Concurrent NetKAT

Jana Wagemaker, Nate Foster, Tobias Kappé
Dexter Kozen, Jurriaan Rot, Alexandra Silva
Radboud Universiteit, Cornell University, ILLC, University of Amsterdam

Overview

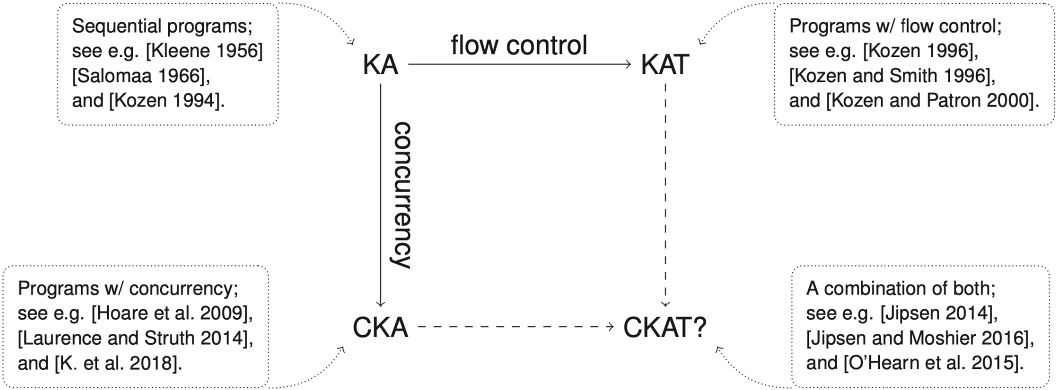
- Kleene algebra: axiomatisation of regular languages, used to reason about simple programs

- Kleene algebra: axiomatisation of regular languages, used to reason about simple programs
- Many variants and extensions that enable analysis of more complicated programs:
  - Kleene algebra with tests (KAT)

Overview

- Kleene algebra: axiomatisation of regular languages, used to reason about simple programs
- Many variants and extensions that enable analysis of more complicated programs:
  - Kleene algebra with tests (KAT)
  - NetKAT
  - ...

- Kleene algebra: axiomatisation of regular languages, used to reason about simple programs
- Many variants and extensions that enable analysis of more complicated programs:
  - Kleene algebra with tests (KAT)
  - NetKAT
  - ...
- Key limitation of NetKAT: stateless and sequential


Sequential programs;
see e.g. [Kleene 1956]
[Salomaa 1966],
and [Kozen 1994].
KA
flow control
KAT
Programs w/ flow control;
see e.g. [Kozen 1996],
[Kozen and Smith 1996],
and [Kozen and Patron 2000].
concurrency
Programs w/ concurrency;
see e.g. [Hoare et al. 2009],
[Laurence and Struth 2014],
and [K. et al. 2018].
CKA
CKAT?
A combination of both;
see e.g. [Jipsen 2014],
[Jipsen and Moshier 2016],
and [O'Hearn et al. 2015].

$$p \wedge q = p \,;\, q \rightsquigarrow p \wedge q \leq p \,;\, q$$

$$\begin{tikzcd}
\mathrm{KA} \arrow[r, "\text{observations}"] \arrow[d, "\text{concurrency}"] & \mathrm{KAO} \arrow[d, dashed] \\
\mathrm{CKA} \arrow[r, dashed] & \mathrm{CKAO}
\end{tikzcd}$$

$$p \wedge q = p \,;\, q \rightsquigarrow p \wedge q \leq p \,;\, q$$

$$\begin{tikzcd}
\mathrm{KA} \arrow[r, "\text{observations}"] \arrow[d, "\text{concurrency}"] & \mathrm{KAO} \arrow[d, dashed] \\
\mathrm{CKA} \arrow[r, dashed] & \mathrm{CKAO}
\end{tikzcd}$$

