On Decidability of Concurrent Kleene Algebra

CONCUR in Berlin – September 5-8, 2017

Paul Brunet, Damien Pous, and Georg Struth

University College London
ENS de Lyon – CNRS
University of Sheffield
Concurrent Kleene Algebra

\[ e, f ::= 0 \mid 1 \mid a \mid e + f \mid e \cdot f \mid e^* \]

- **Kleene algebra**: rational expressions.
  - Can be used to reason about *sequential* programs.
  - Canonical model: regular languages, *i.e.* sets of words.
Concurrent Kleene Algebra

\[ e, f ::= 0 \mid 1 \mid a \mid e + f \mid e \cdot f \mid e^* \mid e \parallel f \]

- **Kleene algebra**: rational expressions.
  - Can be used to reason about **sequential** programs.
  - Canonical model: regular languages, *i.e.* sets of words.

- **bi-Kleene algebra**: series-rational expressions.
  - Can be used to reason about **concurrent** programs.
  - Canonical model: pomset languages *i.e.* sets of **partially ordered** words.

Laurence & Struth, *Completeness theorems for bi-Kleene algebras and series-parallel rational pomset languages*, 2014
Concurrent Kleene Algebra

\[ e, f ::= 0 \mid 1 \mid a \mid e + f \mid e \cdot f \mid e^* \mid e \parallel f \]

- **Kleene algebra**: rational expressions.
  - Can be used to reason about **sequential** programs.
  - Canonical model: regular languages, \textit{i.e.} sets of words.

- **bi-Kleene algebra**: series-rational expressions.
  - Can be used to reason about **concurrent** programs.
  - Canonical model: pomset languages \textit{i.e.} sets of **partially ordered** words.

Laurence & Struth, *Completeness theorems for bi-Kleene algebras and series-parallel rational pomset languages*, 2014

- **Concurrent Kleene algebra**: series-rational expressions.
  - Can be used to reason about **concurrent** programs with a **refinement** order.
  - “Canonical” model: **downwards-closed** pomset languages.

Hoare, Möller, Struth & Wehrman, *Concurrent Kleene algebra and its foundations*, 2011
Concurrent Kleene Algebra

\[ e, f ::= 0 \mid 1 \mid a \mid e + f \mid e \cdot f \mid e^* \mid e \parallel f \]

- **Kleene algebra**: rational expressions.
  - Can be used to reason about **sequential** programs.
  - Canonical model: regular languages, *i.e.* sets of words.
  - Decision procedure: equivalence of **finite state automata**.

- **bi-Kleene algebra**: series-rational expressions.
  - Can be used to reason about **concurrent** programs.
  - Canonical model: pomset languages *i.e.* sets of **partially ordered** words.

Laurence & Struth, *Completeness theorems for bi-Kleene algebras and series-parallel rational pomset languages*, 2014

- **Concurrent Kleene algebra**: series-rational expressions.
  - Can be used to reason about **concurrent** programs with a **refinement** order.
  - “Canonical” model: **downwards-closed** pomset languages.

Hoare, Möller, Struth & Wehrman, *Concurrent Kleene algebra and its foundations*, 2011
Concurrent Kleene Algebra

\[ e, f ::= 0 \mid 1 \mid a \mid e + f \mid e \cdot f \mid e^* \mid e \parallel f \]

- **Kleene algebra**: rational expressions.
  - Can be used to reason about **sequential** programs.
  - Canonical model: regular languages, *i.e.* sets of words.
  - Decision procedure: equivalence of **finite state automata**.

- **bi-Kleene algebra**: series-rational expressions.
  - Can be used to reason about **concurrent** programs.
  - Canonical model: pomset languages *i.e.* sets of **partially ordered** words.
  - Decision procedure: equivalence of **Petri nets**.

Laurence & Struth, *Completeness theorems for bi-Kleene algebras and series-parallel rational pomset languages*, 2014

- **Concurrent Kleene algebra**: series-rational expressions.
  - Can be used to reason about **concurrent** programs with a **refinement** order.
  - “Canonical” model: **downwards-closed** pomset languages.

