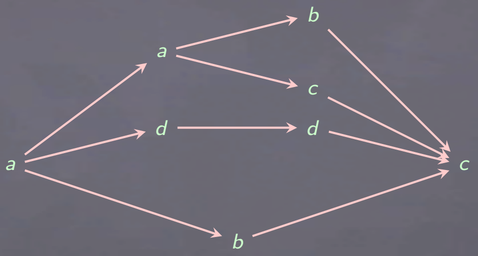
$$P = \langle E_P, \leqslant_P, \lambda_P \rangle$$

partial order $\subseteq E_P \times E_P$

Up-to isomorphism $\equiv$.

Pomsets: concurrent traces

A is some alphabet of actions.



finite set of events

labeling function
$: E_P \to A$

$$P = \langle E_P, \leqslant_P, \lambda_P \rangle$$

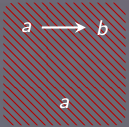
partial order
$\subseteq E_P \times E_P$

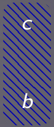
Up-to isomorphism $\equiv$.

Combining pomsets

$a =$ 

$1 =$ 

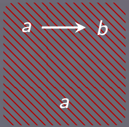
$P_1 =$ 

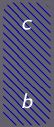
$P_2 =$ 

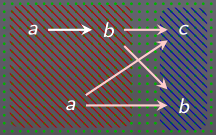
Combining pomsets

$a =$ 

$1 =$ 

$P_1 =$ 

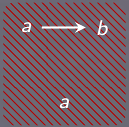
$P_2 =$ 

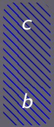
$P_1 ; P_2 =$ 

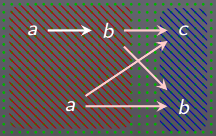
Combining pomsets

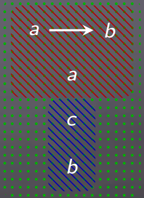
$a =$ 

$1 =$ 

$P_1 =$ 

$P_2 =$ 

$P_1 ; P_2 =$ 

$P_1 \parallel P_2 =$ 

Completeness of biKA

$$
\begin{aligned}
[\![1]\!] &:= \{1\} & [\![0]\!] &:= \emptyset \\
[\![x]\!] &:= \{x\} & [\![e+f]\!] &:= [\![e]\!] \cup [\![f]\!] \\
[\![e \cdot f]\!] &:= \{P;Q \mid P \in [\![e]\!],\ Q \in [\![f]\!]\} & [\![e \parallel f]\!] &:= \{P \parallel Q \mid P \in [\![e]\!],\ Q \in [\![f]\!]\} \\
[\![e^\star]\!] &:= \{P_1;\cdots;P_n \mid n \in \mathbb{N},\ P_i \in [\![e]\!]\} & [\![e^!]\!] &:= \{P_1 \parallel \cdots \parallel P_n \mid n \in \mathbb{N},\ P_i \in [\![e]\!]\}
\end{aligned}
$$

Theorem

$$biKA \vdash e = f \Leftrightarrow [\![e]\!] \equiv [\![f]\!].$$

Laurence & Struth, "Completeness Theorems for Bi-Kleene Algebras and Series-Parallel Rational Pomset Languages", RAMiCS '14

Concurrent Kleene Algebra

Interchange law

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

No parallel iteration

CKA

A concurrent Kleene algebra is a weak bi-Kleene algebra $\langle A, 0, 1, \cdot, \parallel, +, \star \rangle$ satisfying the interchange law.

Interleavings and subsumption

Interchange law

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

$$\begin{tikzcd} a \arrow[r] \arrow[dr] & c \\ b \arrow[r] \arrow[ur] & d \end{tikzcd} \qquad \sqsubseteq \qquad \begin{tikzcd} a \arrow[r] & c \\ b \arrow[r] & d \end{tikzcd}$$

$P \sqsubseteq Q$ when there is a homomorphism from Q to P, i.e. a bijective map $\varphi : E_Q \to E_P$ such that $\lambda_P \circ \varphi = \lambda_Q$ and $\varphi(\leq_Q) \subseteq \leq_P$.

$$L^{\sqsubseteq} := \{P \mid \exists Q \in L : P \sqsubseteq Q\}.$$

Completeness and decidability of CKA

Theorem

The problem of testing whether two given expressions e, f denote the same closed language is ExpSpace-complete.

