Concurrent Kleene Algebra: Free Model and Completeness

Tobias Kappé Paul Brunet Alexandra Silva Fabio Zanasi

University College London

ESOP 2018

Introduction

Let's write a program that outputs $n > 0$ space-separated ☺'s.

Introduction

Let's write a program that outputs $n > 0$ space-separated ☺'s.

$i := 1$
while $i < n$ **do**
    print ☺
    print ␣
    $i := i + 1$
end
print ☺

Introduction

Let's write a program that outputs $n > 0$ space-separated ☺'s.

```
i := 1
while i < n do
    print ☺
    print ␣
    i := i + 1
end
print ☺
```

```
i := 1
print ☺
while i < n do
    print ␣
    print ☺
    i := i + 1
end
```

Introduction

Let's write a program that outputs $n > 0$ space-separated ☺'s.

```
i := 1
while i < n do
    print ☺
    print ␣
    i := i + 1
end
print ☺
```

```
i := 1
print ☺
while i < n do
    print ␣
    print ☺
    i := i + 1
end
```

Are these programs equivalent?

Introduction

Programs are *expressions*

Introduction

Programs are *expressions*, thus we should be able to reason *equationally*.

Introduction

Programs are *expressions*, thus we should be able to reason *equationally*.

Kleene Algebra (KA) provides an *algebraic framework* to do this.

Introduction

program	expression
atomic action	$a, b, \ldots \in \Sigma$

Introduction

program	expression
atomic action	$a, b, \ldots \in \Sigma$
abort execution	0

Introduction

program	expression
atomic action	$a, b, \ldots \in \Sigma$
abort execution	0
no-operation	1

Introduction

program	expression
atomic action	$a, b, \ldots \in \Sigma$
abort execution	0
no-operation	1
nondeterministic choice	$e + f$

Introduction

program	expression
atomic action	$a, b, \ldots \in \Sigma$
abort execution	0
no-operation	1
nondeterministic choice	$e + f$
sequential composition	$e \cdot f$

Introduction

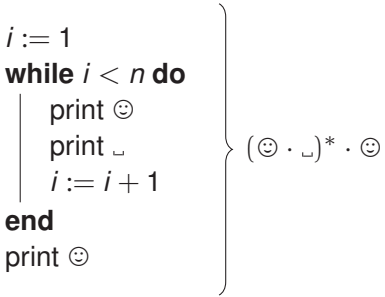
program	expression
atomic action	$a, b, \ldots \in \Sigma$
abort execution	0
no-operation	1
nondeterministic choice	$e + f$
sequential composition	$e \cdot f$
repetition	e^*

Introduction

```
i := 1
while i < n do
    print ☺
    print ␣
    i := i + 1
end
print ☺
```

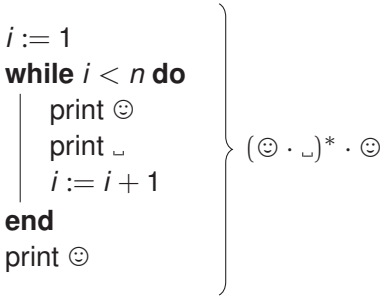
```
i := 1
print ☺
while i < n do
    print ␣
    print ☺
    i := i + 1
end
```

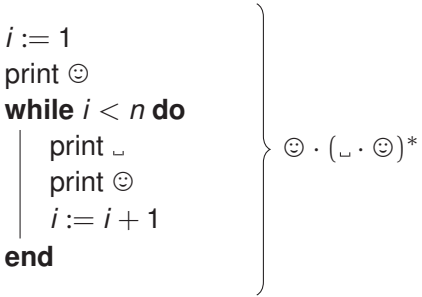
Introduction



$i := 1$
print ☺
while $i < n$ **do**
print ␣
print ☺
$i := i + 1$
end

Introduction





Introduction

Axioms of KA:

$$e + 0 \equiv e \qquad e + e \equiv e \qquad e + f \equiv f + e \qquad e + (f + g) \equiv (e + f) + g$$

$$e \cdot 0 \equiv 0 \equiv 0 \cdot e \qquad e \cdot 1 \equiv e \equiv 1 \cdot e \qquad e \cdot (f \cdot g) \equiv (e \cdot f) \cdot g$$

$$e \cdot (f + g) \equiv e \cdot f + e \cdot g \qquad (e + f) \cdot g \equiv e \cdot g + f \cdot g$$

$$1 + e \cdot e^* \equiv e^* \qquad e \cdot f + g \leqq f \implies e^* \cdot g \leqq f$$