Hoare, Möller, Struth & Wehrman, *Concurrent Kleene algebra and its foundations*, 2011
Concurrent Kleene Algebra

\[ e, f ::= 0 \mid 1 \mid a \mid e + f \mid e \cdot f \mid e^* \mid e \parallel f \]

- **Kleene algebra**: rational expressions.
  - Can be used to reason about *sequential* programs.
  - Canonical model: regular languages, *i.e.* sets of words.
  - Decision procedure: equivalence of *finite state automata*.
- **bi-Kleene algebra**: series-rational expressions.
  - Can be used to reason about *concurrent* programs.
  - Canonical model: pomset languages *i.e.* sets of *partially ordered* words.
  - Decision procedure: equivalence of *Petri nets*.

Laurence & Struth, *Completeness theorems for bi-Kleene algebras and series-parallel rational pomset languages*, 2014

- **Concurrent Kleene algebra**: series-rational expressions.
  - Can be used to reason about *concurrent* programs with a *refinement* order.
  - “Canonical” model: *downwards-closed* pomset languages.
  - Decision procedure: containment of *Petri nets*.

Hoare, Möller, Struth & Wehrman, *Concurrent Kleene algebra and its foundations*, 2011
Concurrent Kleene Algebra

\[ e, f ::= 0 \mid 1 \mid a \mid e + f \mid e \cdot f \mid e^* \mid e \parallel f \]

- **Kleene algebra**: rational expressions.
  - Can be used to reason about **sequential** programs.
  - Canonical model: regular languages, i.e. sets of words.
  - Decision procedure: equivalence of **finite state automata**.

- **bi-Kleene algebra**: series-rational expressions.
  - Can be used to reason about **concurrent** programs.
  - Canonical model: pomset languages i.e. sets of **partially ordered** words.
  - Decision procedure: equivalence of **Petri nets**.

Laurence & Struth, *Completeness theorems for bi-Kleene algebras and series-parallel rational pomset languages*, 2014

- **Concurrent Kleene algebra**: series-rational expressions.
  - Can be used to reason about **concurrent** programs with a **refinement** order.
  - “Canonical” model: **downwards-closed** pomset languages.
  - Decision procedure: containment of **Petri nets**.

Hoare, Möller, Struth & Wehrman, *Concurrent Kleene algebra and its foundations*, 2011
Outline

I. Pomsets

II. Petri Nets

III. Summary and Outlook
Outline

I. Pomsets

II. Petri Nets

III. Summary and Outlook
Let’s visit London!
Let’s visit London!
Let’s visit London!
Let’s visit London!
Pomsets products

\[ P_1 = \begin{array}{c}
  a \\
  b \\
  c
\end{array} \quad P_2 = \begin{array}{c}
  d \\
  e
\end{array} \]

\[ P_1 \cdot P_2 = \begin{array}{c}
  a \\
  b \\
  c \\
  d \\
  e
\end{array} \quad P_1 \parallel P_2 = \begin{array}{c}
  a \\
  b \\
  c \\
  d \\
  e
\end{array} \]
Pomsets products

\[ P_1 = \begin{array}{c}
    a \\
    c \\
\end{array} \rightarrow \begin{array}{c}
    b
\end{array} \]

\[ P_2 = \begin{array}{c}
    d \\
    e
\end{array} \]

\[ P_1 \cdot P_2 = \begin{array}{c}
    a \rightarrow b \rightarrow d \\
    c \rightarrow e
\end{array} \]
Pomsets products

$P_1 = \begin{array}{c} a \\ b \\ c \end{array}$

$P_2 = \begin{array}{c} d \\ e \end{array}$

$P_1 \cdot P_2 = \begin{array}{c} a \\ b \\ d \\ c \\ e \end{array}$

$P_1 \parallel P_2 = \begin{array}{c} a \\ b \\ c \\ d \\ e \end{array}$
Pomset order