B., Pous, & Struth, "On Decidability of Concurrent Kleene Algebra", CONCUR '17

Theorem

$$CKA \vdash e = f \Leftrightarrow [\![e]\!]^{\sqsubseteq} = [\![f]\!]^{\sqsubseteq}.$$

Kappé, B., Silva, & Zanasi, "Concurrent Kleene Algebra: Free Model and Completeness", ESOP '18

Outline

Recent developments in CKA

I. Concurrent Kleene Algebra

II. CKA with observations

III. Partially observable CKA

IV. CKA with boxes

V. Ongoing and future work

CKAT

Slogan

A KAT is a KA with a boolean sub-algebra.
A CKAT is a CKA with a boolean sub-algebra.

CKAT

Slogan

A KAT is a KA with a boolean sub-algebra.
A CKAT is a CKA with a boolean sub-algebra.

$$
\begin{aligned}
t \cdot p \cdot \neg t &\leq p \parallel (t \cdot \neg t) && \text{(CKA axioms)} \\
&= p \parallel (t \wedge \neg t) && (\wedge = \cdot) \\
&= p \parallel \bot && \text{(Boolean axioms)} \\
&= p \parallel 0 && (\bot = 0) \\
&= 0 && \text{(CKA axioms)}
\end{aligned}
$$

CKAT: DOOMED!

Slogan

A KAT is a KA with a boolean sub-algebra.
A CKAT is a CKA with a boolean sub-algebra.

$$
\begin{aligned}
t \cdot p \cdot \neg t &\leq p \parallel (t \cdot \neg t) && \text{(CKA axioms)}\\
&= p \parallel (t \wedge \neg t) && (\wedge = \cdot)\\
&= p \parallel \bot && \text{(Boolean axioms)}\\
&= p \parallel 0 && (\bot = 0)\\
&= 0 && \text{(CKA axioms)}
\end{aligned}
$$

$\leftrightarrow$ For every program and every assertion, the triple $\{t\}\,p\,\{t\}$ holds.
$\leftrightarrow$ Every test is invariant under every program.

Who's to blame?

$$
\begin{aligned}
t \cdot p \cdot \neg t &\leq p \parallel (t \cdot \neg t) && \text{(CKA axioms)}\\
&= p \parallel (t \wedge \neg t) && (\wedge = \cdot)\\
&= p \parallel \bot && \text{(Boolean axioms)}\\
&= p \parallel 0 = 0 && (\bot = 0 + \text{CKA axioms})
\end{aligned}
$$

Who's to blame?

$$\begin{aligned} t \cdot p \cdot \neg t &\leq p \parallel (t \cdot \neg t) && \text{(CKA axioms)} \\ &= p \parallel (t \wedge \neg t) && \boxed{(\wedge = \cdot)} \\ &= p \parallel \bot && \text{(Boolean axioms)} \\ &= p \parallel 0 = 0 && (\bot = 0 + \text{CKA axioms}) \end{aligned}$$

$$a \wedge b \;=\; a \cdot b$$

"If we observe a, and then observe b without any action in between, then both observations are made on the <u>same</u> state. Therefore that state simultaneously satisfies a and b."

Who's to blame?

$$
\begin{aligned}
t \cdot p \cdot \neg t &\leq p \parallel (t \cdot \neg t) && \text{(CKA axioms)} \\
&= p \parallel (t \wedge \neg t) && \boxed{\cancel{(\wedge = \cdot)}} \\
&= p \parallel \bot && \text{(Boolean axioms)} \\
&= p \parallel 0 = 0 && (\bot = 0 + \text{CKA axioms})
\end{aligned}
$$

$$\cancel{a \wedge b \;=\; a \cdot b}$$

"If we observe a, and then observe b without any action in between, then both observations are made on the same state. Therefore that state simultaneously satisfies a and b."

$$\boxed{a \wedge b \;\leq\; a \cdot b}$$

CKAO - Syntax

$$e, f \in E_{A \cup B_T} ::= 0 \mid 1 \mid a \in A \mid t \in B_T \mid e \cdot f \mid e \parallel f \mid e + f \mid e^\star$$

