

AVANCÉES RÉCENTES  
SUR LES  
ALGÈBRES DE KLEENE CONCURRENTES

SÉMINAIRE MAREL - LIRMM

Avril 2021

Paul Brunet  
University College London

# QUI SUIS-JE?

Diplômes	🎓	Licence d'informatique (ENS de Cachan antenne de Bretagne) .....	2009
	🎓	Master MPRI (Université Paris VII) .....	2013
	🎓	Doctorat d'informatique (Université de Lyon) .....	2016

« Algèbres de Relations : des algorithmes aux preuves formelles »  
thèse effectuée avec Damien Pous au LIP (ENS de Lyon)

# QUI SUIS-JE?

## Diplômes

- 🎓 Licence d'informatique (ENS de Cachan antenne de Bretagne) ..... 2009
- 🎓 Master MPRI (Université Paris VII) ..... 2013
- 🎓 Doctorat d'informatique (Université de Lyon) ..... 2016  
« Algèbres de Relations : des algorithmes aux preuves formelles »  
thèse effectuée avec Damien Pous au LIP (ENS de Lyon)

## Enseignements

- 🏛️ University College London ..... 2017-2019
  - 📖 Calculabilité and complexité (niveau L3/M1)
- 🏛️ ÉNS de Lyon ..... 2016
  - 📖 Sémantique and Vérification (niveau M1)
- 🏛️ UCB Lyon 1 ..... 2013-2016
  - 📖 Calculabilité and complexité (niveau M1)
  - 📖 Théorie des langages formels (niveau L3)
  - 📖 Logique classique (niveau L3)
  - 📖 Algorithmique et Programmation Impérative (niveau L1)
  - 📖 Algorithmique numérique (niveau L3)

# ACTIVITÉS DE RECHERCHE, VUES DE LOIN...

## Emploi

- 👉 Postdoc à l'ENS de Lyon avec Damien Pous (2016)
- 👉 Postdoc à UCL (GB) avec Alexandra Silva (2017-2018)
- 👉 Postdoc à UCL (GB) avec David Pym (2018-...)

# ACTIVITÉS DE RECHERCHE, VUES DE LOIN...

## Emploi

- 👉 Postdoc à l'ENS de Lyon avec Damien Pous (2016)
- 👉 Postdoc à UCL (GB) avec Alexandra Silva (2017-2018)
- 👉 Postdoc à UCL (GB) avec David Pym (2018-...)

## Co-auteurs

- 👤 D. Pous (ENS de Lyon)
- 👥 A. Silva, D. Pym, F. Zanasi, T. Kappé, S. Docherty (UCL)
- 👤 B. Luttik (TU Eindhoven)
- 👥 J. Rot, J. Wagemaker (Radboud University, Nijmegen)
- 👤 G. Struth (University of Sheffield)
- 👤 I. Stucke (Kiel University)

## Publications

- 📖 Journaux
  - ▶ JLAMP (2016, 2019)
  - ▶ LMCS (2017)
- 💬 Conférences
  - ▶ LiCS (2015)
  - ▶ ICALP (2019)
  - ▶ CONCUR (2017<sub>x2</sub>, 2019, 2020)
  - ▶ FSCD (2020)
  - ▶ FOSSaCS (2020)
  - ▶ ESOP (2018)
  - ▶ MFCS (2016, 2017)
  - ▶ CSL (2020)
  - ▶ RAMiCS (2014)

# ACTIVITÉS DE RECHERCHE, VUES DE LOIN...

## Emploi

- 👉 Postdoc à l'ENS de Lyon avec Damien Pous (2016)
- 👉 Postdoc à UCL (GB) avec Alexandra Silva (2017-2018)
- 👉 Postdoc à UCL (GB) avec David Pym (2018-...)

## Co-auteurs

- 👤 D. Pous (ENS de Lyon)
- 👥 A. Silva, D. Pym, F. Zanasi, T. Kappé, S. Docherty (UCL)
- 👤 B. Luttik (TU Eindhoven)
- 👥 J. Rot, J. Wagemaker (Radboud University, Nijmegen)
- 👤 G. Struth (University of Sheffield)
- 👤 I. Stucke (Kiel University)

## Publications


- 📖 Journaux
  - ▶ JLAMP (2016, 2019)
  - ▶ LMCS (2017)
- 💬 Conférences
  - ▶ LiCS (2015)
  - ▶ ICALP (2019)
  - ▶ CONCUR (2017<sub>x2</sub>, 2019, 2020)
  - ▶ FSCD (2020)
  - ▶ FOSSaCS (2020)
  - ▶ ESOP (2018)
  - ▶ MFCS (2016, 2017)
  - ▶ CSL (2020)
  - ▶ RAMiCS (2014)

## Et aussi

- 👉 Membre de comités de programme : WOLLIC 2021, CONCUR 2021.
- 👉 Jury de thèse de Joshua Moerman en juillet 2019 (Radboud University Nijmegen, NL)

# AVANCÉES RÉCENTES SUR CKA

## Plan

 I. Introduction

II. Algèbre de Kleene concurrente

III. Observations booléennes

IV. Observations partielles

V. Boîtes

VI. Travaux en cours et à venir

# ALGÈBRE DE KLEENE CONCURRENTE

## Concurrent Kleene Algebra

C.A.R. Tony Hoare<sup>1</sup>, Bernhard Möller<sup>2</sup>, Georg Struth<sup>3</sup>, and Ian Wehrman<sup>4</sup>

<sup>1</sup> Microsoft Research, Cambridge, UK

<sup>2</sup> Universität Augsburg, Germany

<sup>3</sup> University of Sheffield, UK

<sup>4</sup> University of Texas at Austin, USA

2009

CKA est proposée.



# ALGÈBRE DE KLEENE CONCURRENTE

## On Locality and the Exchange Law for Concurrent Processes

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

<sup>1</sup> Microsoft Research Cambridge

<sup>2</sup> Queen Mary University of London

<sup>3</sup> Universität Augsburg

<sup>4</sup> University of Sheffield

2009

2011

CKA est proposée.

Introduction de modèles de  
CKA, comparaison avec la  
logique de séparation.

# ALGÈBRE DE KLEENE CONCURRENTE

## 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 est proposée.

2011

Introduction de modèles de  
CKA, comparaison avec la  
logique de séparation.

2014

Premier théorème de  
complétude (sans la loi  
d'échange), introduction de  
tests dans CKA.

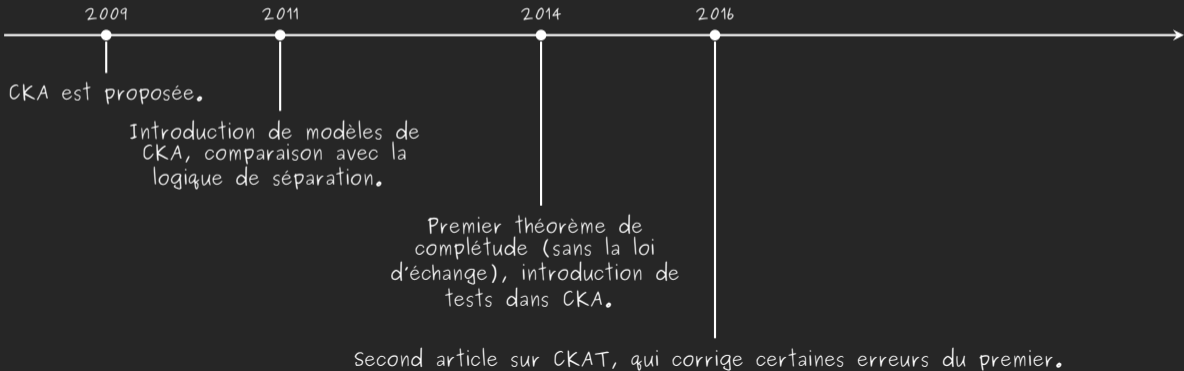
# ALGÈBRE DE KLEENE CONCURRENTE

Concurrent Kleene algebra with tests and branching automata



Peter Jipsen\*, M. Andrew Moshier

Chapman University, Orange, CA 92866, USA



# ALGÈBRE DE KLEENE CONCURRENTE

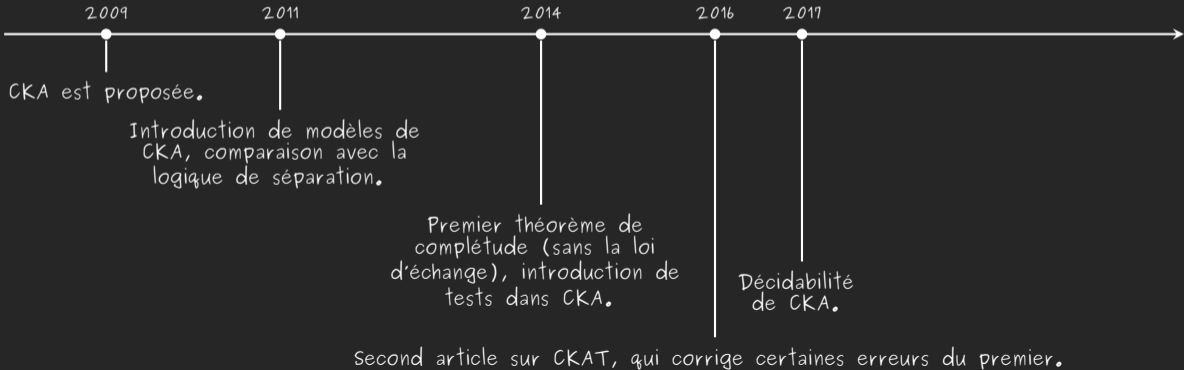
## On Decidability of Concurrent Kleene Algebra<sup>\*†</sup>

