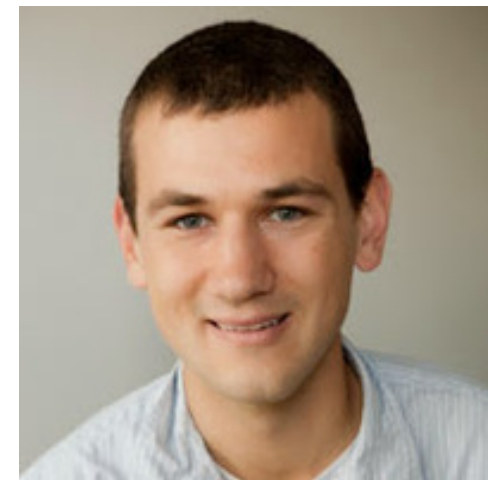


Concurrent NetKAT

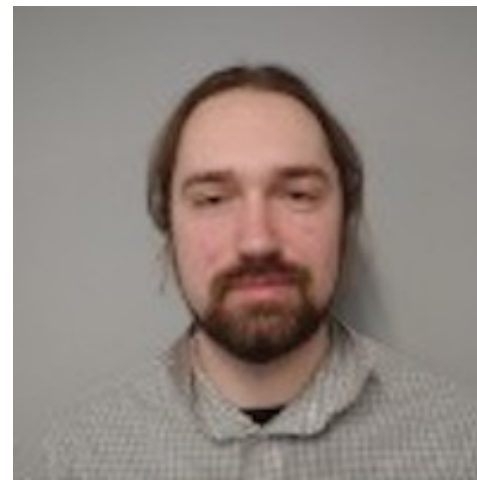
Modeling and analyzing multi-packet and global state interaction