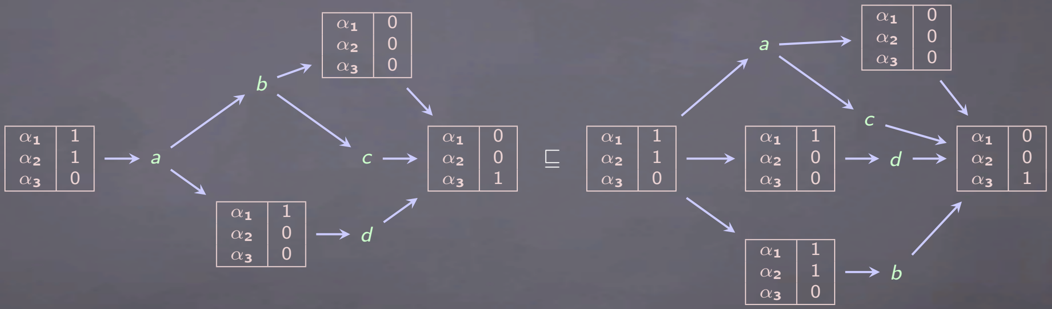
$$t, t_1, t_2 \in B_T ::= \top \mid \bot \mid \alpha \in T \mid t_1 \wedge t_2 \mid t_1 \vee t_2 \mid \neg t$$

The axioms of CKAO

- The axioms of CKA.
- For tests, the axioms of boolean algebra.
- The following "glue" axioms:

$$t_1 \vee t_2 = t_1 + t_2 \qquad t_1 \wedge t_2 \leq t_1 \cdot t_2 \qquad \bot = 0$$

CKAO - Model



Theorem

$$CKAO \vdash e = f \Leftrightarrow [\![e]\!]{\downarrow} = [\![f]\!]{\downarrow}.$$

Interlude: (C)KA with hypotheses

H: set of hypotheses $e \leq f$ over some fixed alphabet A.

- extra structure on the alphabet (e.g. $\alpha \wedge \beta = \beta \wedge \alpha$);
- extra structure on traces (e.g. $\alpha \leq \alpha \cdot \alpha$)
- other domain-specific assumptions.

Theorem

$$CKA + H \vdash e = f \Rightarrow [\![e]\!]{\downarrow}^H = [\![f]\!]{\downarrow}^H$$

- Doumane, Kuperberg, Pous, & Pradic, "Kleene Algebra with Hypotheses", FoSSaCS '19
- Kappé, B., Silva, Wagemaker, & Zanasi, "Concurrent Kleene Algebra with Observations: from Hypotheses to Completeness", FoSSaCS '20

Outline

Recent developments in CKA

I. Concurrent Kleene Algebra

II. CKA with observations

III. Partially observable CKA

IV. CKA with boxes

V. Ongoing and future work

Litmus test: sequential consistency

```
{ r0 == 0 && r1 == 0 }

   x := 1   ||  y := 1
   r0 := y  ||  r1 := x

{ !( r0 == 1 || r1 == 1 ) }
```

Ingredients:

- ☞ Assignments $x \leftarrow 1$
- ☞ Observations $r_0 = 0$

What kind of observations do we need?
First attempt: boolean algebra

☞ Atomic observations: $\textsc{Var} == \textsc{Val}$ e.g. $r_0 == 1$

What kind of observations do we need?

First attempt: boolean algebra

- <u>Atomic observations</u>: $\textsc{Var} == \textsc{Val}$ e.g. $r_0 == 1$
- <u>Boolean formula</u>: set of memory states $\textsc{Var} \to \textsc{Val}$ e.g.

r_0	1
r_1	0

What kind of observations do we need?
First attempt: Boolean algebra

- <u>Atomic observations</u>: $\textsc{Var} == \textsc{Val}$ e.g. $r_0 == 1$
- <u>Boolean formula</u>: set of memory states $\textsc{Var} \to \textsc{Val}$ e.g. $\begin{array}{|c|c|} r_0 & 1 \\ r_1 & 0 \end{array}$
- <u>Assignments</u>: $\sum_{s \in State} s \cdot (v \leftarrow n) \cdot s[v \mapsto n]$, i.e.

$$[\![x \leftarrow 1]\!] := \left\{ \begin{array}{|c|c|} x & 0 \\ y & 0 \end{array} \to [x \leftarrow 1] \to \begin{array}{|c|c|} x & 1 \\ y & 0 \end{array}, \begin{array}{|c|c|} x & 0 \\ y & 1 \end{array} \to [x \leftarrow 1] \to \begin{array}{|c|c|} x & 1 \\ y & 1 \end{array} \right\}$$

What kind of observations do we need?