Definition

$P_1 \sqsubseteq P_2$ if there is a function $\varphi$: $P_2 \rightarrow P_1$ such that:

1. $\varphi$ is a bijection
2. $\varphi$ preserves labels
3. $\varphi$ preserves ordered pairs

Gischer, The equational theory of pomsets, 1988

Grabowski, On partial languages, 1981

Notation

$\sqsubseteq_S := \{ P | \exists P' \in S : P \sqsubseteq P' \}$

Brunet, Pous, & Struth, Concurrent Kleene Algebra
Pomset order

Definition

\[ P_1 \sqsubseteq P_2 \] if there is a function \( \varphi : P_2 \rightarrow P_1 \) such that:

1. \( \varphi \) is a bijection
2. \( \varphi \) preserves labels
3. \( \varphi \) preserves ordered pairs

Grabowski, *On partial languages*, 1981
Pomset order

Definition

$P_1 \sqsubseteq P_2$ if there is a function $\varphi : P_2 \rightarrow P_1$ such that:

1. $\varphi$ is a bijection
2. $\varphi$ preserves labels
3. $\varphi$ preserves ordered pairs

Grabowski, *On partial languages*, 1981

Notation

$\sqsubseteq S := \{ P \mid \exists P' \in S : P \sqsubseteq P' \}$. 
Rational pomset languages

e, f ∈ ΠΣ ::= a | 0 | 1 | e · f | e || f | e + f | e*.
Rational pomset languages

\[ e, f \in \mathbb{E}_\Sigma ::= a \mid 0 \mid 1 \mid e \cdot f \mid e \parallel f \mid e + f \mid e^*. \]

\[
\begin{align*}
[a] := & \{ a \} \\
[0] := & \emptyset \\
[e \cdot f] := & [e] \cdot [f] \\
[e^*] := & \bigcup_{n \in \mathbb{N}} [e]^n \\
[1] := & \{ \square \} \\
[e + f] := & [e] \cup [f] \\
[e \parallel f] := & [e] \parallel [f]
\end{align*}
\]
Rational pomset languages

\[ e, f \in \mathbb{E}_\Sigma ::= a \mid 0 \mid 1 \mid e \cdot f \mid e \parallel f \mid e + f \mid e^* . \]

\[
\begin{align*}
[a] &:= \{ a \} \\
[0] &:= \emptyset \\
[e \cdot f] &:= [e] \cdot [f] \\
[e^*] &:= \bigcup_{n \in \mathbb{N}} [e]^n \\
[1] &:= \{ \} \\
[e + f] &:= [e] \cup [f] \\
[e \parallel f] &:= [e] \parallel [f]
\end{align*}
\]

**Definition**

A set of pomsets \( S \) is called a **rational pomset language** if there is an expression \( e \in \mathbb{E}_\Sigma \) such that \( S = [e] \).
Two decision problems

biKA
Given two expressions $e, f$, are $[e]$ and $[f]$ equal?

CKA
Given two expressions $e, f$, are $\downarrow[e]$ and $\downarrow[f]$ equal?
Outline

I. Pomsets

II. Petri Nets

III. Summary and Outlook
Labelled Petri nets

Brunet, Pous, & Struth
Concurrent Kleene Algebra
Labelled Petri nets
Labelled Petri nets
Labelled Petri nets
Labelled Petri nets
Labelled Petri nets

Petri Nets

Brunet, Pous, & Struth
Concurrent Kleene Algebra
11/23
Labelled Petri nets
Labelled Petri nets
Labelled Petri nets

\[ \tau \rightarrow a \rightarrow b \rightarrow c \rightarrow \tau \]

Brunet, Pous, & Struth

Concurrent Kleene Algebra
Labelled Petri nets
Labelled Petri nets
Labelled Petri nets

\[ \tau \]

Brunet, Pous, & Struth
Concurrent Kleene Algebra
Labelled Petri nets
Labelled Petri nets

Brunet, Pous, & Struth
Labelled Petri nets

Brunet, Pous, & Struth
Concurrent Kleene Algebra
11/23
Labelled Petri nets

Transition-pomset
Labelled Petri nets

Brunet, Pous, & Struth

Concurrent Kleene Algebra

11/23
Recognisable pomset languages

Language generated by a net

\([\mathcal{N}]\) is the set of pomset-traces of accepting runs of \(\mathcal{N}\).