Introduction

Axioms of KA:

$$e + 0 \equiv e \qquad e + e \equiv e \qquad e + f \equiv f + e \qquad e + (f + g) \equiv (e + f) + g$$

$$e \cdot 0 \equiv 0 \equiv 0 \cdot e \qquad e \cdot 1 \equiv e \equiv 1 \cdot e \qquad e \cdot (f \cdot g) \equiv (e \cdot f) \cdot g$$

$$e \cdot (f + g) \equiv e \cdot f + e \cdot g \qquad (e + f) \cdot g \equiv e \cdot g + f \cdot g$$

$$1 + e \cdot e^* \equiv e^* \qquad e \cdot f + g \leqq f \implies e^* \cdot g \leqq f$$

Introduction

$$☺ \cdot (␣ \cdot ☺)^* \equiv (☺ \cdot ␣)^* \cdot ☺$$

Introduction

Theorem (Kozen 1990)

The axioms for KA *are sound &* ***complete*** *for equivalence:*

$$e \equiv f \iff \mathcal{L}(e) = \mathcal{L}(f)$$

$\mathcal{L}(e)$ *is the regular language interpretation of* e*.*

Introduction

Theorem (Kozen 1990)

The axioms for KA *are sound &* ***complete*** *for equivalence:*

$$e \equiv f \iff \mathcal{L}(e) = \mathcal{L}(f)$$

$\mathcal{L}(e)$ *is the regular language interpretation of* e*.*

Upshot:

- to check KA equivalence is to check regular language equivalence
- through Kleene's theorem, this means checking DFA equivalence
- sophisticated (near-linear) algorithms exist to do this

Adding concurrency

Which *new* axioms do we need for parallel composition?

Adding concurrency

Which *new* axioms do we need for parallel composition?

$$e \parallel f \equiv f \parallel e$$

Adding concurrency

Which *new* axioms do we need for parallel composition?

$$e \parallel f \equiv f \parallel e \qquad\qquad e \parallel (f \parallel g) \equiv (e \parallel f) \parallel g$$

Adding concurrency

Which *new* axioms do we need for parallel composition?

$$e \parallel f \equiv f \parallel e \qquad e \parallel (f \parallel g) \equiv (e \parallel f) \parallel g$$

$$e \parallel 1 \equiv e$$

Adding concurrency

Which *new* axioms do we need for parallel composition?

$$e \parallel f \equiv f \parallel e \qquad\qquad e \parallel (f \parallel g) \equiv (e \parallel f) \parallel g$$

$$e \parallel 1 \equiv e \qquad\qquad e \parallel 0 \equiv 0$$

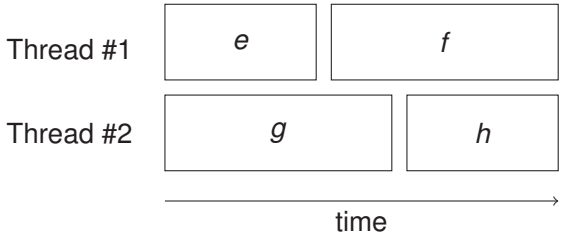
Adding concurrency

Which *new* axioms do we need for parallel composition?