First attempt: Boolean algebra

- Atomic observations: $\textsc{Var} == \textsc{Val}$ e.g. $r_0 == 1$
- Boolean formula: set of memory states $\textsc{Var} \to \textsc{Val}$ e.g. $\begin{array}{|c|c|}\hline r_0 & 1 \\ r_1 & 0 \\ \hline\end{array}$
- Assignments: $\sum_{s \in State} s \cdot (v \leftarrow n) \cdot s[v \mapsto n]$, i.e.

$$[\![x \leftarrow 1]\!] := \left\{ \begin{array}{|c|c|}\hline x & 0 \\ y & 0 \\ \hline\end{array} \to [x \leftarrow 1] \to \begin{array}{|c|c|}\hline x & 1 \\ y & 0 \\ \hline\end{array}\ ,\ \begin{array}{|c|c|}\hline x & 0 \\ y & 1 \\ \hline\end{array} \to [x \leftarrow 1] \to \begin{array}{|c|c|}\hline x & 1 \\ y & 1 \\ \hline\end{array} \right\}$$

Problem: parallel composition?

$$\begin{array}{|c|c|}\hline x & 0 \\ y & 0 \\ \hline\end{array} \to [x \leftarrow 1] \to \begin{array}{|c|c|}\hline x & 1 \\ y & 0 \\ \hline\end{array}$$

$$\begin{array}{|c|c|}\hline x & 0 \\ y & 0 \\ \hline\end{array} \to [y \leftarrow 1] \to \begin{array}{|c|c|}\hline x & 0 \\ y & 1 \\ \hline\end{array}$$

Algebra of partial observations

Idea: Instead of memory state $\text{Var} \to \text{Val}$, consider partial functions $\text{Var} \rightharpoonup \text{Val}$.

PCDL of observations

$$t, t_1, t_2 \in O_T ::= \top \mid \bot \mid \alpha \in T \mid t_1 \wedge t_2 \mid t_1 \vee t_2 \mid \overline{t}$$

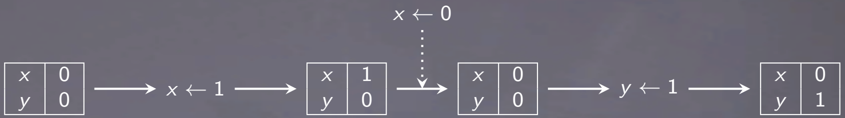
Same axioms as BA regarding $\vee, \wedge, \top, \bot$, plus:

- ☞ $p \leq \overline{q} \Leftrightarrow p \wedge q = \bot$
- ☞ $\overline{v = n} = \bigvee_{m \neq n} v = m$

Causality vs compositionality

$$\begin{array}{|c|c|}\hline x & 0 \\ y & 0 \\ \hline\end{array} \xrightarrow{\quad x \leftarrow 1 \quad} \begin{array}{|c|c|}\hline x & 1 \\ y & 0 \\ \hline\end{array} \xrightarrow{x \leftarrow 0} \begin{array}{|c|c|}\hline x & 0 \\ y & 0 \\ \hline\end{array} \xrightarrow{\quad y \leftarrow 1 \quad} \begin{array}{|c|c|}\hline x & 0 \\ y & 1 \\ \hline\end{array}$$

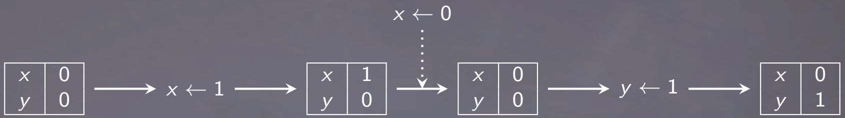
Causality vs compositionality



Solution: we need to explicitly close the system.

$$[\![e]\!] \quad \rightarrow \quad [\![e]\!] \cap \mathit{CausalPomsets}.$$

Causality vs compositionality



Solution: we need to explicitly close the system.

$$[\![e]\!] \quad \to \quad [\![e]\!] \cap \mathit{CausalPomsets}.$$

Litmus test:

$$t := (r_0 = 0 \wedge r_1 = 0) \cdot ((x \leftarrow 1 \cdot r_0 \leftarrow y) \parallel (y \leftarrow 1 \cdot r_1 \leftarrow x)) \cdot \overline{(r_0 = 1 \vee r_1 \vee 1)}$$

$$[\![t]\!] \cap \mathit{CausalPomsets} = \emptyset$$

Outline

Recent developments in CKA

I. Concurrent Kleene Algebra

II. CKA with observations

III. Partially observable CKA

IV. CKA with boxes

V. Ongoing and future work

Mutual exclusion

```
      print(counter);
                 ||
x:=counter;     ||   y:=counter;
x:=x+1;         ||   y:=y+1;
counter:=x;     ||   counter:=y;
                 ||
      print(counter);
```