Paul Brunet<sup>1</sup>, Damien Pous<sup>2</sup>, and Georg Struth<sup>3</sup>

<sup>1</sup> Univ. Lyon, CNRS, ENS de Lyon, UCB Lyon 1, LIP, France

<sup>2</sup> Univ. Lyon, CNRS, ENS de Lyon, UCB Lyon 1, LIP, France

<sup>3</sup> Department of Computer Science, The University of Sheffield, UK

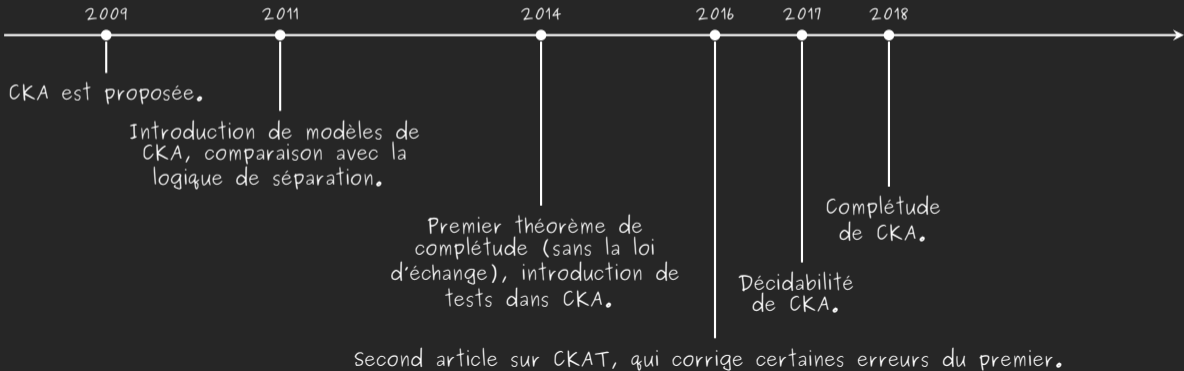


# ALGÈBRE DE KLEENE CONCURRENTE

## Concurrent Kleene Algebra: Free Model and Completeness






Tobias Kappé<sup>(✉)</sup>, Paul Brunet, Alexandra Silva, and Fabio Zanasi

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



# ALGÈBRE DE KLEENE

## 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

2009

2011

2014

2016

2017

2018

2020

## 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

CKA es

de  
la loi  
de

Complétude  
de CKA.

Décidabilité  
de CKA.

CKA avec observations  
(totales/partielles), CKA  
avec boîtes.

Second article sur CKAT, qui corrige certaines erreurs du premier.

# ALGÈBRE DE KLEENE - LA THÉORIE DES LANGAGES RÉGULIERS

$$e, f \in E_A^{ka} ::= 0 \mid 1 \mid a \in A \mid e \cdot f \mid e + f \mid e^*$$

Interprétation : langages réguliers

$$[[ \cdot ]]: E_A^{ka} \rightarrow \mathcal{P}(A^*)$$

exemple :

$$\begin{aligned} [[a \cdot ((a + b) \cdot a)^*]] &= \left\{ \begin{array}{l} \text{mots sur l'alphabet } \{a, b\} \text{ de lon-} \\ \text{gueur impaire et tels qu'une lettre} \\ \text{sur deux est un } a \end{array} \right\} \\ &= \{a, aaa, aba, aaaaa, abaaa, aaaba, ababa, \dots\} \end{aligned}$$

# ALGÈBRE DE KLEENE - LA THÉORIE DES LANGAGES RÉGULIERS

## Les axiomes de KA

$$\begin{array}{lll} e + e = e & e + f = f + e & e + (f + g) = (e + f) + g \\ e + 0 = 0 & e \cdot 1 = e = 1 \cdot e & e \cdot (f \cdot g) = (e \cdot f) \cdot g \\ e \cdot 0 = 0 = 0 \cdot e & e \cdot (f + g) = e \cdot f + e \cdot g & (e + f) \cdot g = e \cdot g + f \cdot g \\ e^* = 1 + e \cdot e^* & e \cdot f \leq f \Rightarrow e^* \cdot f \leq f & \end{array}$$

## Théorème

$$KA \vdash e = f \Leftrightarrow \llbracket e \rrbracket = \llbracket f \rrbracket.$$

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



# KAT - LA THÉORIE DES PROGRAMMES IMPÉRATIFS

## Syntaxe

$$e, f \in \mathbf{E}_{A,B}^{\text{kat}} ::= 0 \mid 1 \mid a \in A \mid t? \mid e \cdot f \mid e + f \mid e^*$$
$$t, t_1, t_2 \in \mathbf{T}_B ::= T \mid \perp \mid \alpha \in B \mid t_1 \wedge t_2 \mid t_1 \vee t_2 \mid \neg t$$

Encode un langage `while` simple :

`if b then p else q`  $\mapsto b? \cdot p + (\neg b)? \cdot q$

`while b do p`  $\mapsto (b? \cdot p)^* \cdot (\neg b)?$

# KAT - LA THÉORIE DES PROGRAMMES IMPÉRATIFS

Syntaxe

$e, f \in E_{A,B}^{kat} ::= 0 \mid 1 \mid a \in A \mid t? \mid e \cdot f \mid e + f \mid e^*$   
 $t, t_1, t_2 \in T_B ::= T \mid \perp \mid \alpha \in B \mid t_1 \wedge t_2 \mid t_1 \vee t_2 \mid \neg t$

Encode un langage while simple :

if  $b$  then  $p$  else  $q \mapsto b? \cdot p + (\neg b)? \cdot q$

while  $b$  do  $p \mapsto (b? \cdot p)^* \cdot (\neg b)?$

# KAT - LA THÉORIE DES PROGRAMMES IMPÉRATIFS

Syntaxe

$e, f \in E_{A,B}^{kat} ::= 0 \mid 1 \mid a \in A \mid t? \mid e \cdot f \mid e + f \mid e^*$   
 $t, t_1, t_2 \in T_B ::= T \mid \perp \mid \alpha \in B \mid t_1 \wedge t_2 \mid t_1 \vee t_2 \mid \neg t$

Encode un langage while simple :

if  $b$  then  $p$  else  $q \mapsto b? \cdot p + (\neg b)? \cdot q$

while  $b$  do  $p \mapsto (b? \cdot p)^* \cdot (\neg b)?$

# KAT - LA THÉORIE DES PROGRAMMES IMPÉRATIFS

Syntaxe

$e, f \in E_{A,B}^{kat} ::= 0 \mid 1 \mid a \in A \mid t? \mid e \cdot f \mid e + f \mid e^*$   
 $t, t_1, t_2 \in T_B ::= T \mid \perp \mid \alpha \in B \mid t_1 \wedge t_2 \mid t_1 \vee t_2 \mid \neg t$

Encode un langage while simple :

if  $b$  then  $p$  else  $q \mapsto b? \cdot p + (\neg b)? \cdot q$

while  $b$  do  $p \mapsto (b? \cdot p)^* \cdot (\neg b)?$

# KAT - LA THÉORIE DES PROGRAMMES IMPÉRATIFS

Syntaxe

Annotations: skip, action atomique, test, composition séquentielle, choix non-déterministe, boucle non-déterministe

tuer l'exécution

$$e, f \in E_{A,B}^{\text{kat}} ::= 0 \mid 1 \mid a \in A \mid t? \mid e \cdot f \mid e + f \mid e^*$$
$$t, t_1, t_2 \in T_B ::= T \mid \perp \mid \alpha \in B \mid t_1 \wedge t_2 \mid t_1 \vee t_2 \mid \neg t$$