Definition

A set of pomsets \(S\) is a **recognisable pomset language** if there is a net \(\mathcal{N}\) such that \(S = [\mathcal{N}]\).
From expressions to automata

\[ N(0) := \rightarrow \quad N(1) := \rightarrow \quad N(a) := \rightarrow a \]

\[ N(e_1 + e_2) := \]

\[ N(e_1 \parallel e_2) := \]

\[ N(e_1 \cdot e_2) := \]

\[ N(e^*) := \]
Solving biKA

**Lemma**

\[ [e] = [\mathcal{N}(e)]. \]

**Corollary**

Rational pomset languages are recognisable.
Solving biKA

Lemma

\[ [e] = [N(e)] . \]

Corollary

Rational pomset languages are recognisable.

Theorem

Testing containment of pomset-trace languages of two Petri nets is an \textit{ExpSpace}-complete problem.

Jategaonkar & Meyer, \textit{Deciding true concurrency equivalences on safe, finite nets}, 1996

Corollary

The problem biKA lies in the class \textit{ExpSpace}.
What about CKA?

\[ \llbracket e \rrbracket = \llbracket f \rrbracket \]
What about CKA?

\[ \llbracket e \rrbracket = \llbracket f \rrbracket \iff \llbracket e \rrbracket \subseteq \llbracket f \rrbracket \land \llbracket e \rrbracket \supseteq \llbracket f \rrbracket \]
What about CKA?

\[
\preceq [e] = \preceq [f] \iff \preceq [e] \subseteq \preceq [f] \land \preceq [e] \supseteq \preceq [f]
\]

\[
\iff [e] \subseteq \preceq [f] \land [e] \supseteq \preceq [f]
\]
What about CKA?

\[ [e] = [f] \iff [e] \subseteq [f] \land [e] \supseteq [f] \]
\[ \iff [e] \subseteq [f] \land [e] \supseteq [f] \]
\[ \iff [\mathcal{N}(e)] \subseteq [\mathcal{N}(f)] \land [\mathcal{N}(e)] \supseteq [\mathcal{N}(f)] \]
What about CKA?

\[ \llbracket e \rrbracket = \llbracket f \rrbracket \iff \llbracket e \rrbracket \subseteq \llbracket f \rrbracket \land \llbracket e \rrbracket \supseteq \llbracket f \rrbracket \]
\[ \iff [e] \subseteq [f] \land [e] \supseteq [f] \]
\[ \iff [\mathcal{N}(e)] \subseteq [\mathcal{N}(f)] \land [\mathcal{N}(e)] \supseteq [\mathcal{N}(f)] \]

Problem

Let \( \mathcal{N}_1, \mathcal{N}_2 \) be well behaved nets. Is it true that for every run \( R_1 \) of \( \mathcal{N}_1 \) there is a run \( R_2 \) in \( \mathcal{N}_2 \) such that

\[ \mathcal{P}om(R_1) \subseteq \mathcal{P}om(R_2)? \]
Idea of the algorithm

Problem
Let $\mathcal{N}_1, \mathcal{N}_2$ be well behaved nets. Is it true that for every run $R_1$ of $\mathcal{N}_1$ there is a run $R_2$ in $\mathcal{N}_2$ such that

$$\text{Pom}(R_1) \sqsubseteq \text{Pom}(R_2)?$$
Idea of the algorithm

Problem

Let $\mathcal{N}_1, \mathcal{N}_2$ be well behaved nets. Is it true that for every run $R_1$ of $\mathcal{N}_1$ there is a run $R_2$ in $\mathcal{N}_2$ such that

$$\mathcal{Pom}(R_1) \sqsubseteq \mathcal{Pom}(R_2)?$$

▶ build an automaton $\mathcal{A}_1$ for $\llbracket \mathcal{N}_1 \rrbracket$
Idea of the algorithm