Mutual exclusion

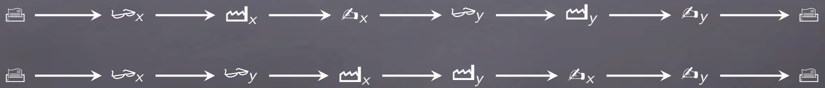
```
print(counter);

x:=counter;   ||   y:=counter;
x:=x+1;       ||   y:=y+1;
counter:=x;   ||   counter:=y;

print(counter);
```

Mutual exclusion

```
        print(counter);
                 ||
x:=counter;      ||    y:=counter;
x:=x+1;          ||    y:=y+1;
counter:=x;      ||    counter:=y;
                 ||
        print(counter);
```

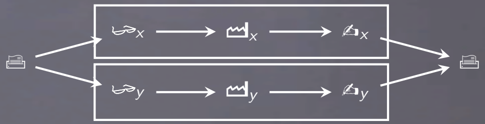
Mutual exclusion

```
        print(counter);
atomic{              || atomic{
   x:=counter;       ||    y:=counter;
   x:=x+1;           ||    y:=y+1;
   counter:=x;       ||    counter:=y;
}                    || }
        print(counter);
```



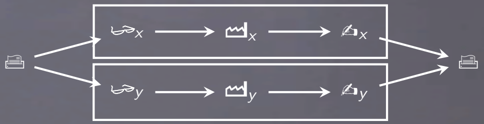
Mutual exclusion

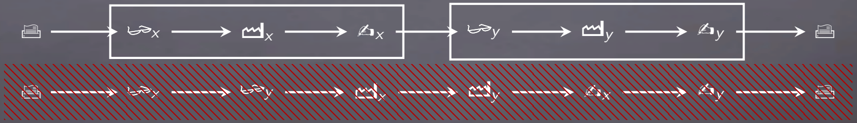
```
        print(counter);
atomic{              || atomic{
   x:=counter;       ||    y:=counter;
   x:=x+1;           ||    y:=y+1;
   counter:=x;       ||    counter:=y;
}                    || }
        print(counter);
```



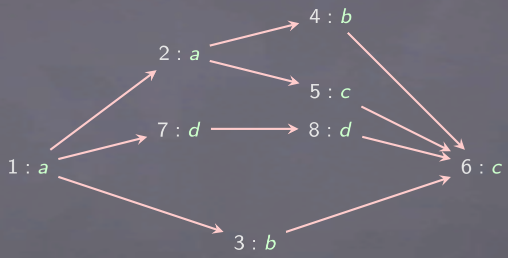
Mutual exclusion

```
       print(counter);
atomic{          || atomic{
  x:=counter;    ||   y:=counter;
  x:=x+1;        ||   y:=y+1;
  counter:=x;    ||   counter:=y;
}                || }
       print(counter);
```