Encode un langage while simple :

if  $b$  then  $p$  else  $q \mapsto b? \cdot p + (\neg b)? \cdot q$

while  $b$  do  $p \mapsto (b? \cdot p)^* \cdot (\neg b)?$

# KAT - LA THÉORIE DES PROGRAMMES IMPÉRATIFS

## Syntaxe

$e, f \in \mathbb{E}_{A,B}^{\text{kat}} ::= 0 \mid 1 \mid a \in A \mid t? \mid e \cdot f \mid e + f \mid e^*$   
 $t, t_1, t_2 \in \mathbb{T}_B ::= T \mid \perp \mid \alpha \in B \mid t_1 \wedge t_2 \mid t_1 \vee t_2 \mid \neg t$

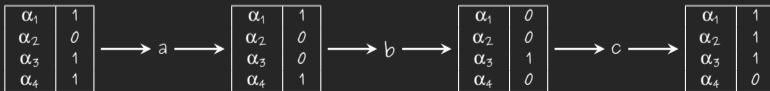
Encode un langage while simple :

if  $b$  then  $p$  else  $q \mapsto b? \cdot p + (\neg b)? \cdot q$

while  $b$  do  $p \mapsto (b? \cdot p)^* \cdot (\neg b)?$

## Interprétation : langages de chaînes gardées

chaînes gardées : séquences alternant états  $\in 2^B$  & actions  $\in A$ .



# KAT - LA THÉORIE DES PROGRAMMES IMPÉRATIFS

Les axiomes de KAT :

☞ Les axiomes de KA.

☞ Pour les tests, les axiomes des algèbres booléennes.

☞ Les axiomes "glue" suivants :

$$(t_1 \vee t_2)? = t_1? + t_2? \quad (t_1 \wedge t_2)? = t_1? \cdot t_2? \quad \top? = 1 \quad \perp? = 0$$

## Théorème

$$\text{KAT} \vdash e = f \Leftrightarrow \llbracket e \rrbracket = \llbracket f \rrbracket.$$

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

# KAT - LA THÉORIE DES PROGRAMMES IMPÉRATIFS

Les axiomes de KAT :

👉 Les axiomes de KA.

👉 Pour les tests, les axiomes des algèbres booléennes.

👉 Les axiomes "glue" suivants :

$$(t_1 \vee t_2)? = t_1? + t_2? \quad (t_1 \wedge t_2)? = t_1? \cdot t_2? \quad \top? = 1 \quad \perp? = 0$$

## Théorème

$$\text{KAT} \vdash e = f \Leftrightarrow \llbracket e \rrbracket = \llbracket f \rrbracket.$$

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

Encode la logique de Hoare propositionnelle :

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



# KAT - LA THÉORIE DES PROGRAMMES IMPÉRATIFS

Les axiomes de KAT :

👉 Le axiomes de KA.

👉 Pour les tests, les axiomes des algèbres booléennes.

👉 Les axiomes "glue" suivants :

$$(t_1 \vee t_2)? = t_1? + t_2? \quad (t_1 \wedge t_2)? = t_1? \cdot t_2? \quad \top? = 1 \quad \perp? = 0$$

## Théorème

$$\text{KAT} \vdash e = f \Leftrightarrow \llbracket e \rrbracket = \llbracket f \rrbracket.$$

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

Encode la logique de Hoare propositionnelle :

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

Peut-on faire pareil pour les programmes concurrents ?

# AVANCÉES RÉCENTES SUR CKA

## Plan

I. Introduction

II. Algèbre de Kleene concurrente

III. Observations booléennes

IV. Observations partielles

V. Boîtes

VI. Travaux en cours et à venir

# AVANCÉES RÉCENTES SUR CKA

## Plan

I. Introduction



II. Algèbre de Kleene concurrente

III. Observations booléennes

IV. Observations partielles

V. Boîtes

VI. Travaux en cours et à venir

# BI-ALGÈBRE DE KLEENE

$$e, f \in E_A^{\text{bika}} ::= 1 \mid 0 \mid a \in A \mid e \cdot f \mid e \parallel f \mid e + f \mid e^* \mid e'$$

## Définition

Une bi-algèbre de Kleene est une structure  $\langle \mathcal{A}, 0, 1, \cdot, \parallel, +, *, ! \rangle$  telle que :

- ☞  $\langle \mathcal{A}, 0, 1, \cdot, +, * \rangle$  est une KA
- ☞  $\langle \mathcal{A}, 0, 1, \parallel, +, ! \rangle$  est une KA commutative.

# BI-ALGÈBRE DE KLEENE

$$e, f \in E_A^{\text{bika}} ::= 1 \mid 0 \mid a \in A \mid e \cdot f \mid e \parallel f \mid e + f \mid e^* \mid e'$$

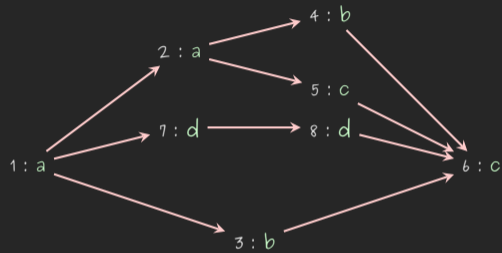
## Définition

Une bi-algèbre de Kleene est une structure  $\langle \mathcal{A}, 0, 1, \cdot, \parallel, +, *, ! \rangle$  telle que :

- ☞  $\langle \mathcal{A}, 0, 1, \cdot, +, * \rangle$  est une KA
- ☞  $\langle \mathcal{A}, 0, 1, \parallel, +, ! \rangle$  est une KA commutative.

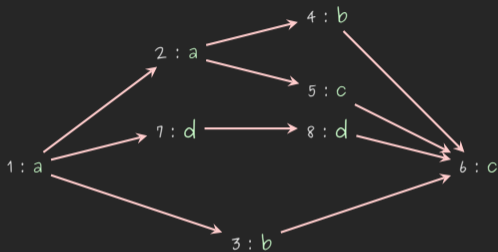
À quoi ressemble la bi-KA libre ?

# POMSETS - TRACES CONCURRENTES



# POMSETS - TRACES CONCURRENTES

A est un alphabet d'actions.



ensemble fini d'évènements

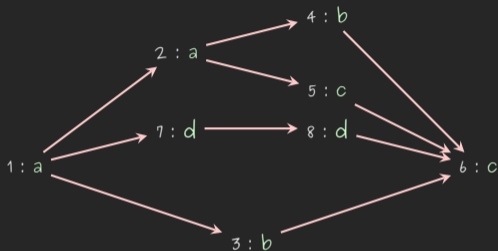
fonction d'étiquetage  
 $: E_P \rightarrow A$

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

ordre partiel  
 $\subseteq E_P \times E_P$

# POMSETS - TRACES CONCURRENTES

A est un alphabet d'actions.



ensemble fini d'évènements

fonction d'étiquetage  
 $: E_P \rightarrow A$

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

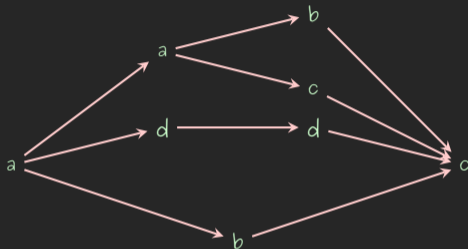
ordre partiel  
 $\subseteq E_P \times E_P$

À isomorphisme près  $\equiv$ .



# POMSETS - TRACES CONCURRENTES

A est un alphabet d'actions.



ensemble fini d'évènements

fonction d'étiquetage  
 $: E_P \rightarrow A$

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

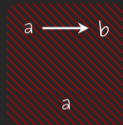
ordre partiel  
 $\subseteq E_P \times E_P$

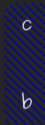
À isomorphisme près  $\equiv$ .