Problem

Let $\mathcal{N}_1, \mathcal{N}_2$ be well behaved nets. Is it true that for every run $R_1$ of $\mathcal{N}_1$ there is a run $R_2$ in $\mathcal{N}_2$ such that

$$\mathcal{Pom}(R_1) \sqsubseteq \mathcal{Pom}(R_2)?$$

- build an automaton $\mathcal{A}_1$ for $[\mathcal{N}_1]$
- build an automaton $\mathcal{A}_2$ for $[\mathcal{N}_1] \cap \mathcal{F}[\mathcal{N}_2]$
Idea of the algorithm

Problem
Let $\mathcal{N}_1, \mathcal{N}_2$ be well behaved nets. Is it true that for every run $R_1$ of $\mathcal{N}_1$ there is a run $R_2$ in $\mathcal{N}_2$ such that

$$\mathcal{Pom}(R_1) \sqsubseteq \mathcal{Pom}(R_2)?$$

- build an automaton $A_1$ for $\downarrow \mathcal{N}_1$
- build an automaton $A_2$ for $\downarrow \mathcal{N}_1 \cap \sqsubseteq \downarrow \mathcal{N}_2$
- $\mathcal{N}_1 \subseteq \sqsubseteq \downarrow \mathcal{N}_2$ if and only if $\mathcal{L}(A_1) = \mathcal{L}(A_2)$. 

Brunet, Pous, & Struth

Concurrent Kleene Algebra
Transition automaton

The diagram above represents a transition automaton with labeled transitions. The states are A, B, C, D, E, F, and G. The transitions are labeled as $t_1$, $t_2$, $t_a$, $t_b$, $t_c$, and $t_d$. The automaton starts at state A and moves through states B, C, D, E, F, and G following the specified transitions.
Transition automaton

Brunet, Pous, & Struth
Concurrent Kleene Algebra
17/23
Massaging runs

Petri Nets

Brunet, Pous, & Struth

Concurrent Kleene Algebra
Massaging runs

Petri Nets

Brunet, Pous, & Struth
Concurrent Kleene Algebra
18/23
Massaging runs

Petri Nets

Brunet, Pous, & Struth
Concurrent Kleene Algebra

18/23
Massaging runs
Massaging runs
Massaging runs

Petri Nets

Brunet, Pous, & Struth
Reduction to automata

Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be some polite nets, of size $n, m$.

**Lemma**

There is an automaton $\mathcal{A}(\mathcal{N}_1)$ with $O(2^n)$ states that recognises the set of accepting runs in $\mathcal{N}_1$. 

**Lemma**

There is an automaton $\mathcal{N}_1 \preceq \mathcal{N}_2$ with $O(2^n + m + nm)$ states that recognises the set of accepting runs in $\mathcal{N}_1$ whose pomset belongs to $\sqsubseteq_{\mathcal{J}} \mathcal{N}_2$. 

Brunet, Pous, & Struth  Concurrent Kleene Algebra  19/23
Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be some polite nets, of size $n, m$.

**Lemma**

There is an automaton $\mathcal{A}(\mathcal{N}_1)$ with $O(2^n)$ states that recognises the set of accepting runs in $\mathcal{N}_1$.

**Lemma**

There is an automaton $\mathcal{N}_1 \prec \mathcal{N}_2$ with $O(2^{n+m+nm})$ states that recognises the set of accepting runs in $\mathcal{N}_1$ whose pomset belongs to $\sqsubseteq[[\mathcal{N}_2]]$. 
Outline

I. Pomsets

II. Petri Nets

III. Summary and Outlook
Main result

Theorem

Given two expressions $e, f \in \mathbb{E}_\Sigma$, we can test if $[e] \subseteq [f]$ in ExpSpace.