Pomsets with boxes



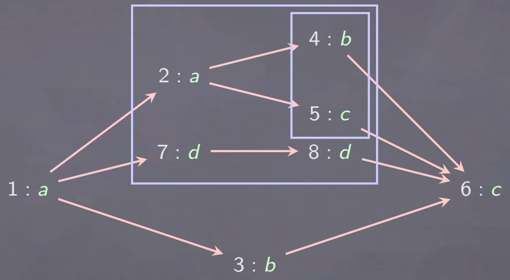
finite set of events

labeling function
$: E_P \to A$

$$P = \langle E_P, \leqslant_P, \lambda_P \quad \rangle$$

partial order
$\subseteq E_P \times E_P$

Pomsets with boxes



finite set of events

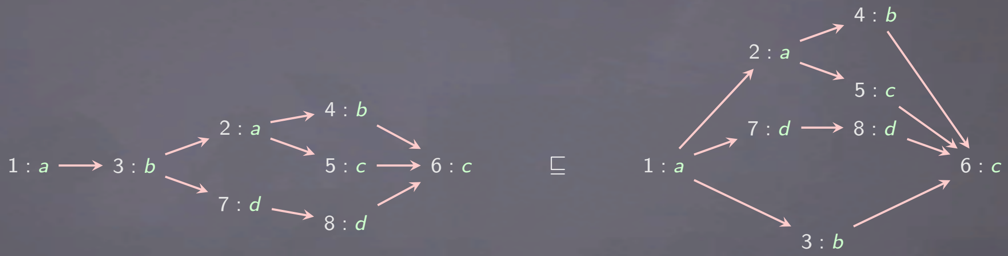
labeling function $: E_P \to A$

$$P = \langle E_P, \leqslant_P, \lambda_P, B_P \rangle$$

partial order $\subseteq E_P \times E_P$

set of boxes $\subseteq \mathcal{P}^{\neq\emptyset}(E_P)$

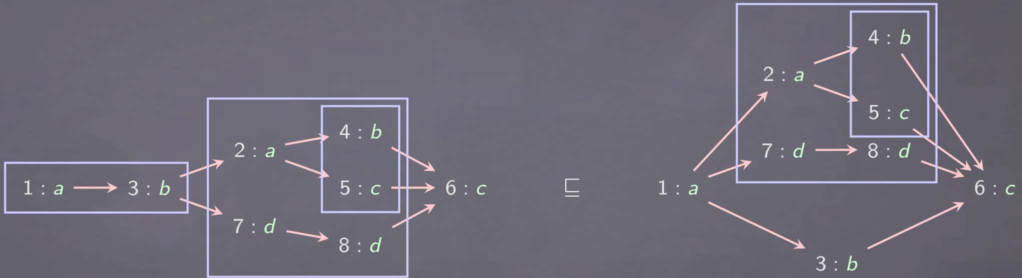
Subsumption with boxes



$P \sqsubseteq Q$ when there is a homomorphism from Q to P, i.e. a bijective map $\varphi : E_Q \to E_P$ such that

1) $\lambda_P \circ \varphi = \lambda_Q$
2) $\varphi(\leq_Q) \subseteq \leq_P$

Subsumption with boxes



$P \sqsubseteq Q$ when there is a homomorphism from Q to P, i.e. a bijective map $\varphi : E_Q \to E_P$ such that

1) $\lambda_P \circ \varphi = \lambda_Q$
2) $\varphi(\leq_Q) \subseteq \leq_P$
3) $\varphi(\mathcal{B}_P) \subseteq \mathcal{B}_Q$

Axiomatisation

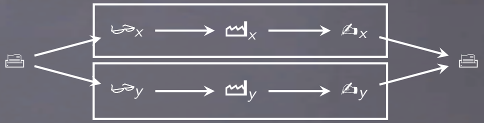
$$[[e]] = [e]$$
$$[1] = 1$$
$$[0] = 0$$
$$[e + f] = [e] + [f]$$
$$[e] \leq e$$

Claim

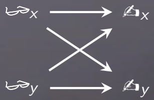
$$\llbracket e \rrbracket = \llbracket f \rrbracket \Leftrightarrow \quad CKA + B \vdash e = f\,.$$

Mutual exclusion (II)

```
        print(counter);
atomic{             || atomic{
  x:=counter;       ||   y:=counter;
  x:=x+1;           ||   y:=y+1;
  counter:=x;       ||   counter:=y;
}                   || }
        print(counter);
```



Breaking mutual exclusion $\leftrightarrow$ admitting an execution with the following "pattern":



Pomset logic

$$\varphi, \psi ::= \bot \mid a \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \varphi \blacktriangleright \psi \mid \varphi \star \psi \mid [\varphi] \mid (\!|\varphi|\!)$$

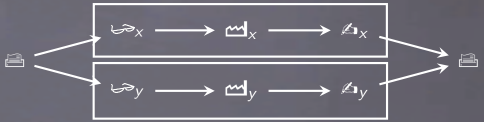
- ☞ $P \models \varphi \blacktriangleright \psi$ iff $\exists P_1, P_2$ such that $P \sqsupseteq P_1 \cdot P_2$ and $P_1 \models \varphi$ and $P_2 \models \psi$
- ☞ $P \models \varphi \star \psi$ iff $\exists P_1, P_2$ such that $P \sqsupseteq P_1 \parallel P_2$ and $P_1 \models \varphi$ and $P_2 \models \psi$
- ☞ $P \models [\varphi]$ iff $\exists Q$ such that $P \sqsupseteq [Q]$ and $Q \models \varphi$
- ☞ $P \models (\!|\varphi|\!)$ iff $\exists P', Q$ such that $P \sqsupseteq P'$ and $P' \ni Q$ and $Q \models \varphi$.