Jana Wagemaker,



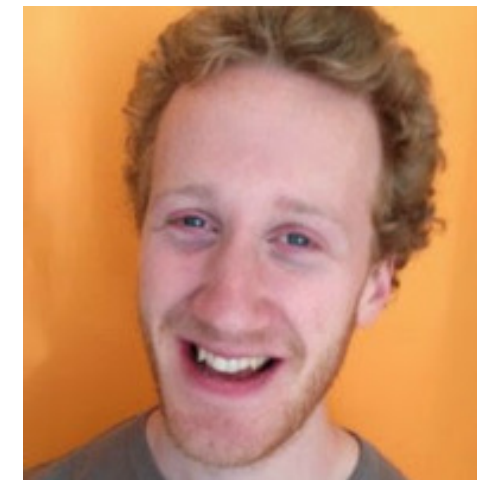
Nate Foster,



Tobias Kappé,



Dexter Kozen,

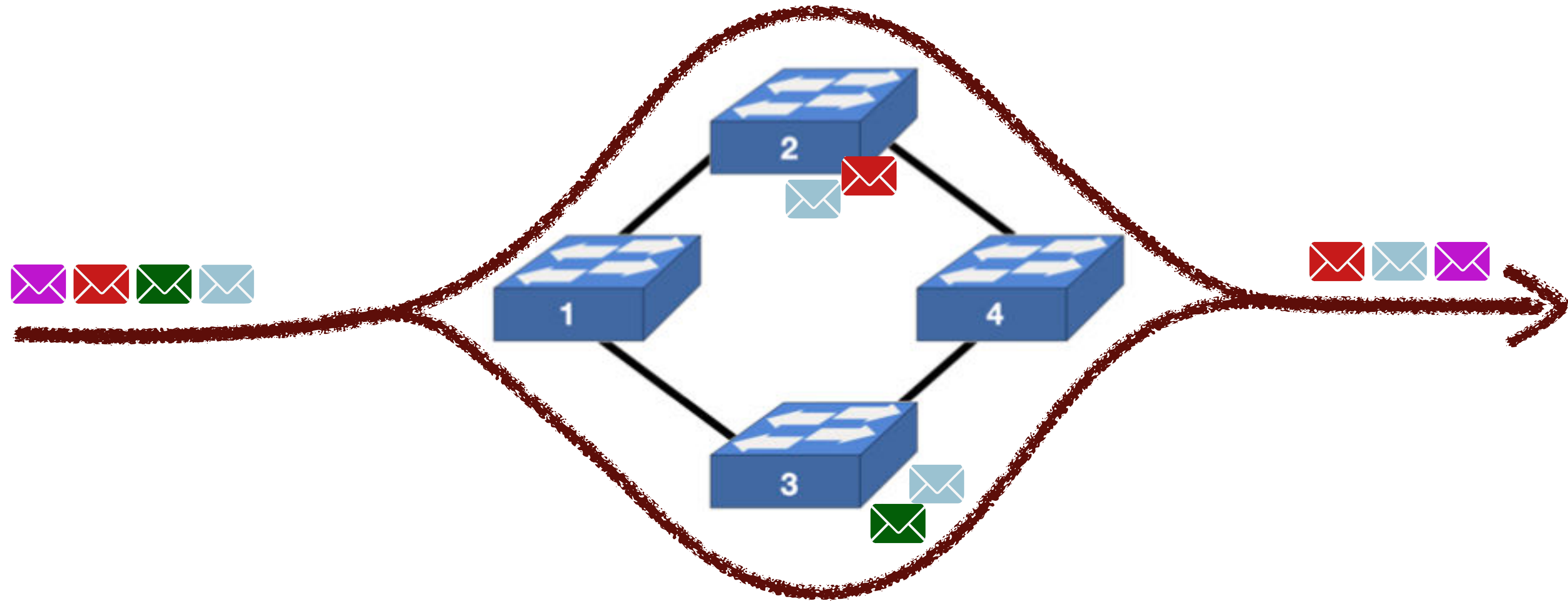


Jurriaan Rot,

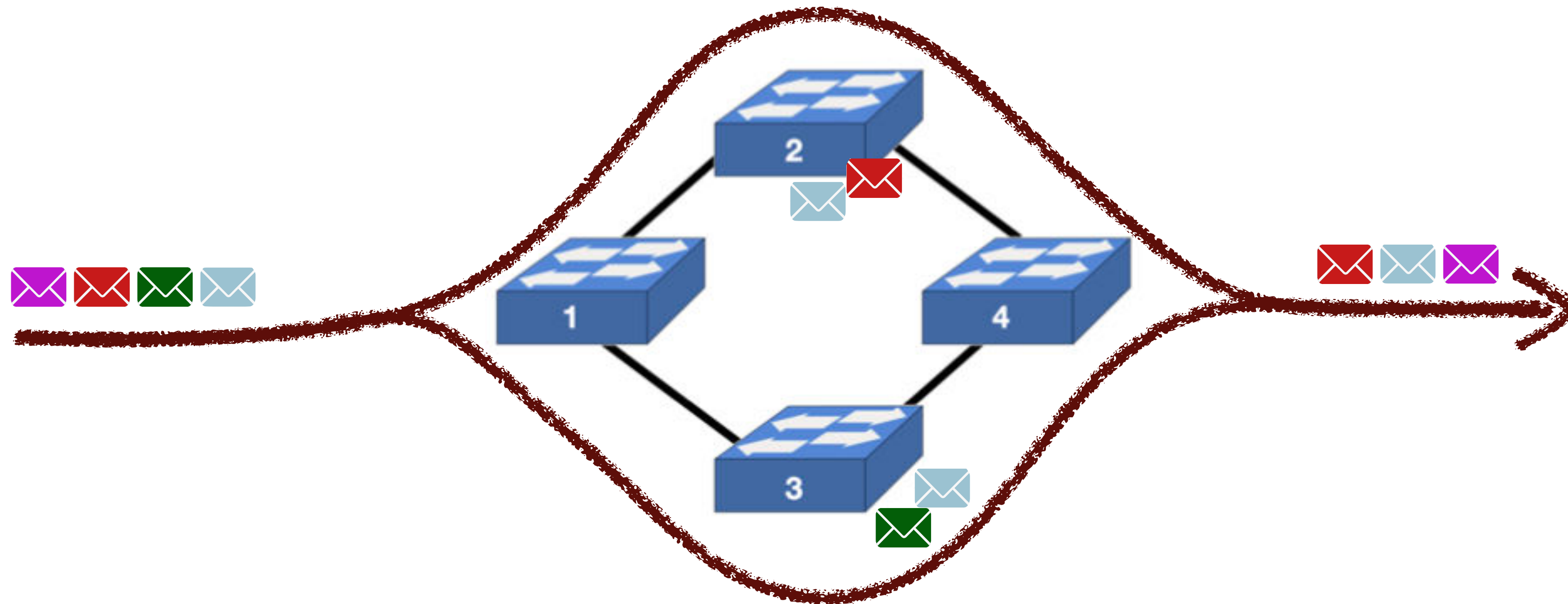


Alexandra Silva

Context

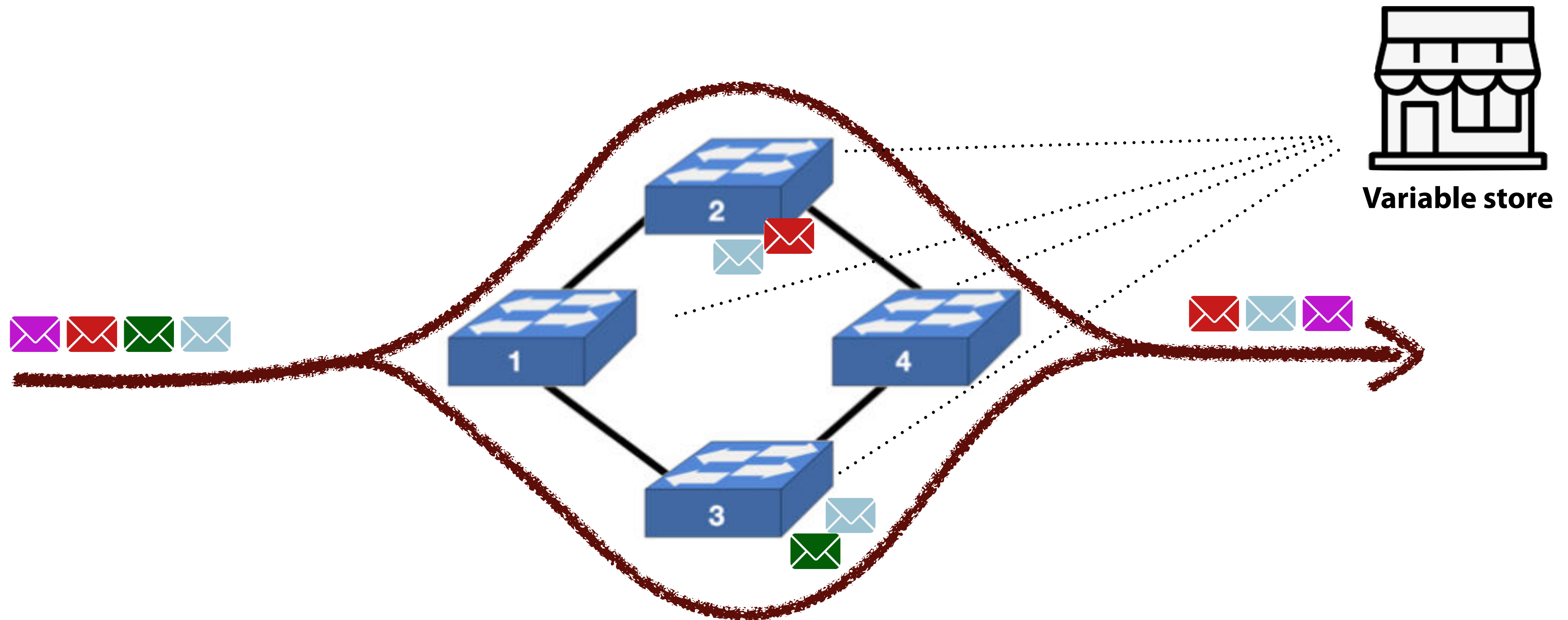


Context



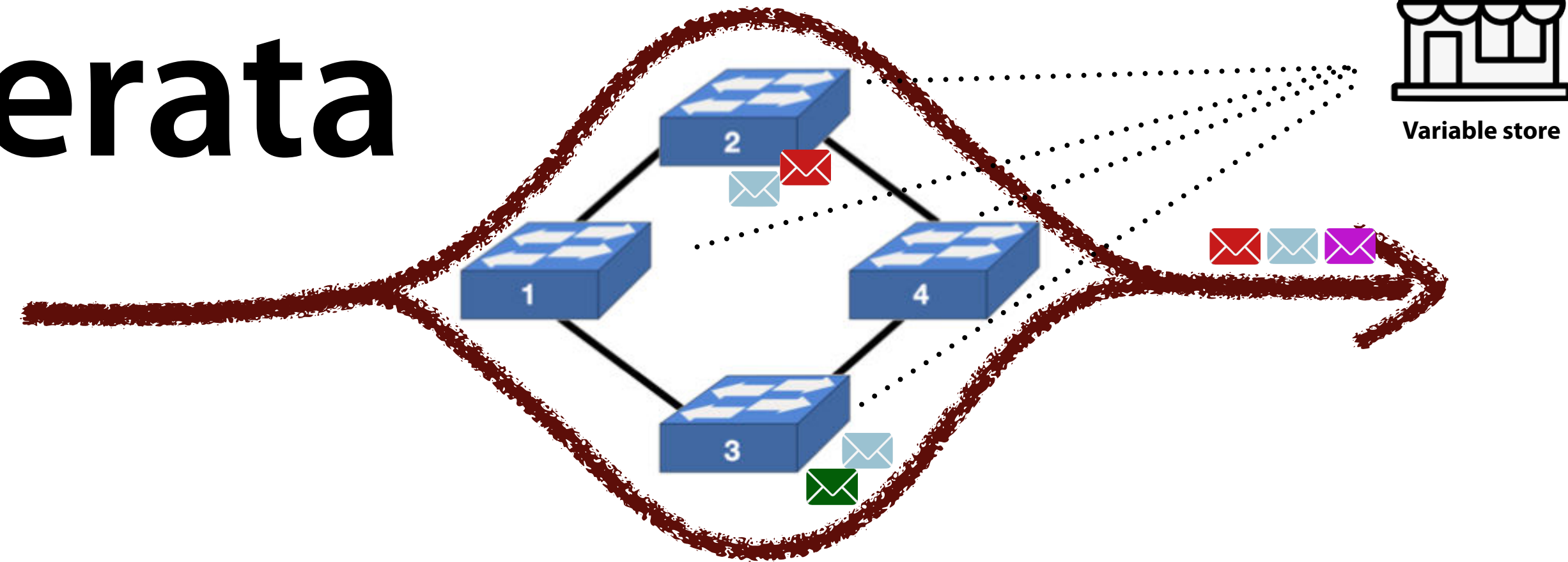
Do I always see a green packet in 3 before I see a red packet in 2?

Context



Do I always see a green packet in 3 before I see a red packet in 2?

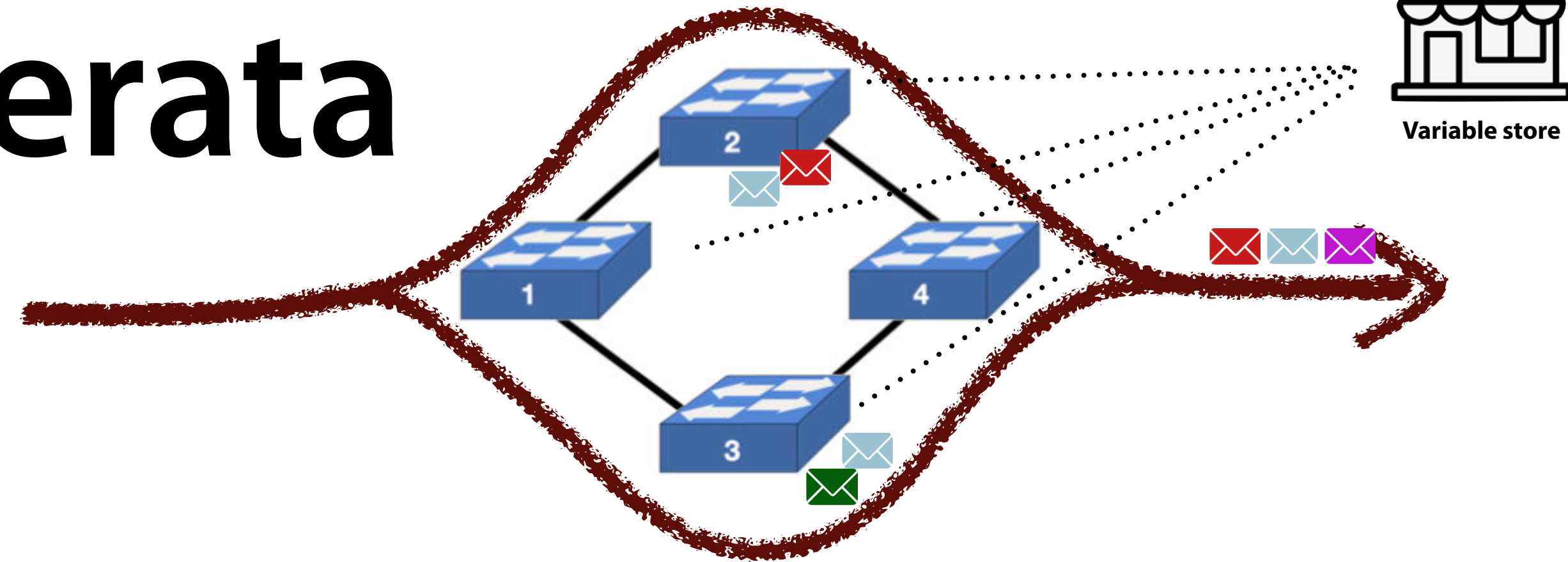
Desiderata



Do I always see a green packet in 3 before I see a red packet in 2?