# COMBINER LES POMSETS

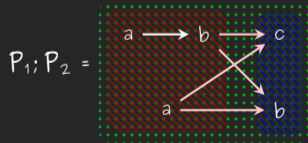
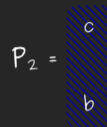
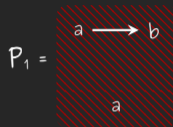
$a =$  

$1 =$  

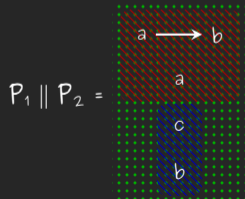
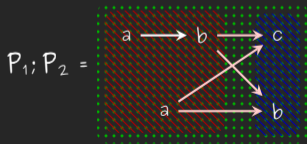
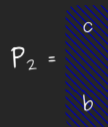
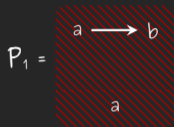
$P_1 =$  

$P_2 =$  

# COMBINER LES POMSETS



# COMBINER LES POMSETS



# COMPLÉTUDE DE BIKA

$$\llbracket 1 \rrbracket := \{1\}$$

$$\llbracket a \rrbracket := \{a\}$$

$$\llbracket e \cdot f \rrbracket := \{P;Q \mid P \in \llbracket e \rrbracket, Q \in \llbracket f \rrbracket\}$$

$$\llbracket e^* \rrbracket := \{P_1; \dots; P_n \mid n \in \mathbb{N}, P_i \in \llbracket e \rrbracket\}$$

$$\llbracket 0 \rrbracket := \emptyset$$

$$\llbracket e + f \rrbracket := \llbracket e \rrbracket \cup \llbracket f \rrbracket$$

$$\llbracket e \parallel f \rrbracket := \{P \parallel Q \mid P \in \llbracket e \rrbracket, Q \in \llbracket f \rrbracket\}$$

$$\llbracket e' \rrbracket := \{P_1 \parallel \dots \parallel P_n \mid n \in \mathbb{N}, P_i \in \llbracket e \rrbracket\}$$

## Théorème

$$\text{biKA} \vdash e = f \Leftrightarrow \llbracket e \rrbracket \equiv \llbracket f \rrbracket.$$

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

# ENTRELACEMENTS ET SUBSOMPTION

Loi d'échange faible

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

$$\begin{array}{ccc} a \longrightarrow c & & a \longrightarrow c \\ & \searrow \nearrow & \\ b \longrightarrow d & \sqsubseteq & b \longrightarrow d \end{array}$$

$P \sqsubseteq Q$  quand il y a un homomorphisme de  $Q$  à  $P$ , c.à.d. une application bijective  $\varphi: E_Q \rightarrow E_P$  telle que  $\lambda_P \circ \varphi = \lambda_Q$  et  $\varphi(\leq_Q) \subseteq \leq_P$ .

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

# ALGÈBRE DE KLEENE CONCURRENTE

Loi d'échange faible

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

Pas d'itération parallèle

CKA

Une algèbre de Kleene concurrente est une bi-algèbre de Kleene faible  $(\mathcal{A}, 0, 1, \cdot, \parallel, +, *)$  qui satisfait la loi d'échange faible.

# COMPLÉTUDE ET DÉCIDABILITÉ DE CKA

## Théorème

Tester que deux expressions dénotent le même langage clos est un problème ExpSpace-complet.

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

## Théorème

$$\text{CKA} \vdash e = f \Leftrightarrow \llbracket e \rrbracket^{\mathbb{E}} = \llbracket f \rrbracket^{\mathbb{E}}.$$

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




# AVANCÉES RÉCENTES SUR CKA

## Plan

I. Introduction

II. Algèbre de Kleene concurrente

 III. Observations booléennes

IV. Observations partielles

V. Boîtes

VI. Travaux en cours et à venir

# CONCURRENT KLEENE ALGEBRA WITH TESTS

## Slogan

☞ KAT : KA avec une sous-algèbre booléenne.

☞ CKAT : CKA avec une sous-algèbre booléenne.

# CONCURRENT KLEENE ALGEBRA WITH TESTS

## Slogan

☞ KAT : KA avec une sous-algèbre booléenne.

☞ CKAT : CKA avec une sous-algèbre booléenne.

$$\begin{aligned} t? \cdot p \cdot (\neg t)? &\leq p \parallel (t? \cdot (\neg t)?) && \text{(axiomes de CKA)} \\ &= p \parallel (t \wedge \neg t)? && (a \wedge b)? = a? \cdot b? \\ &= p \parallel \perp? && \text{(axiomes booléens)} \\ &= p \parallel 0 && (\perp? = 0) \\ &= 0 && \text{(axiomes de CKA)} \end{aligned}$$

# CONCURRENT KLEENE ALGEBRA WITH TESTS

## Slogan

☞ KAT : KA avec une sous-algèbre booléenne.

☞ CKAT : CKA avec une sous-algèbre booléenne.



$$\begin{aligned} t? \cdot p \cdot (\neg t)? &\leq p \parallel (t? \cdot (\neg t)?) && \text{(axiomes de CKA)} \\ &= p \parallel (t \wedge \neg t)? && (a \wedge b)? = a? \cdot b? \\ &= p \parallel \perp? && \text{(axiomes booléens)} \\ &= p \parallel 0 && (\perp? = 0) \\ &= 0 && \text{(axiomes de CKA)} \end{aligned}$$

↔ Pour tout programme et toute assertion, le triplet  $\{t\} p \{t\}$  est valide.

↔ Chaque test est un invariant de tous les programmes.

# ~~CONCURRENT KLEENE ALGEBRA WITH TESTS~~

## slogan

☛ KAT : KA avec une sous-algèbre booléenne.

☛ CKAT : CKA avec une sous-algèbre booléenne.



$$\begin{aligned} t? \cdot p \cdot (\neg t)? &\leq p \parallel (t? \cdot (\neg t)?) && \text{(axiomes de CKA)} \\ &= p \parallel (t \wedge \neg t)? && (a \wedge b)? = a? \cdot b? \\ &= p \parallel \perp? && \text{(axiomes booléens)} \\ &= p \parallel 0 && (\perp? = 0) \\ &= 0 && \text{(axiomes de CKA)} \end{aligned}$$

↔ Pour tout programme et toute assertion, le triplet  $\{t\} p \{t\}$  est valide.

↔ Chaque test est un invariant de tous les programmes.

# À QUI LA FAUTE?

$$\begin{aligned}t^? \cdot p \cdot (-t)^? &\leq p \parallel (t^? \cdot (-t)^?) \\ &= p \parallel (t \wedge -t)^? \\ &= p \parallel \perp^? \\ &= p \parallel 0 = 0\end{aligned}$$

(axiomes de CKA)

$$(a \wedge b)^? = a^? \cdot b^?$$

(axiomes booléens)

( $\perp^? = 0$  + axiomes de CKA)

# À QUI LA FAUTE?

$$\begin{aligned}t^? \cdot p \cdot (-t)^? &\leq p \parallel (t^? \cdot (-t)^?) \\ &= p \parallel (t \wedge -t)^? \\ &= p \parallel \perp^? \\ &= p \parallel 0 = 0\end{aligned}$$

(axiomes de CKA)

$$(a \wedge b)^? = a^? \cdot b^?$$

(axiomes booléens)

( $\perp^? = 0$  + axiomes de CKA)

# À QUI LA FAUTE?

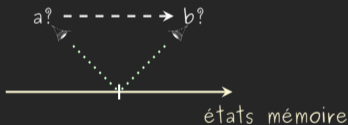
$$\begin{aligned}t^? \cdot p \cdot (-t)^? &\leq p \parallel (t^? \cdot (-t)^?) \\ &= p \parallel (t \wedge -t)^? \\ &= p \parallel \perp^? \\ &= p \parallel 0 = 0\end{aligned}$$

(axiomes de CKA)

$$(a \wedge b)^? = a^? \cdot b^?$$

(axiomes booléens)

( $\perp^? = 0$  + axiomes de CKA)





# À QUI LA FAUTE?

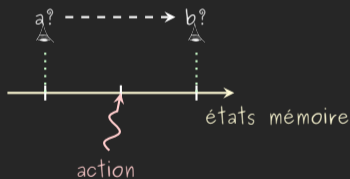
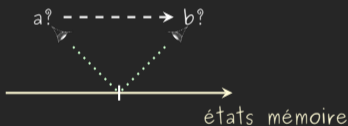
$$\begin{aligned} t? \cdot p \cdot (-t)? &\leq p \parallel (t? \cdot (-t)? \\ &= p \parallel (t \wedge -t)? \\ &= p \parallel \perp? \\ &= p \parallel 0 = 0 \end{aligned}$$

(axiomes de CKA)

$$(a \wedge b)? = a? \cdot b?$$

(axiomes booléens)

( $\perp? = 0$  + axiomes de CKA)



# À QUI LA FAUTE?

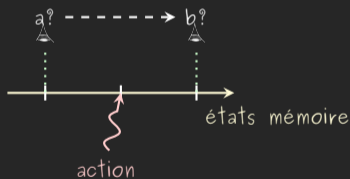
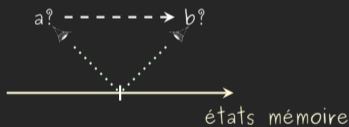
$$\begin{aligned} t? \cdot p \cdot (-t)? &\leq p \parallel (t? \cdot (-t)?) \\ &= p \parallel (t \wedge -t)? \\ &= p \parallel \perp? \\ &= p \parallel 0 = 0 \end{aligned}$$