Proof.

1. build $N(e)$ and $N(f)$;
2. build $A(N(e))$ and $N(e) \prec N(f)$;
3. compare them.
Main result

Theorem

Given two expressions $e, f \in E_\Sigma$, we can test if $[e] \subseteq [f]$ in $\text{ExpSpace}$.

Proof.
1. build $\mathcal{N}(e)$ and $\mathcal{N}(f)$;
2. build $\mathcal{A}(\mathcal{N}(e))$ and $\mathcal{N}(e) \prec \mathcal{N}(f)$;
3. compare them.

\[\square\]

Theorem

The problem $\text{CKA}$ is $\text{ExpSpace}$-complete.

Proof.
1. In the class $\text{ExpSpace}$: see above.
2. $\text{ExpSpace}$-hard: Reduction from the universality problem for regular expressions with interleaving.

Mayer & Stockmeyer, The complexity of word problems – this time with interleaving, 1994
To sum up

Done:

- Reduction of biKA and CKA to Petri nets.
- New automaton-like semantics for Petri nets.
- biKA is ExpSpace-solvable.
- CKA is ExpSpace-complete.

To do:

- Extend the algorithm to a larger class of Petri nets.
- Add tests because they're useful!
- Add names because they're fun!
- Insert your favourite operator here...
To sum up

**Done:**
- Reduction of biKA and CKA to *Petri nets.*

**To do:**
- Extend the algorithm to a larger class of Petri nets.
- Add tests because they’re useful!
- Add names because they’re fun!
- Insert your favourite operator here...
To sum up

Done:
- Reduction of biKA and CKA to Petri nets.
- New automaton-like semantics for Petri nets.

To do:
- Extend the algorithm to a larger class of Petri nets.
- Add tests because they’re useful!
- Add names because they’re fun!
- Insert your favourite operator here...
**To sum up**

**Done:**
- Reduction of biKA and CKA to Petri nets.
- New automaton-like semantics for Petri nets.
- biKA is ExpSpace-solvable.

**To do:**
To sum up

Done:

- Reduction of biKA and CKA to **Petri nets**.
- New automaton-like **semantics** for Petri nets.
- biKA is **ExpSpace-solvable**.
- CKA is **ExpSpace-complete**.

To do:

- Extend the algorithm to a larger class of Petri nets.
- Add tests because they're useful!
- Add names because they're fun!
To sum up

**Done:**
- Reduction of biKA and CKA to **Petri nets**.
- New automaton-like **semantics** for Petri nets.
- biKA is **ExpSpace-solvable**.
- CKA is **ExpSpace-complete**.

**To do:**
- Extend the algorithm to a larger class of Petri nets.
To sum up

Done:

- Reduction of biKA and CKA to Petri nets.
- New automaton-like semantics for Petri nets.
- biKA is ExpSpace-solvable.
- CKA is ExpSpace-complete.

To do:

- Extend the algorithm to a larger class of Petri nets.
- Add tests because they’re useful!
To sum up

**Done:**
- Reduction of biKA and CKA to **Petri nets**.
- New automaton-like **semantics** for Petri nets.
- biKA is **ExpSpace-solvable**.
- CKA is **ExpSpace-complete**.

**To do:**
- Extend the algorithm to a larger class of Petri nets.
- Add **tests** because they’re useful!
- Add **names** because they’re fun!
To sum up

**Done:**
- Reduction of biKA and CKA to *Petri nets*.
- New automaton-like *semantics* for Petri nets.
- biKA is *ExpSpace-solvable*.
- CKA is *ExpSpace-complete*.

**To do:**
- Extend the algorithm to a larger class of Petri nets.
- Add *tests* because they’re useful!
- Add *names* because they’re fun!
- *Insert you favourite operator here*...
That’s all folks!

Thank you!

See more at:
Outline

I. Pomsets

II. Petri Nets

III. Summary and Outlook