- ✓ Language to specify and verify network behaviour
- ✓ Compositional semantics
- ✓ Multiple Packets
- ✓ Access to a variable store

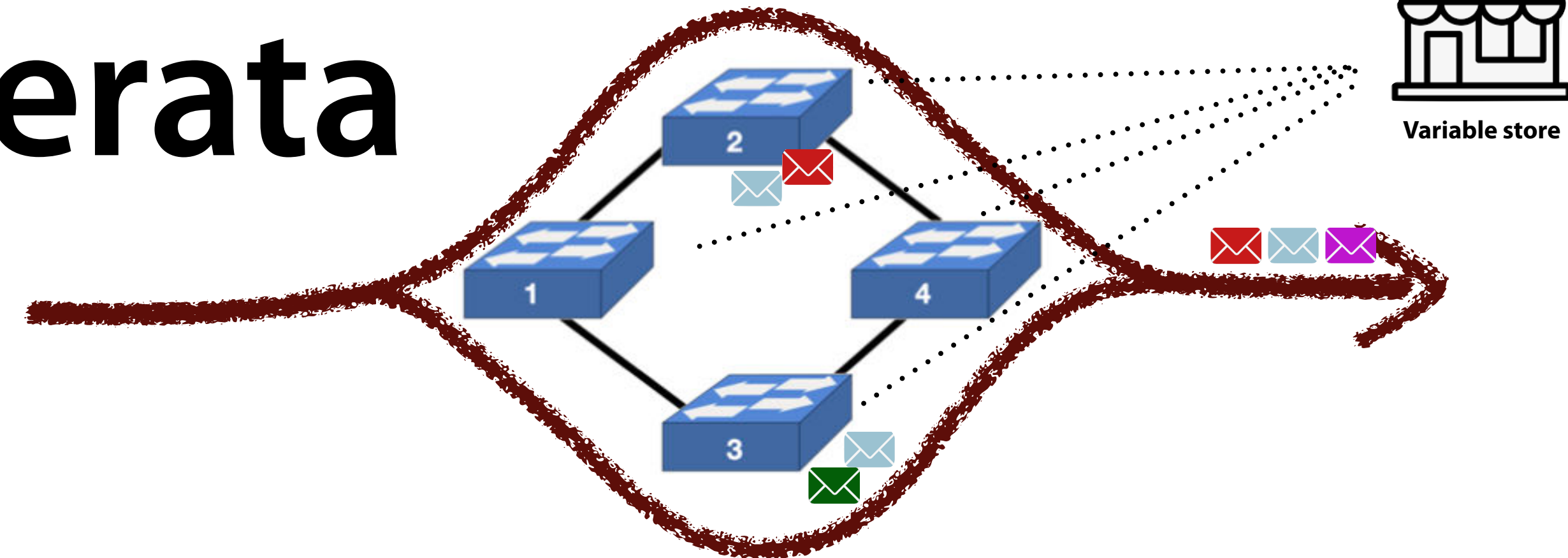
Desiderata



Do I always see a green packet in 3 before I see a red packet in 2?

- ✓ Language to specify and verify network behaviour
 - ✓ Compositional semantics
 - ✓ Multiple Packets
 - ✓ Access to a variable store
- } **Concurrency**

Desiderata



Do I always see a green packet in 3 before I see a red packet in 2?

- NetKAT** {
- ✓ Language to specify and verify network behaviour
 - ✓ Compositional semantics
 - ✓ Multiple Packets
 - ✓ Access to a variable store
- } **Concurrency**

NetKAT

Kleene Algebra with Tests
(KAT)

+

Specialised constructs particular to
network topology and packet
switching

$\text{pol} ::= \text{drop}$
| skip
| $\text{field} = \text{val}$
| $\text{pol1} + \text{pol2}$
| $\text{pol1} ; \text{pol2}$
| $!\text{pol}$

| pol^*
| $\text{field} := \text{val}$
| dup

NetKAT

Kleene Algebra with Tests
(KAT)

+

Specialised constructs particular to
network topology and packet
switching

```
pol ::= drop  
      | skip  
      | field = val  
      | pol1 + pol2  
      | pol1 ; pol2  
      | !pol
```

```
| pol*  
| field := val  
| dup
```

Boolean Algebra

NetKAT

Kleene Algebra with Tests
(KAT)

+

Specialised constructs particular to
network topology and packet
switching

pol ::= drop

| skip

| field = val

| pol1 + pol2

| pol1 ; pol2

| !pol

| pol*

| field := val

| dup

Boolean Algebra

+

Kleene Algebra

NetKAT

Kleene Algebra with Tests
(KAT)

+

Specialised constructs particular to
network topology and packet
switching

pol ::= drop
 | skip
 | field = val
 | pol1 + pol2
 | pol1 ; pol2
 | !pol

 | pol*
 | field := val
 | dup

Boolean Algebra

+

Kleene Algebra

+

Packet Primitives

NetKAT

Kleene Algebra with Tests
(KAT)

+

Specialised constructs particular to
network topology and packet
switching

pol ::= drop
 | skip
 | field = val
 | pol1 + pol2
 | pol1 ; pol2
 | !pol

 | pol*
 | field := val
 | dup

Boolean Algebra

+

Kleene Algebra

+

Packet Primitives

NetKAT

Kleene Algebra with Tests
(KAT)

+

Specialised constructs particular to
network topology and packet
switching

pol ::= drop
| skip
| field = val
| pol1 + pol2
| pol1 ; pol2
| !pol

| pol*
| field := val
| dup

Boolean Algebra

+

Kleene Algebra

+

Packet Primitives

Basic actions and tests in KAT are pairs
(field, val)