(axiomes de CKA)

$$\boxed{\cancel{(a \wedge b)? \leq a? \cdot b?}}$$

(axiomes booléens)

( $\perp? = 0$  + axiomes de CKA)



$$\boxed{(a \wedge b)? \leq a? \cdot b?}$$

# CKAO - SYNTAXE

$e, f \in E_{A,B}^{ckao} ::= 0 \mid 1 \mid a \in A \mid t? \mid e \cdot f \mid e \parallel f \mid e + f \mid e^*$

$t, t_1, t_2 \in O_A^{bool} ::= T \mid \perp \mid \alpha \in B \mid t_1 \wedge t_2 \mid t_1 \vee t_2 \mid \neg t$

## Axiomes de CKAO

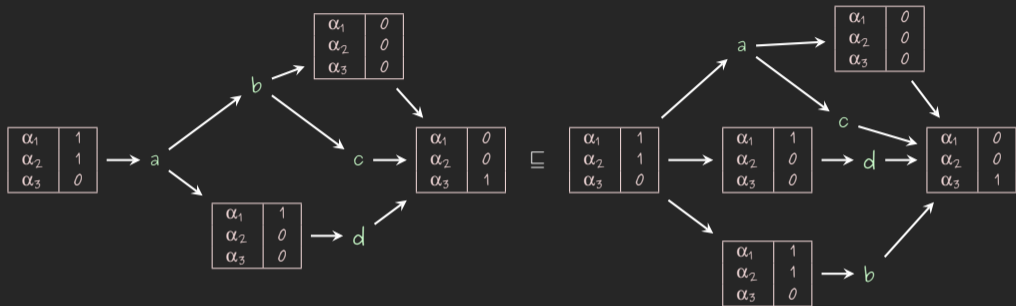
- ☞ Tous les axiomes de CKA.
- ☞ Pour les observations, les axiomes booléens.
- ☞ Les axiomes "glue" suivants :

$$(t_1 \vee t_2)? = t_1? + t_2?$$

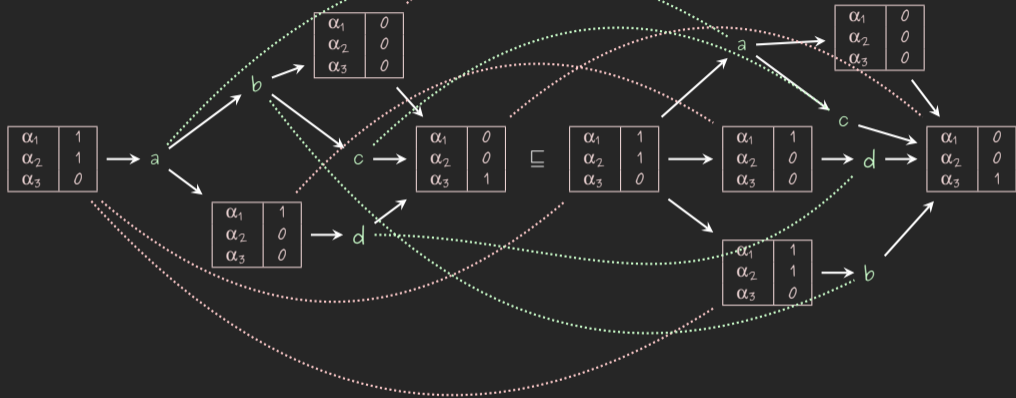
$$(t_1 \wedge t_2)? \leq t_1? \cdot t_2?$$

$$\perp? = 0$$

# CKAO - MODÈLE



# CKAO - MODÈLE





## INTERLUDE - (C)KA MODULO HYPOTHÈSES

$H$  : ensemble d'hypothèses  $e \leq f$  sur un alphabet  $A$  donné.

☞ structure algébrique de l'alphabet (e.g.  $\alpha \wedge \beta = \beta \wedge \alpha$ ) ;

☞ propriétés dynamiques (e.g.  $\alpha \leq \alpha \cdot \alpha$ )

☞ autres propriétés du système modélisé.

### Théorème

$$\text{CKA} + H \vdash e = f \Rightarrow \llbracket e \rrbracket \downarrow^H = \llbracket f \rrbracket \downarrow^H$$

☞ 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


# AVANCÉES RÉCENTES SUR CKA

## Plan

I. Introduction

II. Algèbre de Kleene concurrente

III. Observations booléennes

 IV. Observations partielles

V. Boîtes

VI. Travaux en cours et à venir



# LITMUS TEST - COHÉRENCE SÉQUENTIELLE

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

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

```
{ r0 == 1 || r1 == 1 }
```

Ingrédients :

☞ Affectations  $x \leftarrow 1$

☞ Observations  $r_0 \mapsto 0$

# DE QUELLES OBSERVATIONS AVONS NOUS BESOIN?

Première tentative : algèbre booléenne

☞ Observations atomiques :  $V_{AR} \mapsto V_{AL}$

e.g.  $r_0 \mapsto 1$

# DE QUELLES OBSERVATIONS AVONS NOUS BESOIN?

Première tentative : algèbre booléenne

☞ Observations atomiques :  $V_{AR} \mapsto V_{AL}$

e.g.  $r_0 \mapsto 1$

☞ Formules booléennes : ensembles d'états mémoire  $V_{AR} \rightarrow V_{AL}$

e.g. 

$r_0$	1
$r_1$	0

# DE QUELLES OBSERVATIONS AVONS NOUS BESOIN?

Première tentative : algèbre booléenne

👉 Observations atomiques :  $V_{AR} \mapsto V_{AL}$  e.g.  $r_0 \mapsto 1$

👉 Formules booléennes : ensembles d'états mémoire  $V_{AR} \rightarrow V_{AL}$  e.g. 

$r_0$	1
$r_1$	0

👉 Affectations :  $\sum_{s \in \text{state}} s \cdot (v \leftarrow n) \cdot s[v \mapsto n]$ , c.à.d.

$$\llbracket x \leftarrow 1 \rrbracket := \left\{ \begin{array}{|c|c|} \hline x & 0 \\ \hline y & 0 \\ \hline \end{array} \rightarrow [x \leftarrow 1] \rightarrow \begin{array}{|c|c|} \hline x & 1 \\ \hline y & 0 \\ \hline \end{array}, \begin{array}{|c|c|} \hline x & 0 \\ \hline y & 1 \\ \hline \end{array} \rightarrow [x \leftarrow 1] \rightarrow \begin{array}{|c|c|} \hline x & 1 \\ \hline y & 1 \\ \hline \end{array}, \dots \right\}$$

# DE QUELLES OBSERVATIONS AVONS NOUS BESOIN?

Première tentative : algèbre booléenne

👉 Observations atomiques :  $V_{AR} \mapsto V_{AL}$

e.g.  $r_0 \mapsto 1$

👉 Formules booléennes : ensembles d'états mémoire  $V_{AR} \rightarrow V_{AL}$

e.g. 

$r_0$	1
$r_1$	0

👉 Affectations :  $\sum_{s \in \text{state}} s \cdot (v \leftarrow n) \cdot s[v \mapsto n]$ , c.à.d.

$$\llbracket x \leftarrow 1 \rrbracket := \left\{ \begin{array}{|c|c|} \hline x & 0 \\ \hline y & 0 \\ \hline \end{array} \rightarrow [x \leftarrow 1] \rightarrow \begin{array}{|c|c|} \hline x & 1 \\ \hline y & 0 \\ \hline \end{array}, \begin{array}{|c|c|} \hline x & 0 \\ \hline y & 1 \\ \hline \end{array} \rightarrow [x \leftarrow 1] \rightarrow \begin{array}{|c|c|} \hline x & 1 \\ \hline y & 1 \\ \hline \end{array}, \dots \right\}$$

Problème : composition parallèle ?

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



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

# ALGÈBRE D'OBSERVATIONS PARTIELLES

Idée : au lieu d'états mémoire  $V_{AR} \rightarrow V_{AL}$ , on considère des fonctions partielles  $V_{AR} \rightarrow V_{AL}$ .