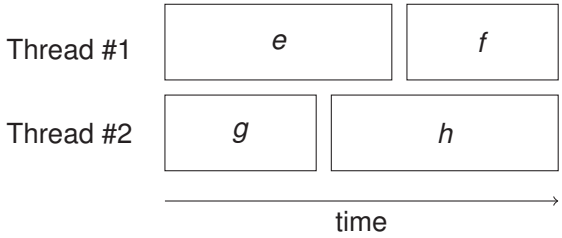
$$e \parallel f \equiv f \parallel e \qquad e \parallel (f \parallel g) \equiv (e \parallel f) \parallel g$$

$$e \parallel 1 \equiv e \qquad e \parallel 0 \equiv 0 \qquad e \parallel (f + g) \equiv e \parallel f + e \parallel g$$

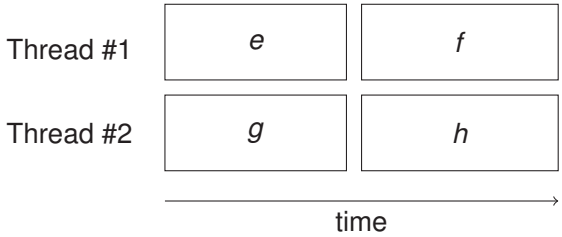
Adding concurrency



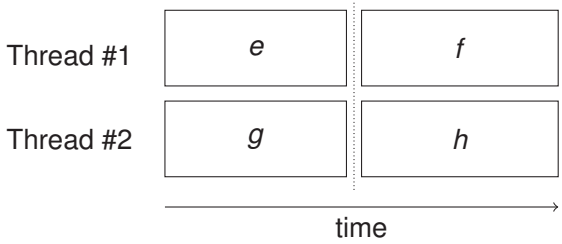
Adding concurrency



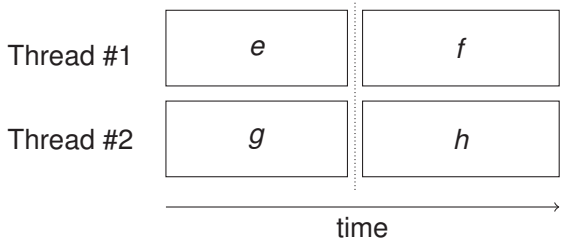
Adding concurrency



Adding concurrency

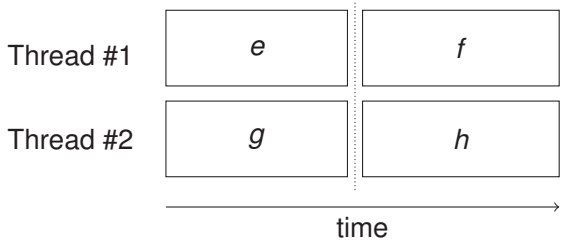


Adding concurrency



Equationally: $(e \parallel g) \cdot (f \parallel h) \leqq (e \cdot f) \parallel (g \cdot h)$.

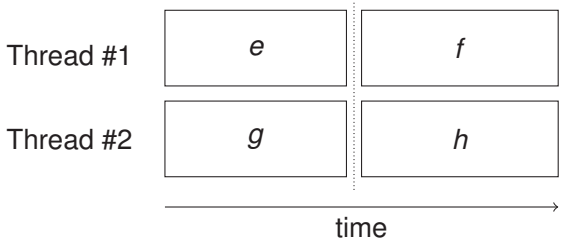
Adding concurrency



Equationally: $(e \parallel g) \cdot (f \parallel h) \leqq (e \cdot f) \parallel (g \cdot h)$.

$$p \leqq q \iff p + q \equiv q$$

Adding concurrency



Equationally: $(e \parallel g) \cdot (f \parallel h) \leqq (e \cdot f) \parallel (g \cdot h)$.

Nondeterministic interleaving as special case: $e \cdot f + f \cdot e \leqq e \parallel f$.

Adding concurrency

Question

Can we have a regular interpretation $[\![-]\!]$ *such that* $e \equiv f \iff [\![e]\!] = [\![f]\!]$*?*

Adding concurrency

Question

Can we have a regular interpretation $[\![-]\!]$ *such that* $e \equiv f \iff [\![e]\!] = [\![f]\!]$*?*

NB: $[\![-]\!]$ should generalize $\mathcal{L}(-)$: for $\|$-less terms, $\mathcal{L}(e)$ should resemble $[\![e]\!]$.