Theorem

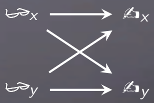
$$P \sqsupseteq Q \Leftrightarrow \forall \varphi, (P \models \varphi \Rightarrow Q \models \varphi)\,.$$

Mutual exclusion (III)

```
        print(counter);
atomic{          || atomic{
  x:=counter;    ||   y:=counter;
  x:=x+1;        ||   y:=y+1;
  counter:=x;    ||   counter:=y;
}                || }
        print(counter);
```



Breaking mutual exclusion $\leftrightarrow$ admitting an execution with the following "pattern":



$\leftrightarrow P \models (\!|(\text{👓}_x \star \text{👓}_y) \blacktriangleright (\text{✍}_x \star \text{✍}_y)|\!)$

Outline

Recent developments in CKA

I. Concurrent Kleene Algebra

II. CKA with observations

III. Partially observable CKA

IV. CKA with boxes

V. Ongoing and future work

Algebras with hypotheses

- ☞ Doumane, Kuperberg, Pous, & Pradic, "Kleene Algebra with Hypotheses", FoSSaCS '19.

Algebras with hypotheses

- ☞ Doumane, Kuperberg, Pous, & Pradic, "Kleene Algebra with Hypotheses", FoSSaCS '19.
- ☞ Kappé, B., Silva, Wagemaker, & Zanasi, "Concurrent Kleene Algebra with Observations: from Hypotheses to Completeness", FoSSaCS '20.

Algebras with hypotheses

- ☞ Doumane, Kuperberg, Pous, & Pradic, "Kleene Algebra with Hypotheses", FoSSaCS '19.
- ☞ Kappé, B., Silva, Wagemaker, & Zanasi, "Concurrent Kleene Algebra with Observations: from Hypotheses to Completeness", FoSSaCS '20.
- ☞ CKA with boxes and hypotheses?

Algebras with hypotheses

- Doumane, Kuperberg, Pous, & Pradic, "Kleene Algebra with Hypotheses", FoSSaCS '19.
- Kappé, B., Silva, Wagemaker, & Zanasi, "Concurrent Kleene Algebra with Observations: from Hypotheses to Completeness", FoSSaCS '20.
- CKA with boxes and hypotheses?

All proofs had to be re-done from scratch.

Algebras with hypotheses

- Doumane, Kuperberg, Pous, & Pradic, "Kleene Algebra with Hypotheses", FoSSaCS '19.
- Kappé, B., Silva, Wagemaker, & Zanasi, "Concurrent Kleene Algebra with Observations: from Hypotheses to Completeness", FoSSaCS '20.
- CKA with boxes and hypotheses?

All proofs had to be re-done from scratch.

Can we do better?

Logics of behaviour

☞ Traditional approaches to program logic rely on states
e.g. Hennessy-Milner Logic, (Propositional) Dynamic Logic...

Logics of behaviour

- ☞ Traditional approaches to program logic rely on states
  e.g. Hennessy-Milner Logic, (Propositional) Dynamic Logic...
- ☞ Pomset logic relies on an abstract notion of "behaviour" instead.

Logics of behaviour

- ☛ Traditional approaches to program logic rely on states
  e.g. Hennessy-Milner Logic, (Propositional) Dynamic Logic...
- ☛ Pomset logic relies on an abstract notion of "behaviour" instead.

What kinds of properties of behaviours are interesting and/or tractable?

Extensions of the model

☞ Merging boxes: $[e \cdot [f] \cdot g] = [e \cdot f \cdot g]$.

Extensions of the model

- Merging boxes: $[e \cdot [f] \cdot g] = [e \cdot f \cdot g]$.

- Beyond partial memory states: Arbitrary coherence relation between atomic observations.

$$v = 1 \asymp v = 0$$

Extensions of the model

- Merging boxes: $[e \cdot [f] \cdot g] = [e \cdot f \cdot g]$.

- Beyond partial memory states: Arbitrary coherence relation between atomic observations.

$$v = 1 \asymp v = 0$$

- Add data: Nominal algebras.

That's all folks!

Thank you!

See more at:
http://paul.brunet-zamansky.fr

Outline

Recent developments in CKA

I. Concurrent Kleene Algebra

II. CKA with observations

III. Partially observable CKA

IV. CKA with boxes

V. Ongoing and future work