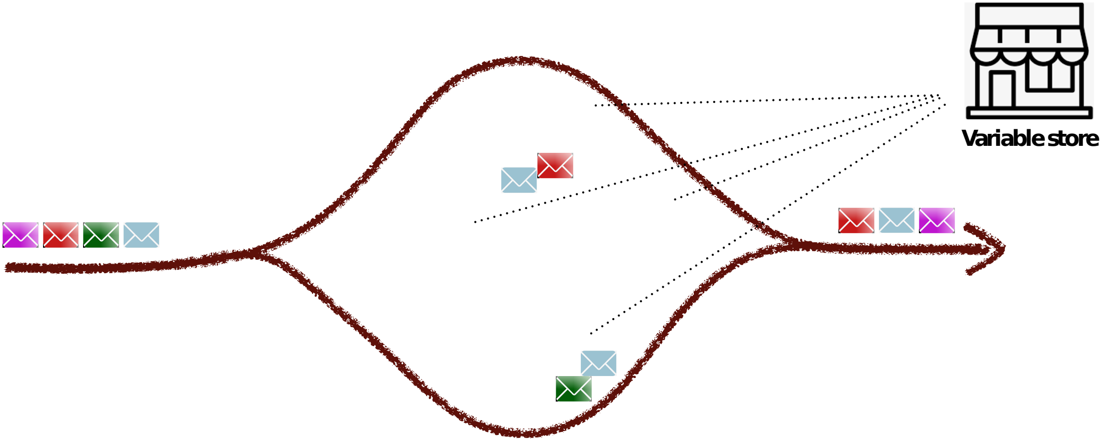
$\Rightarrow$ POCKA: fine-grained reasoning about concurrent programs with conditionals that manipulate a shared global memory.

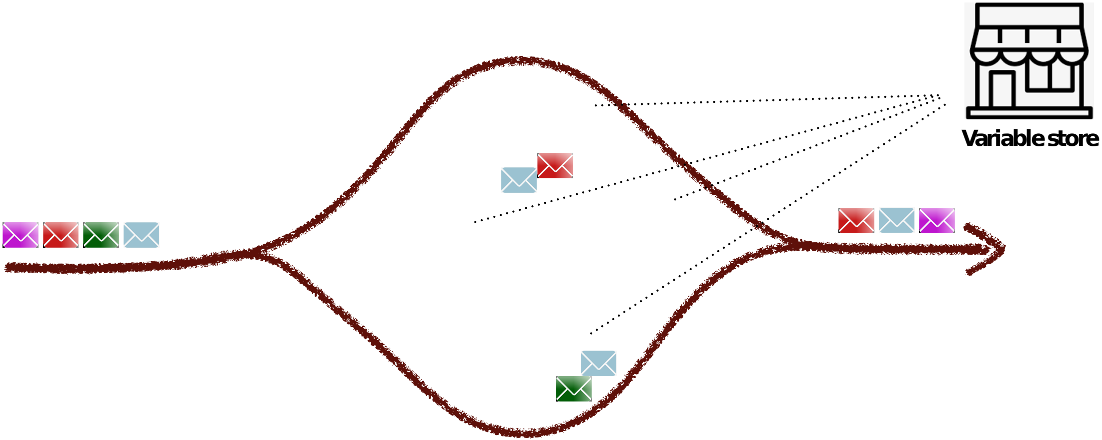
Overview

- Kleene algebra: axiomatisation of regular languages, used to reason about simple programs
- Many variants and extensions that enable analysis of more complicated programs:
  - Kleene algebra with tests (KAT)
  - NetKAT
  - Concurrent extensions (CKA, POCKA, CKAO, …)
  - …
- Key limitation of NetKAT: stateless and sequential

Overview

- Kleene algebra: axiomatisation of regular languages, used to reason about simple programs
- Many variants and extensions that enable analysis of more complicated programs:
  - Kleene algebra with tests (KAT)
  - NetKAT
  - Concurrent extensions (CKA, POCKA, CKAO, ...)
  - ...
- Key limitation of NetKAT: stateless and sequential
- This paper: Concurrent NetKAT, combination of POCKA and a multi-packet extension of NetKAT
  - Sound and complete axiomatisation of CNetKAT
  - Examples of applicability of CNetKAT for modelling and analysing concurrent network behaviours


Variable store



Do we always see a green packet at switch 3 before we see a red packet at switch 2?