A simple program

```
sw = 3 ; pt = 2 ; type= SSH ; pt := 3 ; sw := 5
```



A simple program

```
sw = 3 ; pt = 2 ; type= SSH ; pt := 3 ; sw := 5
```



A simple program

```
sw = 3 ; pt = 2 ; type= SSH ; pt := 3 ; sw := 5
```



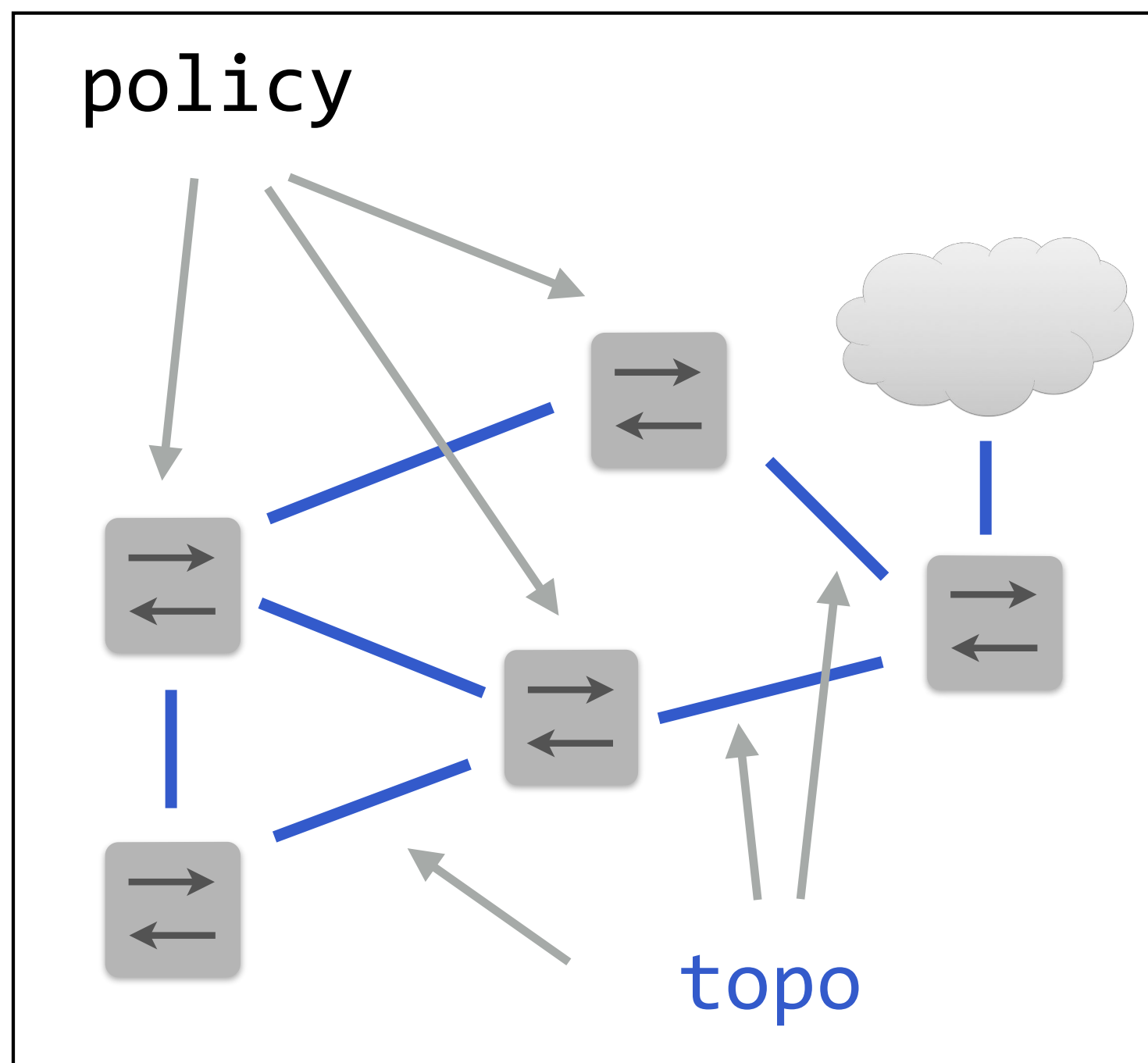
A simple program

`sw = 3 ; pt = 2 ; type= SSH ; pt := 3 ; sw := 5`



Networks in NetKAT

The behaviour of an entire network can be encoded in NetKAT by interleaving steps of processions by switches and topology


$$\begin{aligned} & \text{policy} \\ & + \\ & (\text{policy}; \text{topo}; \text{policy}) \\ & + \\ & (\text{policy}; \text{topo}; \text{policy}; \text{topo}); \text{policy} \\ & \vdots \\ & (\text{policy}; \text{topo})^*; \text{policy} \end{aligned}$$

Networks in NetKAT

Switch forwarding tables and network topologies can be represented in NetKAT using simple encodings

Pattern	Actions
dstport=22	Drop
srcip=10.0.0.1	Forward 1
*	Forward 2



```
if dstport=22 then false  
else if srcip=10.0.0.1 then port := 1  
else port := 2
```

Networks in NetKAT

Switch forwarding tables and network topologies can be represented in NetKAT using simple encodings

Pattern	Actions
dstport=22	Drop
srcip=10.0.0.1	Forward 1
*	Forward 2



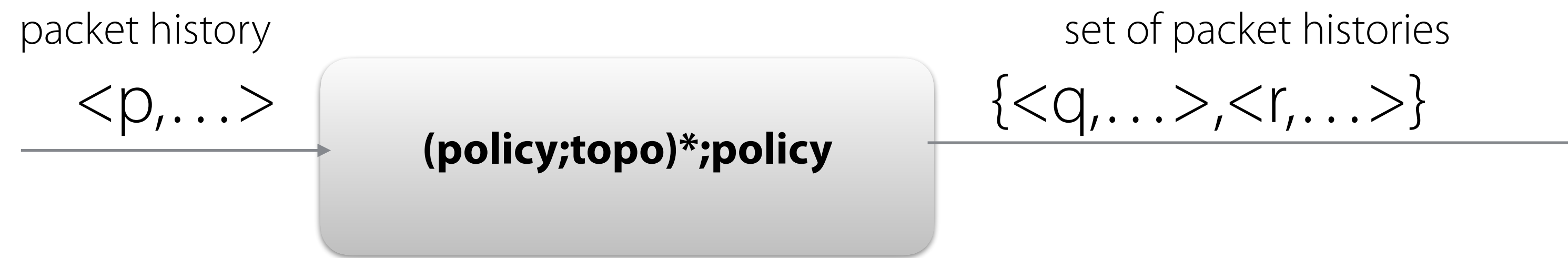
```
if dstport=22 then false
else if srcip=10.0.0.1 then port := 1
else port := 2
```

KAT = simple imperative language

If b **then** p **else** $q = b; p + !b; q$

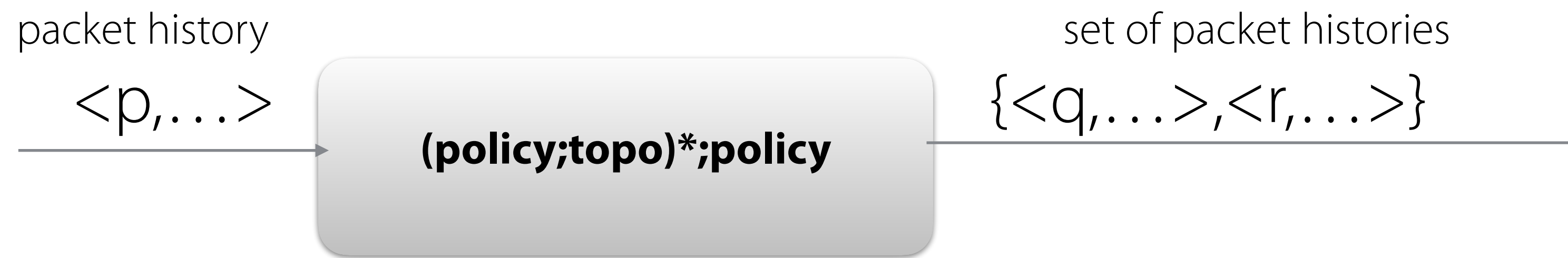
While b **do** $p = (bp)^*!b$

Semantics



$$\llbracket e \rrbracket : H \rightarrow 2^H$$

Semantics



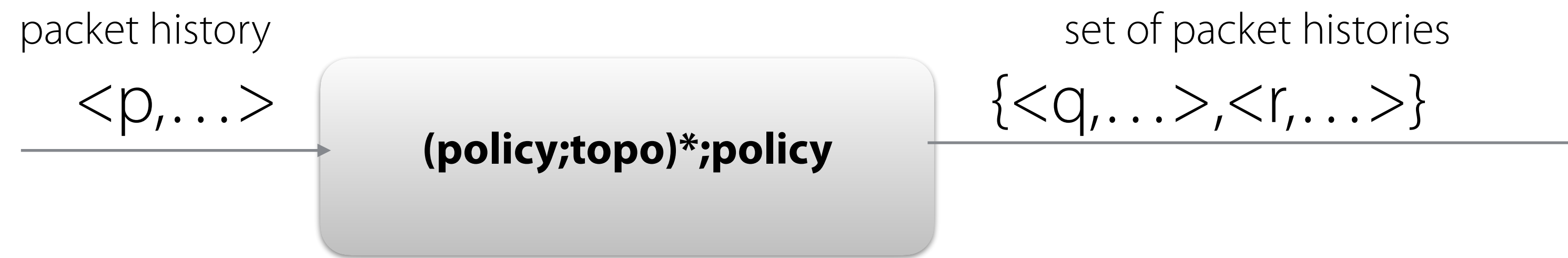
$$\llbracket e \rrbracket : H \rightarrow 2^H$$

$$\llbracket x \leftarrow n \rrbracket (\pi_1 :: \sigma) \triangleq \{ \pi_1[n/x] :: \sigma \}$$

$$\llbracket x = n \rrbracket (\pi_1 :: \sigma) \triangleq \begin{cases} \{ \pi_1 :: \sigma \} & \text{if } \pi_1(x) = n \\ \emptyset & \text{if } \pi_1(x) \neq n \end{cases}$$

$$\llbracket \text{dup} \rrbracket (\pi_1 :: \sigma) \triangleq \{ \pi_1 :: \pi_1 :: \sigma \}$$