Regular interpretation: first attempt

Partially ordered multiset (pomset):

$$a \cdot b \cong a \longrightarrow b$$

Regular interpretation: first attempt

Partially ordered multiset (pomset):

$$a \cdot b \cong a \longrightarrow b \qquad\qquad a \parallel b \cong \begin{array}{c} b \\ \\ a \end{array}$$

Regular interpretation: first attempt

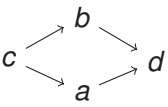
Partially ordered multiset (pomset):

$$a \cdot b \cong a \longrightarrow b \qquad\qquad c \cdot (a \parallel b) \cong c \begin{array}{l} \nearrow b \\ \searrow a \end{array}$$

Regular interpretation: first attempt

Partially ordered multiset (pomset):

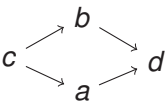
$$a \cdot b \cong a \longrightarrow b \qquad\qquad c \cdot (a \parallel b) \cdot d \cong$$



Regular interpretation: first attempt

Partially ordered multiset (pomset):

$$a \cdot b \cong a \longrightarrow b \qquad\qquad c \cdot (a \parallel b) \cdot d \cong$$



Composition lifts to *sets of pomsets* in the obvious way.

Regular interpretation: first attempt

Straightforward semantics: $(\!|-|\!) : \mathcal{T} \to 2^{\text{Pomsets}}$ given by

$$(\!|0|\!) = \emptyset \qquad (\!|e + f|\!) = (\!|e|\!) \cup (\!|f|\!) \qquad (\!|e^*|\!) = (\!|e|\!)^*$$

$$(\!|1|\!) = \{1\} \qquad (\!|e \cdot f|\!) = (\!|e|\!) \cdot (\!|f|\!)$$

$$(\!|a|\!) = \{a\} \qquad (\!|e \parallel f|\!) = (\!|e|\!) \parallel (\!|f|\!)$$

Regular interpretation: first attempt

Straightforward semantics: $(\!|-|\!) : \mathcal{T} \to 2^{\text{Pomsets}}$ given by

$$
\begin{array}{lll}
(\!|0|\!) = \emptyset & (\!|e + f|\!) = (\!|e|\!) \cup (\!|f|\!) & (\!|e^*|\!) = (\!|e|\!)^* \\
(\!|1|\!) = \{1\} & (\!|e \cdot f|\!) = (\!|e|\!) \cdot (\!|f|\!) & \\
(\!|a|\!) = \{a\} & (\!|e \parallel f|\!) = (\!|e|\!) \parallel (\!|f|\!) &
\end{array}
$$

Problem: $(\!|-|\!)$ is not sound for the exchange law.

Regular interpretation: first attempt

Straightforward semantics: $(\!|-|\!) : \mathcal{T} \to 2^{\text{Pomsets}}$ given by

$$(\!|0|\!) = \emptyset \qquad (\!|e + f|\!) = (\!|e|\!) \cup (\!|f|\!) \qquad (\!|e^*|\!) = (\!|e|\!)^*$$
$$(\!|1|\!) = \{1\} \qquad (\!|e \cdot f|\!) = (\!|e|\!) \cdot (\!|f|\!)$$
$$(\!|a|\!) = \{a\} \qquad (\!|e \parallel f|\!) = (\!|e|\!) \parallel (\!|f|\!)$$

Problem: $(\!|-|\!)$ is not sound for the exchange law.

For instance: $a \cdot b \leqq a \parallel b$ should imply that $(\!|a \cdot b|\!) \subseteq (\!|a \parallel b|\!)$, but

$$(\!|a \cdot b|\!) = \left\{ \; a \longrightarrow b \; \right\} \qquad (\!|a \parallel b|\!) = \left\{ \; a \qquad b \; \right\}$$

Regular interpretation: first attempt

Axioms to build $\approx$ are axioms for $\equiv$, minus exchange law.