A packet is a record of fields $f_1, \ldots, f_N$ and values $v_1, \ldots, v_N$.

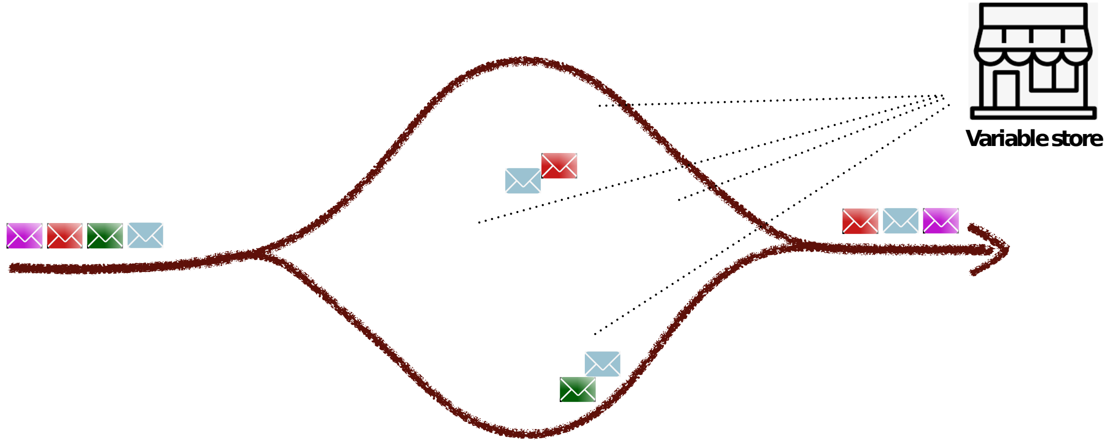
A packet is a record of fields $f_1, \ldots, f_N$ and values $v_1, \ldots, v_N$. Part of syntax:

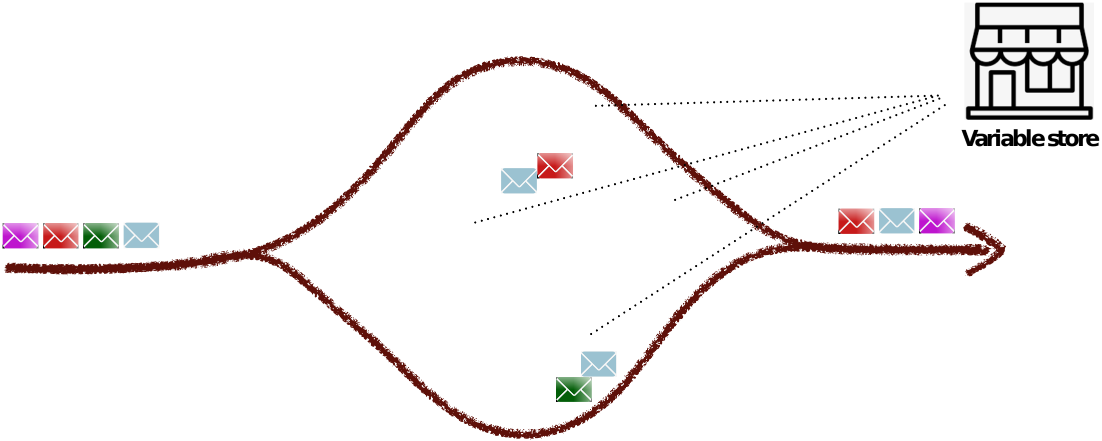
$$f = n \mid f \leftarrow n \mid v = n \mid v \leftarrow n \mid v \leftarrow v'$$

A packet is a record of fields $f_1, \ldots, f_N$ and values $v_1, \ldots, v_N$. Part of syntax:

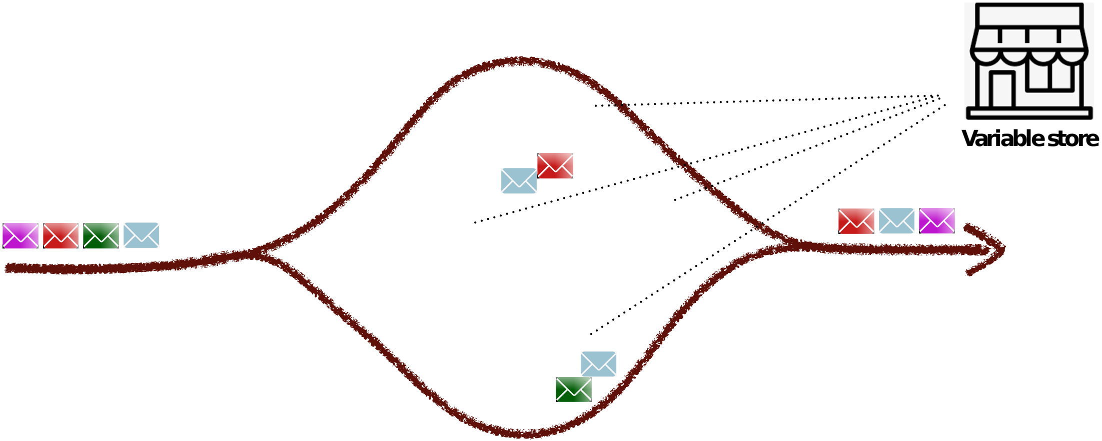
$$f = n \mid f \leftarrow n \mid v = n \mid v \leftarrow n \mid v \leftarrow v'$$

More complicated programs with $\quad + \quad ; \quad * \quad \|$

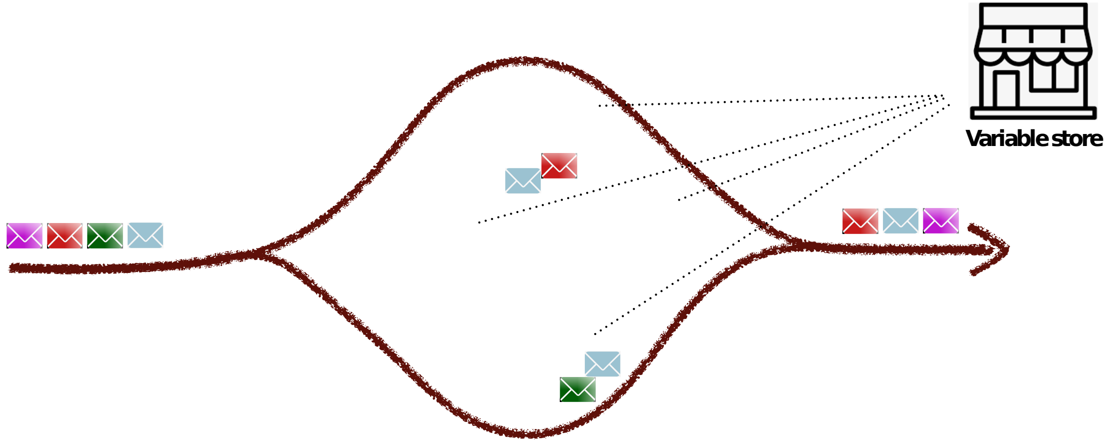

Variable store



$$p_1 := \mathsf{sw}{=}1 \,;\, ((\mathsf{tag}{=}\textcolor{red}{\mathit{red}} \,;\, \mathsf{sw}{\leftarrow}2) \parallel (\mathsf{tag}{=}\textcolor{green}{\mathit{green}} \,;\, \mathsf{sw}{\leftarrow}3))$$



$$\mathrm{sw}{=}1\,;\,((v{=}1\,;\,\mathrm{tag}{=}\mathit{red}\,;\,\mathrm{sw}{\leftarrow}2)\parallel(\mathrm{tag}{=}\mathit{green}\,;\,\mathrm{sw}{\leftarrow}3\,;\,v{\leftarrow}1))$$



$$p \triangleq v \leftarrow 0 \; ; (p_1 \parallel p_2 \parallel p_3 \parallel p_4)^*$$