## PCDL d'observations

$$t, t_1, t_2 \in \mathcal{O}_A^{\text{pocka}} ::= \top \mid \perp \mid \alpha \in \mathcal{B} \mid t_1 \wedge t_2 \mid t_1 \vee t_2 \mid \bar{t}$$

Mêmes axiomes que BA pour  $\vee, \wedge, \top, \perp$ , plus :

$$\text{☞ } p \leq \bar{q} \Leftrightarrow p \wedge q = \perp$$

définition du pseudo-complément

$$\text{☞ } \overline{v \mapsto n} = \bigvee_{m \neq n} v \mapsto m$$

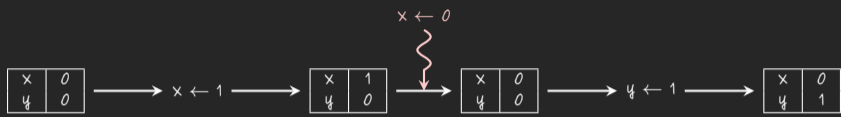
PCDL : trellis distributif pseudo-complémenté

## Théorème

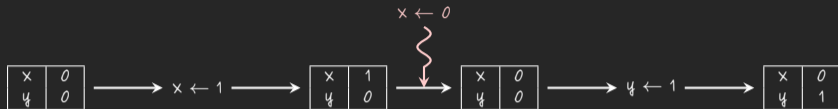
$$\text{POCKA} \vdash e = f \Leftrightarrow \llbracket e \rrbracket \downarrow^{\text{pocka}} = \llbracket f \rrbracket \downarrow^{\text{pocka}}.$$

Wagemaker, B., Docherty, Kappé, Rot, & Silva, "Partially Observable Concurrent Kleene Algebra", CONCUR '20

# CAUSALITÉ VS COMPOSITIONNALITÉ



# CAUSALITÉ VS COMPOSITIONNALITÉ

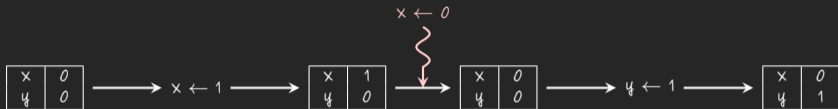


Solution : il faut « clore » le système explicitement.

$\llbracket e \rrbracket \rightarrow \llbracket e \rrbracket \cap \text{CausalPomsets}.$



# CAUSALITÉ VS COMPOSITIONNALITÉ



Solution : il faut « clore » le système explicitement.

$$[[e]] \rightarrow [[e]] \cap \text{CausalPomsets}.$$

Litmus test :

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

$$[[t]] \cap \text{CausalPomsets} = \emptyset$$

# AVANCÉES RÉCENTES SUR CKA

## Plan

I. Introduction

II. Algèbre de Kleene concurrente

III. Observations booléennes

IV. Observations partielles

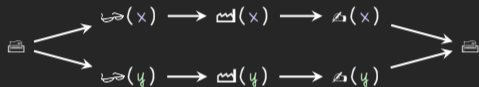


V. Boîtes

VI. Travaux en cours et à venir

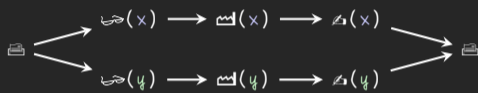
# EXCLUSION MUTUELLE

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



# EXCLUSION MUTUELLE

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

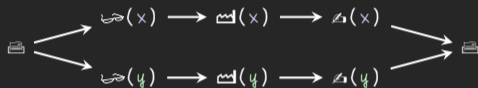


# EXCLUSION MUTUELLE

```

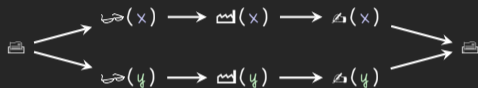
print(counter);
||
x:=counter;      y:=counter;
x:=x+1;          y:=y+1;
counter:=x;      counter:=y;
||
print(counter);

```



# EXCLUSION MUTUELLE

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

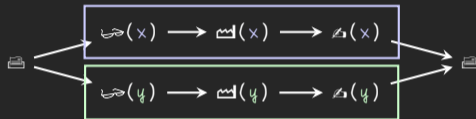


lock icon → lock icon(x) → lock icon(x) → lock icon(x) → lock icon(y) → lock icon(y) → lock icon(y) → lock icon

lock icon → lock icon(x) → lock icon(y) → lock icon(x) → lock icon(y) → lock icon(x) → lock icon(y) → lock icon

# EXCLUSION MUTUELLE

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

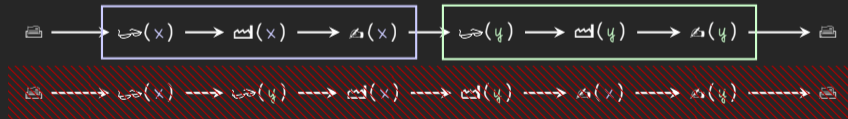
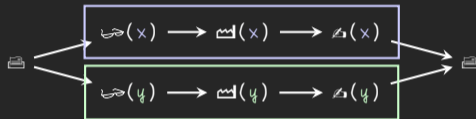


lock → lock(x) → read(x) → update(x) → lock(y) → read(y) → update(y) → lock

lock → lock(x) → lock(y) → read(x) → read(y) → update(x) → update(y) → lock

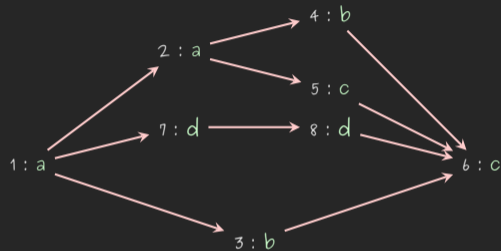
# EXCLUSION MUTUELLE

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





# POMSETS EMBOÎTÉS



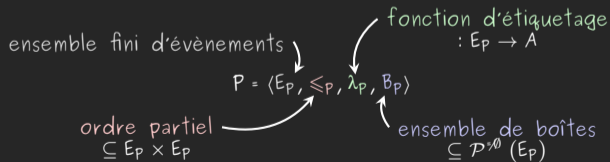
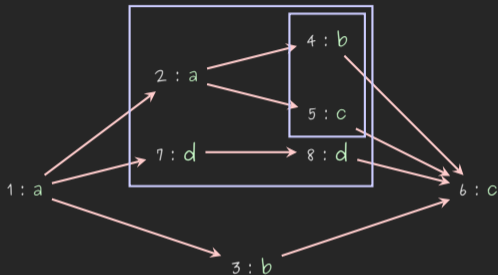
ensemble fini d'évènements  $\rightarrow$

fonction d'étiquetage  $: E_p \rightarrow A$

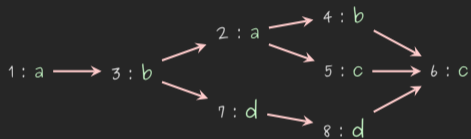
$P = \langle E_p, \leq_p, \lambda_p \rangle$

ordre partiel  $\subseteq E_p \times E_p$

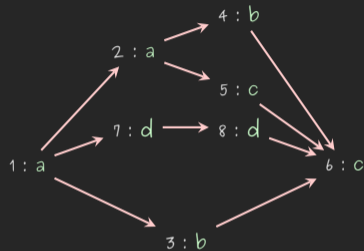
# POMSETS EMBOÎTÉS



# SUBSOMPTION AVEC BOÎTES



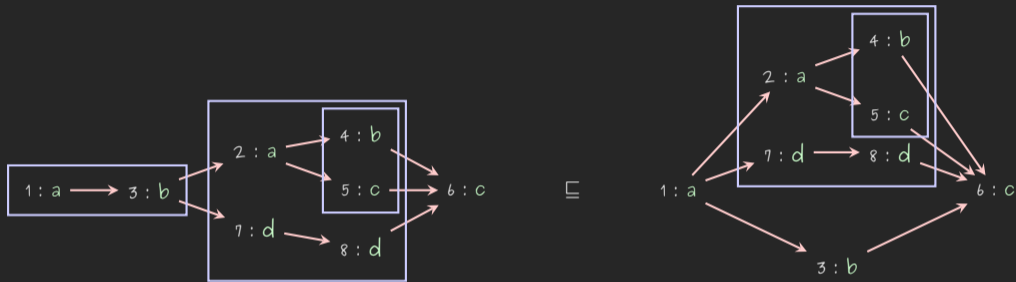
$\sqsubseteq$



$P \sqsubseteq Q$  lorsque'il existe un homomorphisme de  $Q$  vers  $P$ , c.à.d. une bijection  $\varphi: E_Q \rightarrow E_P$  telle que :

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

# SUBSUMPTION AVEC BOÎTES



$P \sqsubseteq Q$  lorsque qu'il existe un homomorphisme de  $Q$  vers  $P$ , c.à.d. une bijection  $\varphi: E_Q \rightarrow E_P$  telle que :

- 1)  $\lambda_P \circ \varphi = \lambda_Q$
- 2)  $\varphi(\leq_Q) \subseteq \leq_P$
- 3)  $\varphi(B_P) \subseteq B_Q$