Regular interpretation: first attempt

Axioms to build $\approx$ are axioms for $\equiv$, minus exchange law.

Theorem (Laurence and Struth 2014)

The axioms for $\approx$ are sound & complete w.r.t. $(\!|-|\!)$:

$$e \approx f \iff (\!|e|\!) = (\!|f|\!)$$

Regular interpretation: second attempt

We define the *subsumption order* $\sqsubseteq$ on pomsets.

Intuition: $U \sqsubseteq V$ if

i. U and V have the same events, and
ii. U has all order in V (and possibly more)

Regular interpretation: second attempt

We define the *subsumption order* $\sqsubseteq$ on pomsets.

Intuition: $U \sqsubseteq V$ if

i. U and V have the same events, and
ii. U has all order in V (and possibly more)

For example:

$$a \longrightarrow b \sqsubseteq a \quad b$$

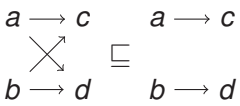
Regular interpretation: second attempt

We define the *subsumption order* $\sqsubseteq$ on pomsets.

Intuition: $U \sqsubseteq V$ if

i. U and V have the same events, and
ii. U has all order in V (and possibly more)

For example:



Regular interpretation: second attempt

"Fixed" semantics: $[\![e]\!] = (\!|\, e \,|\!)\downarrow$.

downward closure w.r.t. $\sqsubseteq$

Regular interpretation: second attempt

"Fixed" semantics: $[\![e]\!] = (\!|\, e \,|\!)\downarrow$.

Previous problem no longer occurs:

$$[\![a \cdot b]\!] = \left\{ \; a \to b \; \right\} \subseteq \left\{ \; a \to b \;,\; a \leftarrow b \;,\; a \quad b \; \right\} = [\![a \parallel b]\!]$$

Regular interpretation: second attempt

"Fixed" semantics: $[\![e]\!] = (\!|\, e \,|\!)\downarrow$.

Previous problem no longer occurs:

$$[\![a \cdot b]\!] = \left\{ \; a \to b \; \right\} \subseteq \left\{ \; a \to b \;,\; a \leftarrow b \;,\; a \quad b \; \right\} = [\![a \parallel b]\!]$$

Lemma (Hoare et al. 2009)

The axioms for $\equiv$ are sound w.r.t. $[\![-]\!]$, i.e., $e \equiv f$ implies $[\![e]\!] = [\![f]\!]$.

Closure

Definition

Let $e \in \mathcal{T}$; a *closure* of e is a term $e{\downarrow}$ such that

1. $e{\downarrow} \equiv e$
2. $[\![e]\!] = (\!| e{\downarrow} |\!)$

Closure

Definition

Let $e \in \mathcal{T}$; a *closure* of e is a term $e{\downarrow}$ such that

1. $e{\downarrow} \equiv e$
2. $[\![e]\!] = (\!| e{\downarrow} |\!)$

Lemma (Laurence and Struth 2017)

If closures exist for all terms, then $\equiv$ is complete w.r.t. $[\![-]\!]$, i.e., $[\![e]\!] = [\![f]\!]$ implies $e \equiv f$.

Closure

Definition

Let $e \in \mathcal{T}$; a *closure* of e is a term $e{\downarrow}$ such that

1. $e{\downarrow} \equiv e$
2. $[\![e]\!] = (\!| e{\downarrow} |\!)$

Lemma (Laurence and Struth 2017)

If closures exist for all terms, then $\equiv$ is complete w.r.t. $[\![-]\!]$, i.e., $[\![e]\!] = [\![f]\!]$ implies $e \equiv f$.

Proof.

If $[\![e]\!] = [\![f]\!]$, □

Closure

Definition

Let $e \in \mathcal{T}$; a *closure* of e is a term $e{\downarrow}$ such that

1. $e{\downarrow} \equiv e$
2. $[\![e]\!] = (\!| e{\downarrow} |\!)$

Lemma (Laurence and Struth 2017)

If closures exist for all terms, then $\equiv$ *is complete w.r.t.* $[\![-]\!]$*, i.e.,* $[\![e]\!] = [\![f]\!]$ *implies* $e \equiv f$*.*

Proof.