Type of Semantics

$$[\![-]\!] : 2^{\mathsf{Pk}} \to 2^{\mathsf{Pom} \cdot 2^{\mathsf{Pk}}}$$

Type of Semantics

$$[\![-]\!] : 2^{\mathsf{Pk}} \to 2^{\mathsf{Pom} \cdot 2^{\mathsf{Pk}}}$$

$\mathbf{u} \cdot b \in [\![p]\!](a)$ means “there is an execution of p on input set a that changes the global variables according to $\mathbf{u}$, and the set of output packets produced is b”

Type of Semantics

$$[\![-]\!] : 2^{\mathsf{Pk}} \to 2^{\mathsf{Pom}\cdot 2^{\mathsf{Pk}}}$$

$\mathbf{u} \cdot b \in [\![p]\!](a)$ means "there is an execution of p on input set a that changes the global variables according to **u**, and the set of output packets produced is b"

$$[\![p \parallel q]\!](a) \triangleq \{(\mathbf{u} \parallel \mathbf{v}) \cdot (b \cup c) \mid \mathbf{u} \cdot b \in [\![p]\!](a), \mathbf{v} \cdot c \in [\![q]\!](a)\}$$

Type of Semantics

$$[\![-]\!] : 2^{\mathsf{Pk}} \to 2^{\mathsf{Pom}\cdot 2^{\mathsf{Pk}}}$$

$\mathbf{u} \cdot b \in [\![p]\!](a)$ means “there is an execution of p on input set a that changes the global variables according to $\mathbf{u}$, and the set of output packets produced is b”

$$[\![p \parallel q]\!](a) \triangleq \{(\mathbf{u} \parallel \mathbf{v}) \cdot (b \cup c) \mid \mathbf{u} \cdot b \in [\![p]\!](a), \mathbf{v} \cdot c \in [\![q]\!](a)\}$$

$$[\![\mathrm{dup}]\!](a) \triangleq \{a \cdot a\}$$

Type of Semantics

$$[\![-]\!] : 2^{\mathsf{Pk}} \to 2^{\mathsf{Pom}\cdot 2^{\mathsf{Pk}}}$$

$\mathbf{u} \cdot b \in [\![p]\!](a)$ means "there is an execution of p on input set a that changes the global variables according to $\mathbf{u}$, and the set of output packets produced is b"

$$[\![p \parallel q]\!](a) \triangleq \{(\mathbf{u} \parallel \mathbf{v}) \cdot (b \cup c) \mid \mathbf{u} \cdot b \in [\![p]\!](a), \mathbf{v} \cdot c \in [\![q]\!](a)\}$$

$$[\![\mathsf{dup}]\!](a) \triangleq \{a \cdot a\}$$

Full semantics: $[\![p]\!]{\downarrow}(a)$

CNetKAT contains the following axioms:

CNetKAT contains the following axioms:

- Kleene algebra axioms

CNetKAT contains the following axioms:

- Kleene algebra axioms
- Axioms for the parallel from concurrent Kleene algebra

CNetKAT contains the following axioms:

- Kleene algebra axioms
- Axioms for the parallel from concurrent Kleene algebra
- NetKAT packet axioms

CNetKAT contains the following axioms:

- Kleene algebra axioms
- Axioms for the parallel from concurrent Kleene algebra
- NetKAT packet axioms
- A Boolean algebra for packet tests

CNetKAT contains the following axioms:

- Kleene algebra axioms
- Axioms for the parallel from concurrent Kleene algebra
- NetKAT packet axioms
- A Boolean algebra for packet tests
- A PCDL for the observations

CNetKAT contains the following axioms:

- Kleene algebra axioms
- Axioms for the parallel from concurrent Kleene algebra
- NetKAT packet axioms
- A Boolean algebra for packet tests
- A PCDL for the observations
- Axioms connecting the BA and PCDL operators to the Kleene algebra ones

CNetKAT contains the following axioms:

- Kleene algebra axioms
- Axioms for the parallel from concurrent Kleene algebra
- NetKAT packet axioms
- A Boolean algebra for packet tests
- A PCDL for the observations
- Axioms connecting the BA and PCDL operators to the Kleene algebra ones
- Axioms connecting the local and the global state

$$\mathsf{drop} \parallel p = p \qquad\qquad \mathsf{abort} \parallel p = \mathsf{abort}$$

$$\mathsf{drop} \parallel p = p \qquad \mathsf{abort} \parallel p = \mathsf{abort}$$

$$t \vee t' = t \parallel t' \qquad o \vee o' = o + o'$$

$$o \leq \overline{o'} \Leftrightarrow o \wedge o' = \bot$$

$$o \leq \overline{o'} \Leftrightarrow o \wedge o' = \bot$$

$[\![v = n]\!]$ contains all partial functions that assign n to v.

$$o \leq \overline{o'} \Leftrightarrow o \wedge o' = \bot$$

$[\![v = n]\!]$ contains all partial functions that assign n to v.

$[\![\overline{v = n}]\!]$ should contain all partial functions that assign a value to v different than n.

Theorem

Let $p, q \in \mathrm{Prg}$. We have CNetKAT $\vdash p = q$ if and only if $[\![p]\!]\!\downarrow = [\![q]\!]\!\downarrow$.

1. Define a normal form for CNetKAT programs:

$$\Pi_a \, ; p = \Pi_a \, ; \sum_{j \in J} \left(u_j \, ; \Pi_{b_j} \right)$$

1. Define a normal form for CNetKAT programs:

$$\Pi_a \,;\, p = \Pi_a \,;\, \sum_{j \in J} \left(u_j \,;\, \Pi_{b_j} \right)$$

2. Obtain completeness for Π_a-shaped programs from NetKAT completeness.

1. Define a normal form for CNetKAT programs:

$$\Pi_a \,;\, p = \Pi_a \,;\, \sum_{j \in J} \left(u_j \,;\, \Pi_{b_j} \right)$$

2. Obtain completeness for Π_a-shaped programs from NetKAT completeness.
3. Using completeness of POCKA, obtain completeness for programs of the form $s \,;\, \Pi_a$ (and sums thereof), where s is a state program.

1. Define a normal form for CNetKAT programs:

$$\Pi_a \mathbin{;} p = \Pi_a \mathbin{;} \sum_{j \in J} \left(u_j \mathbin{;} \Pi_{b_j} \right)$$

2. Obtain completeness for Π_a-shaped programs from NetKAT completeness.
3. Using completeness of POCKA, obtain completeness for programs of the form $s \mathbin{;} \Pi_a$ (and sums thereof), where s is a state program.
4. Lastly, we combine these results to prove that if p and q have the same behaviour on input a, the program $\Pi_a \mathbin{;} p$ is provably equivalent to $\Pi_a \mathbin{;} q$. We can conclude using the extensionality axiom: $\forall a \in 2^{\mathsf{Pk}}.(\Pi_a \mathbin{;} p = \Pi_a \mathbin{;} q) \Rightarrow p = q$.

$$v \leftarrow 1 \,;\, v = 2$$

$$v \leftarrow 1\,;\, v = 2$$

$$(v \leftarrow 1\,;\, v = 2) \parallel v \leftarrow 2$$

$$v \leftarrow 1 \,;\, v = 2$$

$$(v \leftarrow 1 \,;\, v = 2) \parallel v \leftarrow 2$$

How to study isolated behaviour? –> guarded pomsets

Final Remarks

- With these tools we can analyse example from beginning

Final Remarks

- With these tools we can analyse example from beginning
- Missing: a decision procedure

Final Remarks

- With these tools we can analyse example from beginning
- Missing: a decision procedure
- Future work: lots of more case studies