Semantics



$$\llbracket e \rrbracket : H \rightarrow 2^H$$

$$\llbracket x \leftarrow n \rrbracket (\pi_1 :: \sigma) \triangleq \{\pi_1[n/x] :: \sigma\}$$

$$\llbracket x = n \rrbracket (\pi_1 :: \sigma) \triangleq \begin{cases} \{\pi_1 :: \sigma\} & \text{if } \pi_1(x) = n \\ \emptyset & \text{if } \pi_1(x) \neq n \end{cases}$$

$$\llbracket \text{dup} \rrbracket (\pi_1 :: \sigma) \triangleq \{\pi_1 :: \pi_1 :: \sigma\}$$

Compositional

NetKAT proof system

NetKAT proof system

$$x \leftarrow n ; y \leftarrow m \equiv y \leftarrow m ; x \leftarrow n \quad (x \neq y)$$

assignments to distinct fields may be done in either order

$$x \leftarrow n ; y = m \equiv y = m ; x \leftarrow n \quad (x \neq y)$$

an assignment to a field does not affect a different field

NetKAT proof system

$$x \leftarrow n ; y \leftarrow m \equiv y \leftarrow m ; x \leftarrow n \quad (x \neq y)$$

assignments to distinct fields may be done in either order

$$x \leftarrow n ; y = m \equiv y = m ; x \leftarrow n \quad (x \neq y)$$

an assignment to a field does not affect a different field

$$x \leftarrow n \equiv x \leftarrow n ; x = n$$

an assignment causes the field to have that value

$$x = n ; x \leftarrow n \equiv x = n$$

an assignment of a value that the field already has is redundant

$$x \leftarrow n ; x \leftarrow m \equiv x \leftarrow m$$

a second assignment to the same field overrides the first

NetKAT proof system

$$x \leftarrow n ; y \leftarrow m \equiv y \leftarrow m ; x \leftarrow n \quad (x \neq y)$$

assignments to distinct fields may be done in either order

$$x \leftarrow n ; y = m \equiv y = m ; x \leftarrow n \quad (x \neq y)$$

an assignment to a field does not affect a different field

$$x \leftarrow n \equiv x \leftarrow n ; x = n$$

an assignment causes the field to have that value

$$x = n ; x \leftarrow n \equiv x = n$$

an assignment of a value that the field already has is redundant

$$x \leftarrow n ; x \leftarrow m \equiv x \leftarrow m$$

a second assignment to the same field overrides the first

$$x = n ; x = m \equiv 0 \quad (n \neq m) \qquad \left(\sum_n x = n \right) \equiv 1$$

every field has exactly one value

NetKAT proof system

$$x \leftarrow n; y \leftarrow m \equiv y \leftarrow m; x \leftarrow n \quad (x \neq y)$$

assignments to distinct fields may be done in either order

Soundness and Completeness

$$\llbracket p \rrbracket = \llbracket q \rrbracket \iff \vdash p \equiv q$$

$$x = n; x = m \equiv 0 \quad (n \neq m) \qquad (\sum_n x = n) \equiv 1$$

every field has exactly one value

Verification via program equivalence

Can host A communicate with host B?

$$sw = A ; t(pt)^* ; sw = B \equiv 0 \text{ (drop)}.$$

Many more properties reduce to equivalence questions: loop freedom, access control, ...

$$\alpha ; pt(pt)^* ; \alpha = 0$$

Checkpoint

- ✓ Language to specify and verify network behaviour
- ✓ Compositional semantics

Checkpoint

- NetKAT** {
- ✓ Language to specify and verify network behaviour
 - ✓ Compositional semantics

Checkpoint

- NetKAT** {
- ✓ Language to specify and verify network behaviour
 - ✓ Compositional semantics

**Reachability
Congestion**

Checkpoint

- NetKAT** {
- ✓ Language to specify and verify network behaviour
 - ✓ Compositional semantics

**Reachability
Congestion**

- ✓ Multiple Packets
 - ✓ Access to a variable store
- } **Concurrency**

Checkpoint

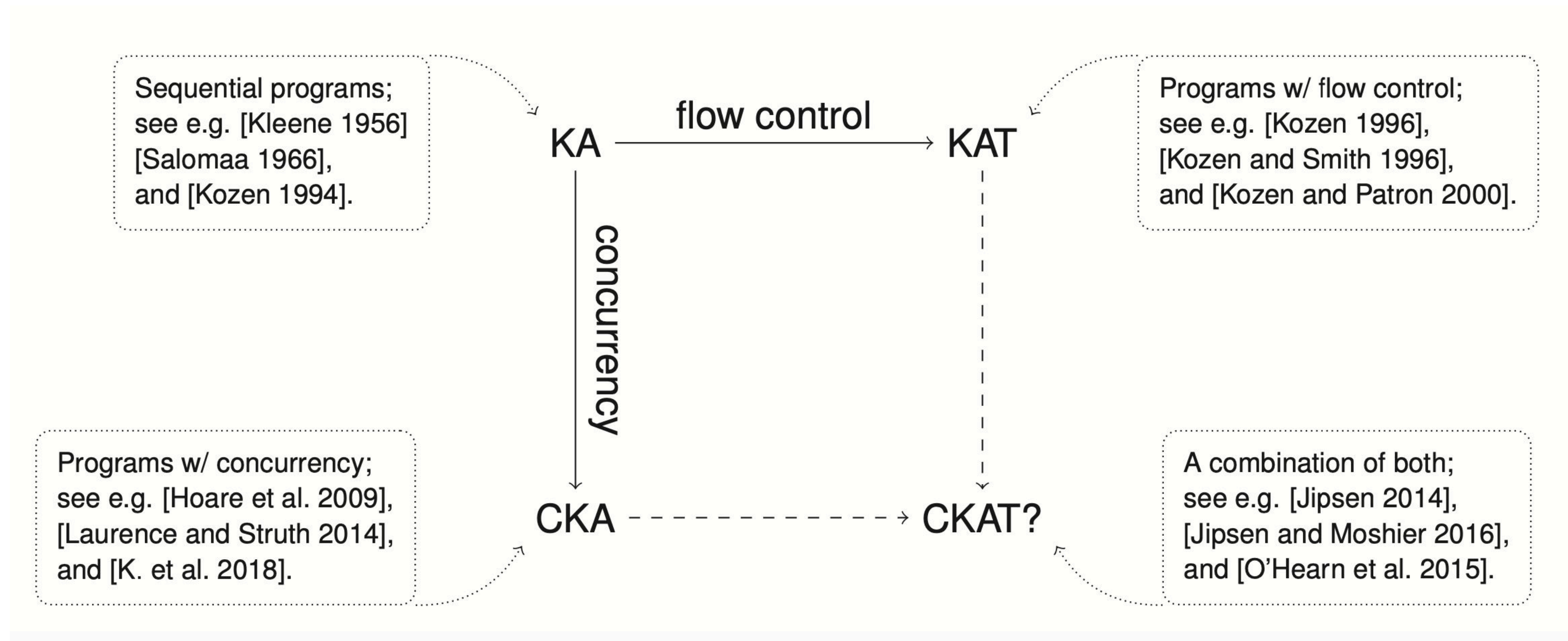
- NetKAT** {
- ✓ Language to specify and verify network behaviour
 - ✓ Compositional semantics

**Reachability
Congestion**

- ✓ Multiple Packets
 - ✓ Access to a variable store
- } **Concurrency**

**Minimal additions to NetKAT: algebra of global state plus
sets of packets plus
parallel processing**

Adding concurrency to KAT



Concurrent Kleene Algebra

(Hoare, Moeller, Struth, Wehrman'09)

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

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

$$e \parallel 1 \equiv e$$

$$e \parallel 0 \equiv 0$$

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

Concurrent Kleene Algebra

(Hoare, Moeller, Struth, Wehrman'09)

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

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

$$e \parallel 1 \equiv e$$

$$e \parallel 0 \equiv 0$$

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

$$(e \parallel g) \cdot (f \parallel h) \leq (e \cdot f) \parallel (g \cdot h).$$

Exchange law

Concurrent Kleene Algebra

(Hoare, Moeller, Struth, Wehrman'09)