# AXIOMATISATION

(On omet l'étoile de la signature)

$$[[e]] = [e]$$

$$[1] = 1$$

$$[0] = 0$$

$$[e + f] = [e] + [f]$$

$$[e] \leq e$$

**Théorème**

Demi-anneau concurrent, i.e. CKA sans étoile

$$CSR + B \vdash e = f \Leftrightarrow [[e]] = [[f]].$$

B. & Pym, "Pomsets with Boxes : Protection, Separation, and Locality in Concurrent Kleene Algebra.", FSCD '20

# AXIOMATISATION

(On omet l'étoile de la signature)



$$[[e]] = [e]$$

$$[1] = 1$$

$$[0] = 0$$

$$[e + f] = [e] + [f]$$

$$[e] \leq e$$

**Théorème**

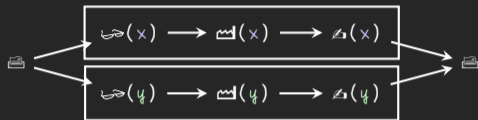
Demi-anneau concurrent, i.e. CKA sans étoile

$$CSR + B \vdash e = f \Leftrightarrow [[e]] = [[f]].$$

B. & Pym, "Pomsets with Boxes : Protection, Separation, and Locality in Concurrent Kleene Algebra.", FSCD '20

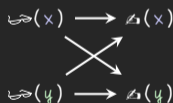
# EXCLUSION MUTUELLE - II

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



Enfreindre l'exclusion mutuelle

↔ admettre une exécution contenant le « motif » suivant :

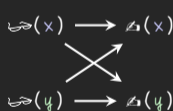


# EXCLUSION MUTUELLE - II

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



Enfreindre l'exclusion mutuelle  $\leftrightarrow$  admettre une exécution contenant le « motif » suivant :



Problème : cette propriété ne s'exprime pas sous la forme  
programme  $\leq$  spécification.



# POMSET LOGIC<sup>1</sup>

$\varphi, \psi ::= \perp \mid a \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \varphi \blacktriangleright \psi \mid \varphi * \psi \mid [\varphi] \mid \langle \varphi \rangle$

☞  $P \models \varphi \blacktriangleright \psi$  ssi  $\exists P_1, P_2$  tels que  $P \sqsupseteq P_1; P_2$  et  $P_1 \models \varphi$  et  $P_2 \models \psi$

☞  $P \models \varphi * \psi$  ssi  $\exists P_1, P_2$  tels que  $P \sqsupseteq P_1 \parallel P_2$  et  $P_1 \models \varphi$  et  $P_2 \models \psi$

☞  $P \models [\varphi]$  ssi  $\exists Q$  tel que  $P \sqsupseteq [Q]$  et  $Q \models \varphi$

☞  $P \models \langle \varphi \rangle$  ssi  $\exists P', Q$  tels que  $P \sqsupseteq P'$  et  $P' \oplus Q$  et  $Q \models \varphi$ .

## Théorème

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

---

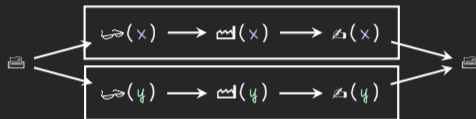
1. À ne pas confondre avec la logique du même nom proposée par Christian Rétoré & Alain Lecomte en 1995

# EXCLUSION MUTUELLE - III

```

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

```



Enfreindre l'exclusion mutuelle

↔ admettre une exécution contenant le « motif » suivant :



$$\leftrightarrow P \models ((\text{lock}(x) * \text{lock}(y)) \triangleright (\text{read}(x) * \text{read}(y)))$$

# AVANCÉES RÉCENTES SUR CKA

## Plan


I. Introduction

II. Algèbre de Kleene concurrente

III. Observations booléennes

IV. Observations partielles

V. Boîtes

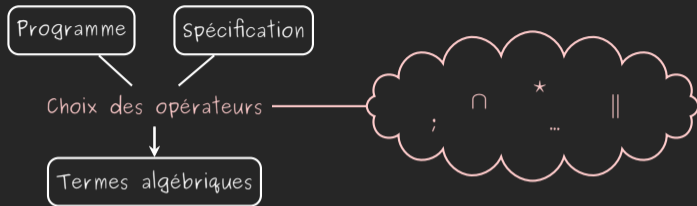
 VI. Travaux en cours et à venir

# PROJET DE RECHERCHE : ALGÈBRES DE PROGRAMMES

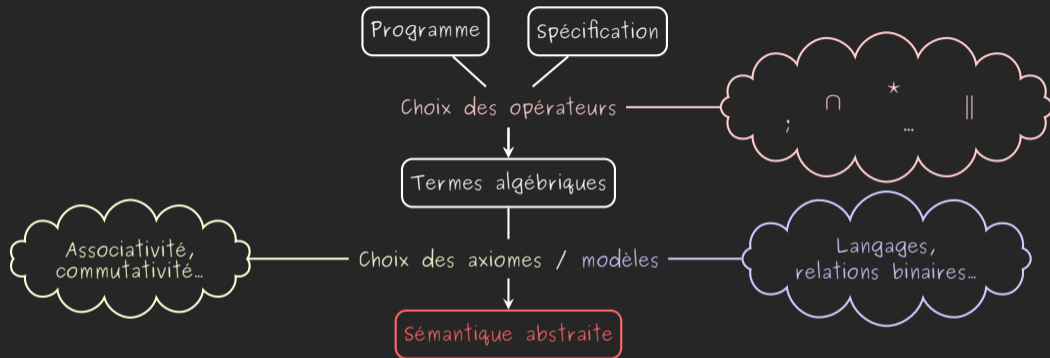
Programme

Spécification

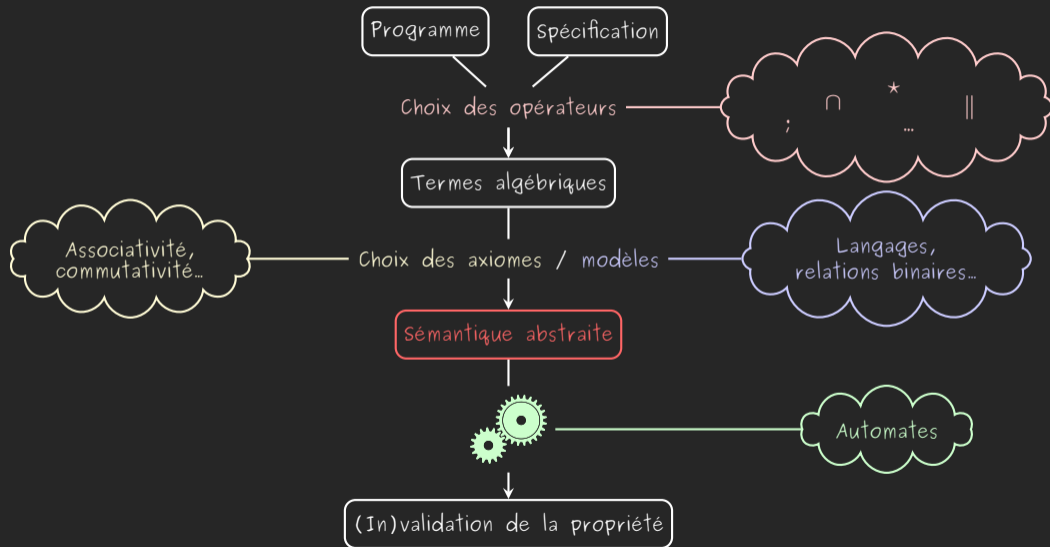
# PROJET DE RECHERCHE : ALGÈBRES DE PROGRAMMES



# PROJET DE RECHERCHE : ALGÈBRES DE PROGRAMMES



# PROJET DE RECHERCHE : ALGÈBRES DE PROGRAMMES



# MÉTA-THÉORIE

☞ Construction modulaire d'algèbres de programme



# MÉTA-THÉORIE

- ☞ Construction modulaire d'algèbres de programme
  - ▶ Quotienter un modèle : algèbres à hypothèses

# MÉTA-THÉORIE

☞ Construction modulaire d'algèbres de programme

▶ Quotienter un modèle : algèbres à hypothèses

▶ KA : Doumane, Kuperberg, Pous, & Pradic, "Kleene Algebra with Hypotheses", FOSSaCS '19

# MÉTA-THÉORIE

## ☞ Construction modulaire d'algèbres de programme

### ▶ Quotienter un modèle : algèbres à hypothèses

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

# MÉTA-THÉORIE

## ☞ Construction modulaire d'algèbres de programme

- ▶ Quotienter un modèle : algèbres à hypothèses
  - ▶ KA : Doumane, Kuperberg, Pous, & Pradic, "Kleene Algebra with Hypotheses", FOSSaCS '19
  - ▶ CKA : Kappé, B., Silva, Wagemaker, & Zanasi, "Concurrent Kleene Algebra with Observations : from Hypotheses to Completeness", FOSSaCS '20
  - ▶ en général ?