If $[\![e]\!] = [\![f]\!]$, then $(\!| e{\downarrow} |\!) = (\!| f{\downarrow} |\!)$, □

Closure

Definition

Let $e \in \mathcal{T}$; a *closure* of e is a term $e{\downarrow}$ such that

1. $e{\downarrow} \equiv e$
2. $[\![e]\!] = (\!| e{\downarrow} |\!)$

Lemma (Laurence and Struth 2017)

If closures exist for all terms, then $\equiv$ is complete w.r.t. $[\![-]\!]$, i.e., $[\![e]\!] = [\![f]\!]$ implies $e \equiv f$.

Proof.

If $[\![e]\!] = [\![f]\!]$, then $(\!| e{\downarrow} |\!) = (\!| f{\downarrow} |\!)$, thus $e{\downarrow} \approx f{\downarrow}$. □

Closure

Definition

Let $e \in \mathcal{T}$; a *closure* of e is a term $e{\downarrow}$ such that

1. $e{\downarrow} \equiv e$
2. $[\![e]\!] = (\!| e{\downarrow} |\!)$

Lemma (Laurence and Struth 2017)

If closures exist for all terms, then $\equiv$ is complete w.r.t. $[\![-]\!]$, i.e., $[\![e]\!] = [\![f]\!]$ implies $e \equiv f$.

Proof.

If $[\![e]\!] = [\![f]\!]$, then $(\!| e{\downarrow} |\!) = (\!| f{\downarrow} |\!)$, thus $e{\downarrow} \approx f{\downarrow}$. Therefore, $e{\downarrow} \equiv f{\downarrow}$. □

Closure

Definition

Let $e \in \mathcal{T}$; a *closure* of e is a term $e{\downarrow}$ such that

1. $e{\downarrow} \equiv e$
2. $[\![e]\!] = (\!| e{\downarrow} |\!)$

Lemma (Laurence and Struth 2017)

If closures exist for all terms, then $\equiv$ is complete w.r.t. $[\![-]\!]$, i.e., $[\![e]\!] = [\![f]\!]$ implies $e \equiv f$.

Proof.

If $[\![e]\!] = [\![f]\!]$, then $(\!| e{\downarrow} |\!) = (\!| f{\downarrow} |\!)$, thus $e{\downarrow} \approx f{\downarrow}$. Therefore, $e \equiv e{\downarrow} \equiv f{\downarrow} \equiv f$. □

Main contribution

Theorem

*If $e \in \mathcal{T}$, then we can **compute** a term $e{\downarrow}$ that is a closure of e.*

Main contribution

Theorem

*If $e \in \mathcal{T}$, then we can **compute** a term $e{\downarrow}$ that is a closure of e.*

Corollary

The axioms for CKA *are sound & complete w.r.t. $[\![-]\!]$:*

$$e \equiv f \iff [\![e]\!] = [\![f]\!]$$

Main contribution

Theorem

If $e \in \mathcal{T}$, then we can ***compute*** *a term $e{\downarrow}$ that is a closure of e.*

Corollary

The axioms for CKA *are sound & complete w.r.t. $[\![-]\!]$:*

$$e \equiv f \iff [\![e]\!] = [\![f]\!]$$

The latter can be decided; c.f. [Brunet, Pous, and Struth 2017].

Further work

- Explore coalgebraic perspective:
  - Efficient equivalence checking through bisimulation?
  - Can completeness be shown coalgebraically?
- Add “parallel star” operator — closure method does not apply.
- Extend *Kleene Algebra with Tests* (KAT) to add concurrency.
- Extend extend NetKAT with concurrency.

Thank you for your attention

Implementation: https://doi.org/10.5281/zenodo.926651.

Extended paper: https://arxiv.org/abs/1710.02787.