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

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

$$e \parallel 1 \equiv e$$

$$e \parallel 0 \equiv 0$$

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

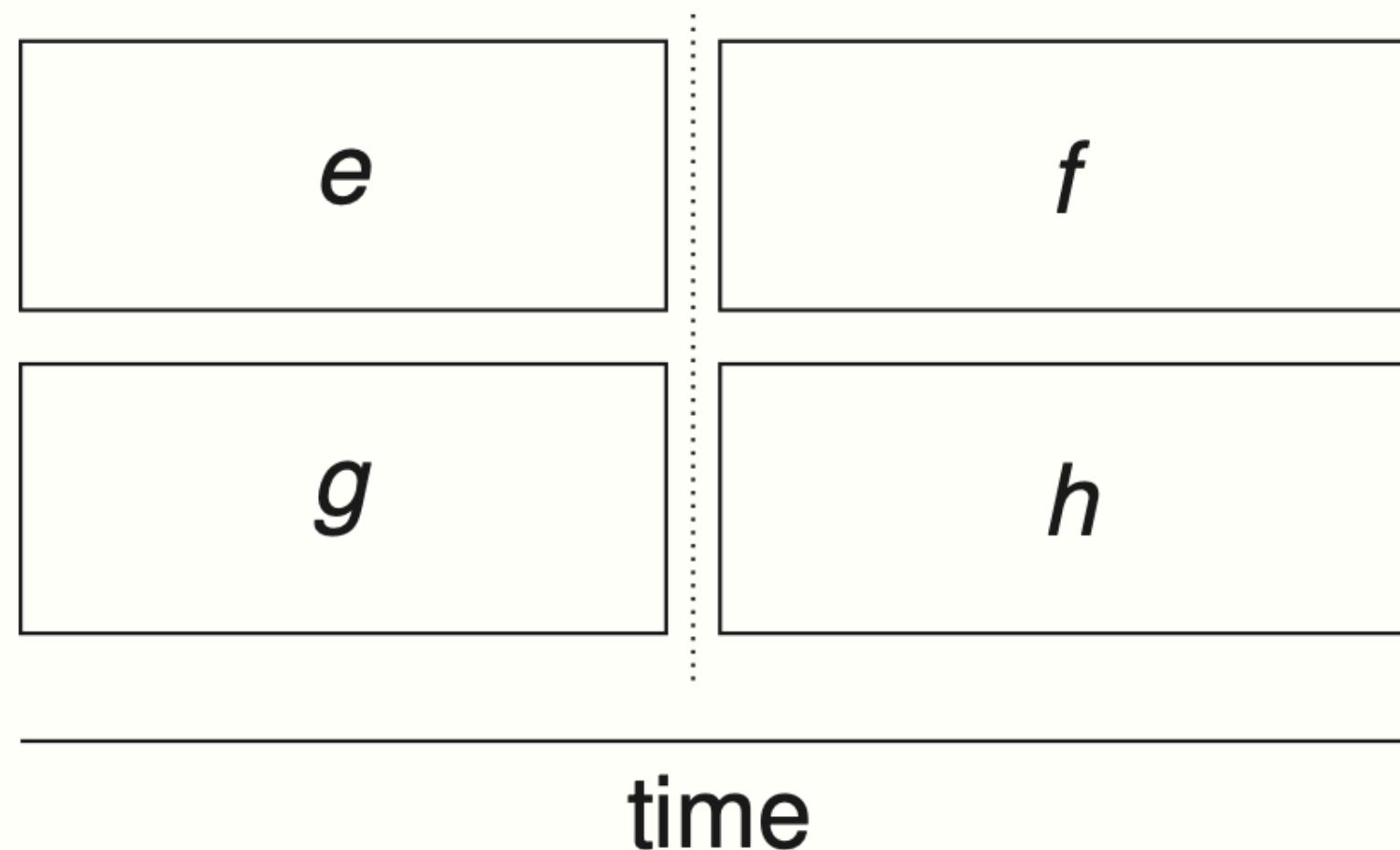
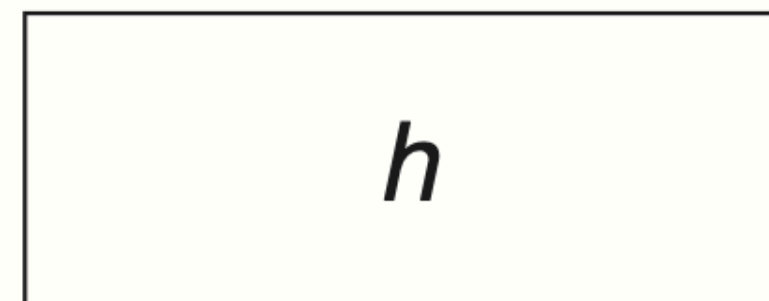
$$(e \parallel g) \cdot (f \parallel h) \leq (e \cdot f) \parallel (g \cdot h).$$

Exchange law

Thread #1



Thread #2



Concurrent Kleene Algebra

(Hoare, Moeller, Struth, Wehrman'09)

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

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

$$e \parallel 1 \equiv e$$

$$e \parallel 0 \equiv 0$$

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

$$(e \parallel g) \cdot (f \parallel h) \leq (e \cdot f) \parallel (g \cdot h).$$

Exchange law

Thread #1



Thread #2



time

Interleaving as
special case:
 $p \parallel q = p \cdot q + q \cdot p$

CKAT [Jipsen'14]

CKA terms

$$e, f ::= \underbrace{p \mid 0 \mid 1 \mid a \in \Sigma \mid e + f \mid e \cdot f \mid e^* \mid e \parallel f}_{\text{KAT terms}}$$

KAT terms

$$p, q ::= \perp \mid \top \mid t \in T \mid p \vee q \mid p \wedge q \mid \bar{p}$$

The problem with CKAT

$$\begin{aligned} p \cdot e \cdot \bar{p} &\leq (p \parallel e) \cdot \bar{p} \\ &\equiv (p \parallel e) \cdot (\bar{p} \parallel 1) \\ &\leq (p \cdot \bar{p}) \parallel (e \cdot 1) \\ &\equiv (p \cdot \bar{p}) \parallel e \end{aligned}$$

The problem with CKAT

$$\begin{aligned} p \cdot e \cdot \bar{p} &\leq (p \parallel e) \cdot \bar{p} \\ &\equiv (p \parallel e) \cdot (\bar{p} \parallel 1) \\ &\leq (p \cdot \bar{p}) \parallel (e \cdot 1) \\ &\equiv (p \cdot \bar{p}) \parallel e \\ &\equiv (p \wedge \bar{p}) \parallel e \\ &\equiv \perp \parallel e \\ &\equiv 0 \parallel e \\ &\equiv 0 \end{aligned}$$

The problem with CKAT

$$\begin{aligned} p \cdot e \cdot \bar{p} &\leq (p \parallel e) \cdot \bar{p} \\ &\equiv (p \parallel e) \cdot (\bar{p} \parallel 1) \\ &\leq (p \cdot \bar{p}) \parallel (e \cdot 1) \\ &\equiv (p \cdot \bar{p}) \parallel e \\ &\equiv (p \wedge \bar{p}) \parallel e \\ &\equiv \perp \parallel e \\ &\equiv 0 \parallel e \\ &\equiv 0 \end{aligned}$$

The problem with CKAT

$$\begin{aligned} p \cdot e \cdot \bar{p} &\leq (p \parallel e) \cdot \bar{p} \\ &\equiv (p \parallel e) \cdot (\bar{p} \parallel 1) \\ &\leq (p \cdot \bar{p}) \parallel (e \cdot 1) \\ &\equiv (p \cdot \bar{p}) \parallel e \\ &\equiv (p \wedge \bar{p}) \parallel e \\ &\equiv \perp \parallel e \\ &\equiv 0 \parallel e \\ &\equiv 0 \end{aligned}$$