# MÉTA-THÉORIE

## ☞ Construction modulaire d'algèbres de programme

- ▶ Quotienter un modèle : algèbres à hypothèses
  - ▶ KA : Doumane, Kuperberg, Pous, & Pradic, "Kleene Algebra with Hypotheses", FOSSaCS '19
  - ▶ CKA : Kappé, B., Silva, Wagemaker, & Zanasi, "Concurrent Kleene Algebra with Observations : from Hypotheses to Completeness", FOSSaCS '20
  - ▶ en général ?
- ▶ Donner une structure à l'alphabet : algèbres empilées (e.g. KAT, CKAO, POCKA, ...)

# MÉTA-THÉORIE

## 👉 Construction modulaire d'algèbres de programme

- ▶ Quotienter un modèle : algèbres à hypothèses
  - ▶ KA : Doumane, Kuperberg, Pous, & Pradic, "Kleene Algebra with Hypotheses", FOSSaCS '19
  - ▶ CKA : Kappé, B., Silva, Wagemaker, & Zanasi, "Concurrent Kleene Algebra with Observations : from Hypotheses to Completeness", FOSSaCS '20
  - ▶ en général ?
- ▶ Donner une structure à l'alphabet : algèbres empilées (e.g. KAT, CKAO, POCKA, ...)
- ▶ Ajouter des opérateurs : "Fusion" d'algèbres

# MÉTA-THÉORIE

## ☞ Construction modulaire d'algèbres de programme

- ▶ Quotienter un modèle : algèbres à hypothèses
  - ▶ KA : Doumane, Kuperberg, Pous, & Pradic, "Kleene Algebra with Hypotheses", FOSSaCS '19
  - ▶ CKA : Kappé, B., Silva, Wagemaker, & Zanasi, "Concurrent Kleene Algebra with Observations : from Hypotheses to Completeness", FOSSaCS '20
  - ▶ en général ?
- ▶ Donner une structure à l'alphabet : algèbres empilées (e.g. KAT, CKAO, POCKA, ...)
- ▶ Ajouter des opérateurs : "Fusion" d'algèbres
  - ▶ (e.g. KA + KA commutative = biKA)

# MÉTA-THÉORIE

## ☞ Construction modulaire d'algèbres de programme

- ▶ Quotienter un modèle : algèbres à hypothèses
  - ▶ KA : Doumane, Kuperberg, Pous, & Pradic, "Kleene Algebra with Hypotheses", FOSSaCS '19
  - ▶ CKA : Kappé, B., Silva, Wagemaker, & Zanasi, "Concurrent Kleene Algebra with Observations : from Hypotheses to Completeness", FOSSaCS '20
  - ▶ en général ?
- ▶ Donner une structure à l'alphabet : algèbres empilées (e.g. KAT, CKAO, POCKA, ...)
- ▶ Ajouter des opérateurs : "Fusion" d'algèbres
  - ▶ (e.g. KA + KA commutative = biKA)

## ☞ Techniques :



# MÉTA-THÉORIE

## ☞ Construction modulaire d'algèbres de programme

- ▶ Quotienter un modèle : algèbres à hypothèses
  - ▶ KA : Doumane, Kuperberg, Pous, & Pradic, "Kleene Algebra with Hypotheses", FOSSaCS '19
  - ▶ CKA : Kappé, B., Silva, Wagemaker, & Zanasi, "Concurrent Kleene Algebra with Observations : from Hypotheses to Completeness", FOSSaCS '20
  - ▶ en général ?
- ▶ Donner une structure à l'alphabet : algèbres empilées (e.g. KAT, CKAO, POCKA, ...)
- ▶ Ajouter des opérateurs : "Fusion" d'algèbres
  - ▶ (e.g. KA + KA commutative = biKA)

## ☞ Techniques :

- ▶ catégories, réécriture, algèbre universelle, théorie des automates, réseaux de Petri, théorie de la démonstration...

# MÉTA-THÉORIE

## 👉 Construction modulaire d'algèbres de programme

- ▶ Quotienter un modèle : algèbres à hypothèses
  - ▶ KA : Doumane, Kuperberg, Pous, & Pradic, "Kleene Algebra with Hypotheses", FOSSaCS '19
  - ▶ CKA : Kappé, B., Silva, Wagemaker, & Zanasi, "Concurrent Kleene Algebra with Observations : from Hypotheses to Completeness", FOSSaCS '20
  - ▶ en général ?
- ▶ Donner une structure à l'alphabet : algèbres empilées (e.g. KAT, CKAO, POCKA, ...)
- ▶ Ajouter des opérateurs : "Fusion" d'algèbres
  - ▶ (e.g. KA + KA commutative = biKA)

## 👉 Techniques :

- ▶ catégories, réécriture, algèbre universelle, théorie des automates, réseaux de Petri, théorie de la démonstration...

## 👉 Formalisation en Coq



# APPLICATION À LA VÉRIFICATION DE PROGRAMMES

☞ Instances du modèle

# APPLICATION À LA VÉRIFICATION DE PROGRAMMES

## ☞ Instances du modèle

- ▶ Langages métiers (DSL)

en s'inspirant de NetKAT (langage basé sur KAT permettant de spécifier et vérifier les SDN)

# APPLICATION À LA VÉRIFICATION DE PROGRAMMES

## ☞ Instances du modèle

- ▶ Langages métiers (DSL)  
en s'inspirant de NetKAT (langage basé sur KAT permettant de spécifier et vérifier les SDN)
- ▶ Algèbre dédiée à un seul type de bugs  
e.g. détection d'interblocage

# APPLICATION À LA VÉRIFICATION DE PROGRAMMES

## ☞ Instances du modèle

- ▶ Langages métiers (DSL)  
en s'inspirant de NetKAT (langage basé sur KAT permettant de spécifier et vérifier les SDN)
- ▶ Algèbre dédiée à un seul type de bugs  
e.g. détection d'interblocage

## ☞ Implémentation d'outils automatiques

# APPLICATION À LA VÉRIFICATION DE PROGRAMMES

## ☞ Instances du modèle

- ▶ Langages métiers (DSL)

  - en s'inspirant de NetKAT (langage basé sur KAT permettant de spécifier et vérifier les SDN)

- ▶ Algèbre dédiée à un seul type de bugs  
e.g. détection d'interblocage

## ☞ Implémentation d'outils automatiques

## ☞ Model-checking : logiques de comportements

# APPLICATION À LA VÉRIFICATION DE PROGRAMMES

## ☞ Instances du modèle

- ▶ Langages métiers (DSL)  
en s'inspirant de NetKAT (langage basé sur KAT permettant de spécifier et vérifier les SDN)
- ▶ Algèbre dédiée à un seul type de bugs  
e.g. détection d'interblocage

## ☞ Implémentation d'outils automatiques

## ☞ Model-checking : logiques de comportements

- ▶ traditionnellement, les logiques de programmes reposent sur une notion d'état (modèles de Kripke & co.)



# APPLICATION À LA VÉRIFICATION DE PROGRAMMES

## ☞ Instances du modèle

- ▶ Langages métiers (DSL)  
en s'inspirant de NetKAT (langage basé sur KAT permettant de spécifier et vérifier les SDN)
- ▶ Algèbre dédiée à un seul type de bugs  
e.g. détection d'interblocage

## ☞ Implémentation d'outils automatiques

## ☞ Model-checking : logiques de comportements

- ▶ traditionnellement, les logiques de programmes reposent sur une notion d'état (modèles de Kripke & co.)
- ▶ Notre « Pomset logic » repose sur une notion de comportement.

# APPLICATION À LA VÉRIFICATION DE PROGRAMMES

## ☞ Instances du modèle

- ▶ Langages métiers (DSL)  
en s'inspirant de NetKAT (langage basé sur KAT permettant de spécifier et vérifier les SDN)
- ▶ Algèbre dédiée à un seul type de bugs  
e.g. détection d'interblocage

## ☞ Implémentation d'outils automatiques

## ☞ Model-checking : logiques de comportements

- ▶ traditionnellement, les logiques de programmes reposent sur une notion d'état (modèles de Kripke & co.)
- ▶ Notre « Pomset logic » repose sur une notion de comportement.

Quelles propriétés des comportements peut-on/souhaite-t-on capturer ?

C'EST TOUT!

Merci de votre attention !

<https://paul.brunet-zamansky.fr>

# AVANCÉES RÉCENTES SUR CKA

## Plan

I. Introduction

II. Algèbre de Kleene concurrente

III. Observations booléennes

IV. Observations partielles

V. Boîtes

VI. Travaux en cours et à venir