The problem with CKAT

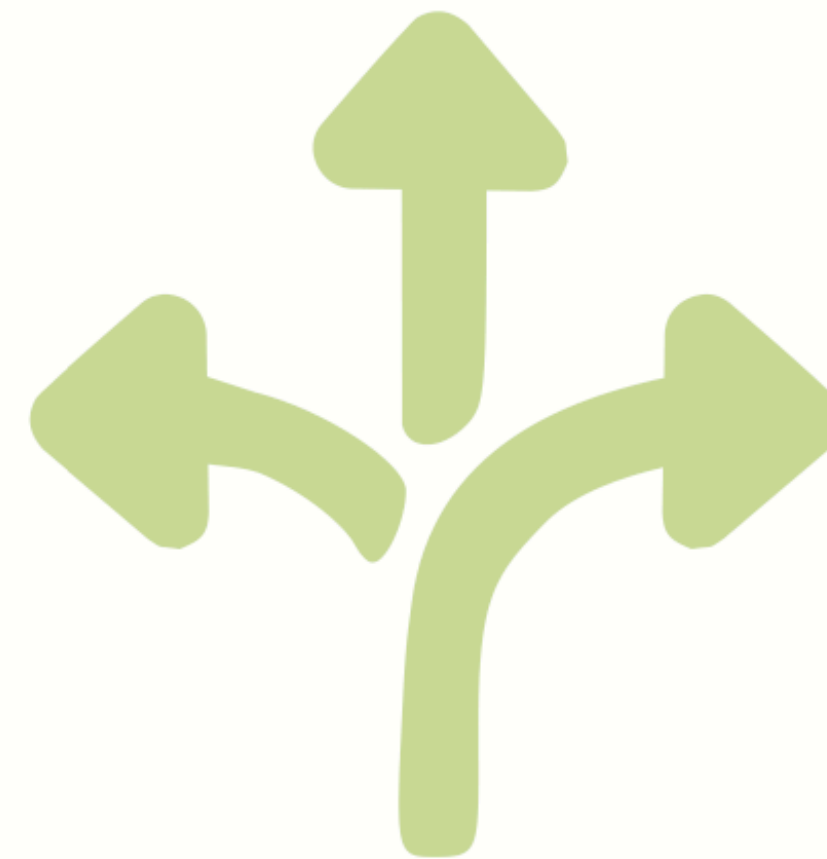
$$\begin{aligned} p \cdot e \cdot \bar{p} &\leq (p \parallel e) \cdot \bar{p} \\ &\equiv (p \parallel e) \cdot (\bar{p} \parallel 1) \\ &\leq (p \cdot \bar{p}) \parallel (e \cdot 1) \\ &\equiv (p \cdot \bar{p}) \parallel e \\ &\equiv (p \wedge \bar{p}) \parallel e \\ &\equiv \perp \parallel e \\ &\equiv 0 \parallel e \\ &\equiv 0 \end{aligned}$$

Every test is an
invariant of every
program!

Crossroads

Non-congruence?

Drop $e \parallel 0 \equiv 0$?



Drop exchange law?

Crossroads



See also [Bergstra and Ponse 2011].

Crossroads



See also [Bergstra and Ponse 2011].

Crossroads



See also [Bergstra and Ponse 2011].

Crossroads

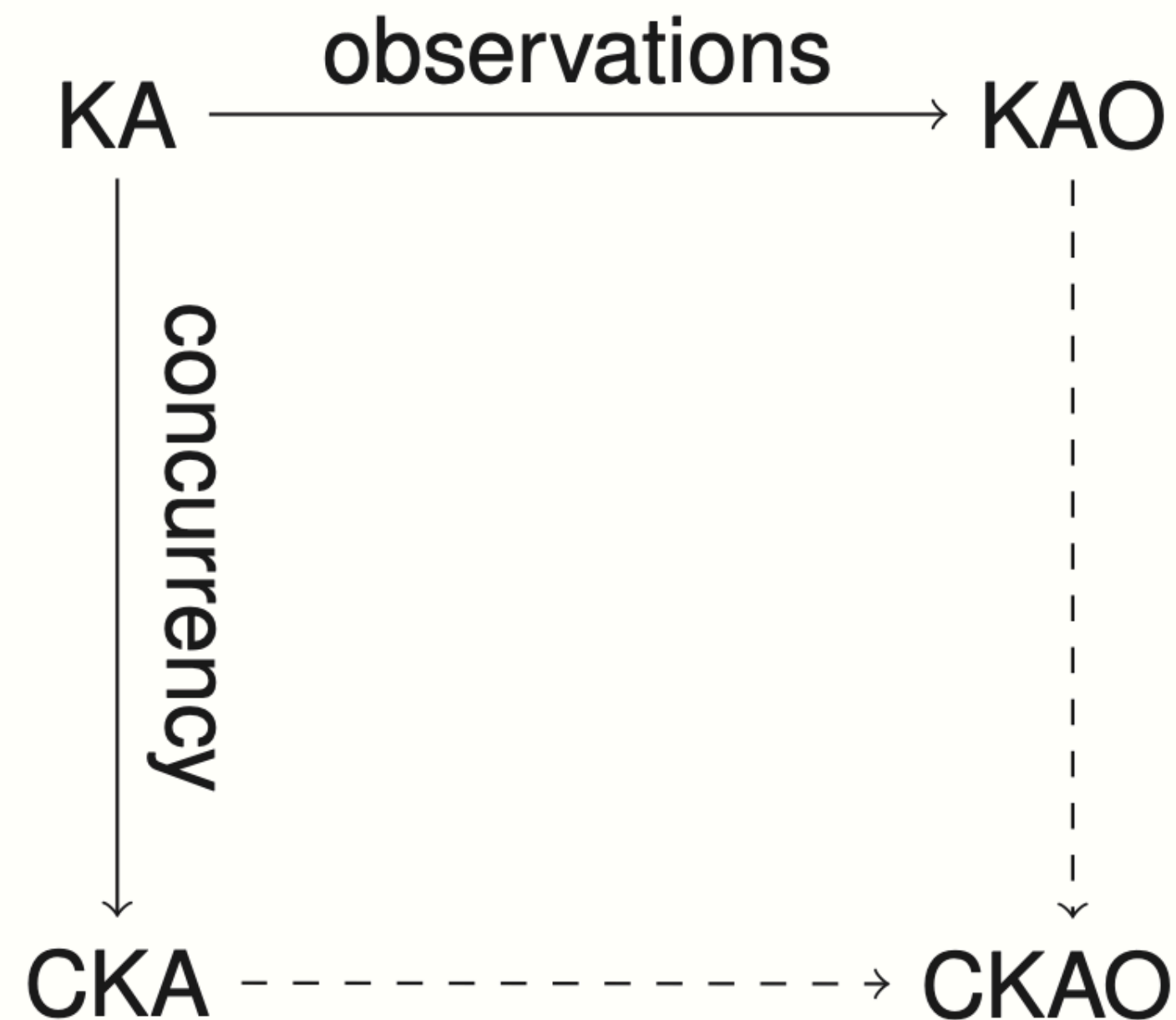


See also [Bergstra and Ponse 2011].

A new algebra

$$p \wedge q \equiv p \cdot q \rightsquigarrow p \wedge q \leq p \cdot q$$

$$1 \not\equiv \top$$



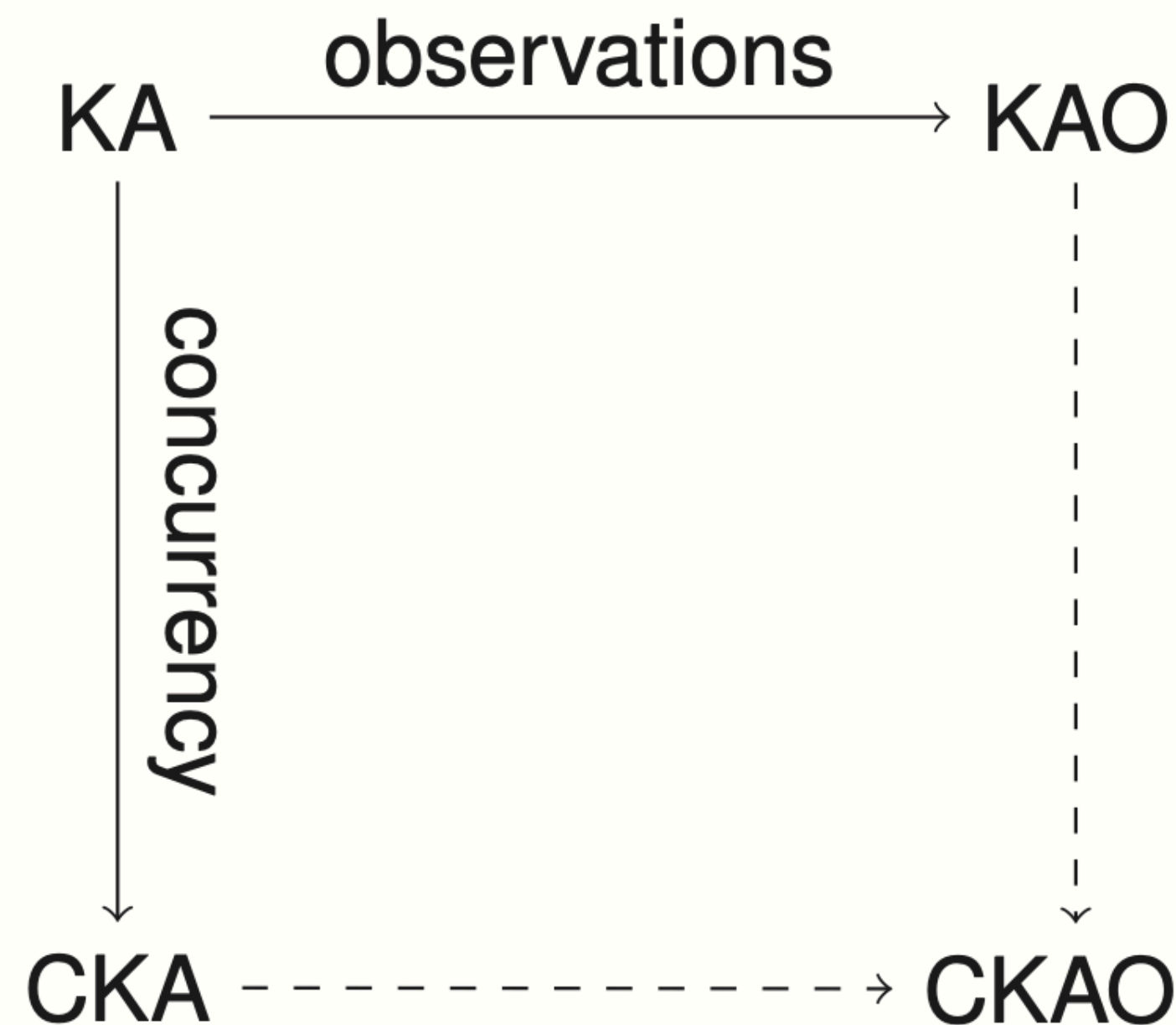
Axioms KA+BA+CKA

Keep conjunction
and sequential
composition distinct

A new algebra

$$p \wedge q \equiv p \cdot q \rightsquigarrow p \wedge q \leq p \cdot q$$

$$1 \not\equiv \top$$



Axioms KA+BA+CKA

Keep conjunction
and sequential
composition distinct

Algebra of global state

- ✓ Multiple Packets
 - ✓ Access to a variable store
- } **Concurrency**

State Fields	$\text{Var} \ni v ::= v_1 \mid \cdots \mid v_i$												
Global State	$\text{St} \ni \alpha, \beta ::= \text{Var} \rightarrow \text{Val}$												
State observations	$O \ni o, o' ::=$ <table><tr><td>$\perp$</td><td><i>Inconsistent</i></td></tr><tr><td>$\top$</td><td><i>Neutral</i></td></tr><tr><td>$v = n$</td><td><i>State test</i></td></tr><tr><td>$o \vee o'$</td><td><i>Union</i></td></tr><tr><td>$o \wedge o'$</td><td><i>Intersection</i></td></tr><tr><td>$\bar{o}$</td><td><i>Complement</i></td></tr></table>	$\perp$	<i>Inconsistent</i>	$\top$	<i>Neutral</i>	$v = n$	<i>State test</i>	$o \vee o'$	<i>Union</i>	$o \wedge o'$	<i>Intersection</i>	$\bar{o}$	<i>Complement</i>
$\perp$	<i>Inconsistent</i>												
$\top$	<i>Neutral</i>												
$v = n$	<i>State test</i>												
$o \vee o'$	<i>Union</i>												
$o \wedge o'$	<i>Intersection</i>												
$\bar{o}$	<i>Complement</i>												
State Actions	$\text{Act} \ni e ::=$ <table><tr><td>$v \leftarrow n$</td><td><i>Change</i></td></tr><tr><td>$v \leftarrow v'$</td><td><i>Copy</i></td></tr></table>	$v \leftarrow n$	<i>Change</i>	$v \leftarrow v'$	<i>Copy</i>								
$v \leftarrow n$	<i>Change</i>												
$v \leftarrow v'$	<i>Copy</i>												

Multiple packets and parallel processing

- ✓ Multiple Packets
 - ✓ Access to a variable store
- } **Concurrency**

$\text{Prg} \ni p, q ::=$	abort	<i>Abort</i>
	o	<i>State Obs</i>
	dup	<i>Duplicate</i>
	$p \parallel q$	<i>Parallel</i>
	skip	<i>Skip</i>
	$f \leftarrow n$	<i>Packet Action</i>
	$p + q$	<i>Choice</i>
	p^*	<i>Iteration</i>
	t	<i>Packet Filter</i>
	e	<i>State Action</i>
	$p ; q$	<i>Sequence</i>
	a	<i>Packet Sets</i>

Multiple packets and parallel processing

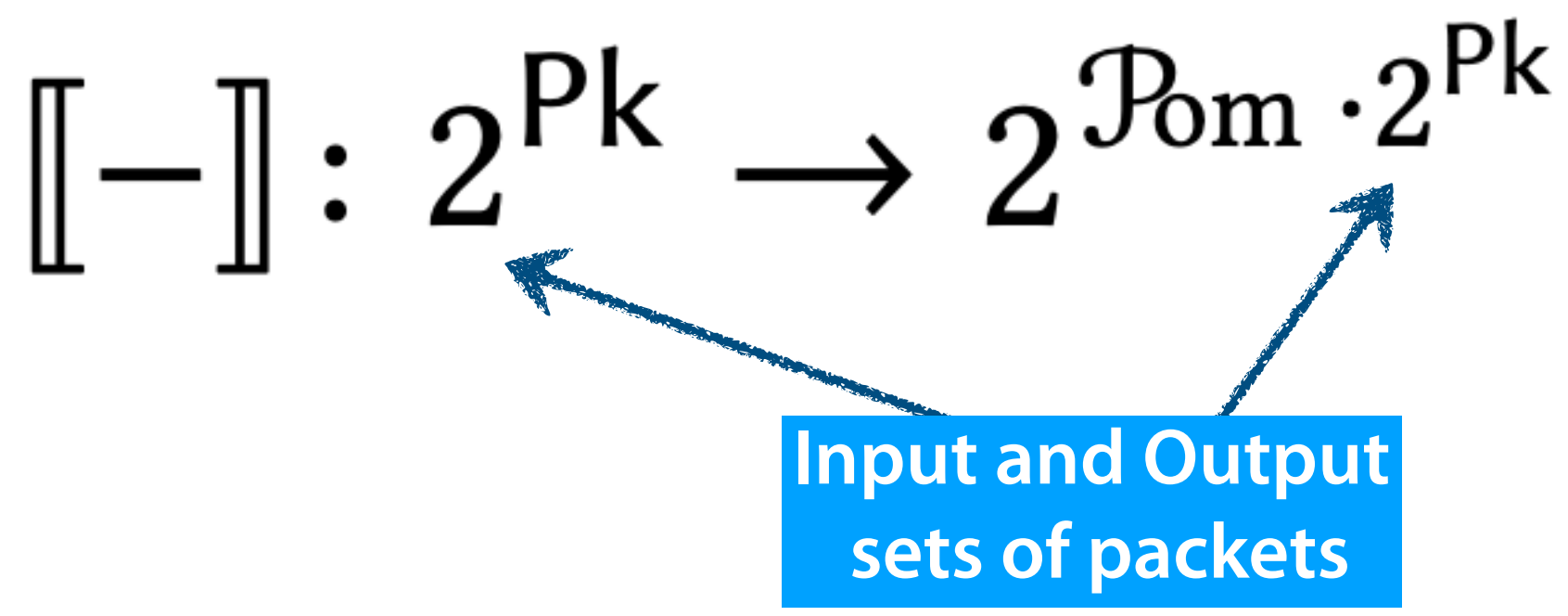
- ✓ Multiple Packets
 - ✓ Access to a variable store
- } **Concurrency**

$$\llbracket - \rrbracket : 2^{P_k} \rightarrow 2^{\mathcal{P}_{om}} \cdot 2^{P_k}$$

Prg $\ni p, q ::=$		abort	<i>Abort</i>
	<i>o</i>		<i>State Obs</i>
	dup		<i>Duplicate</i>
	<i>p q</i>		<i>Parallel</i>
	skip		<i>Skip</i>
	<i>f</i> $\leftarrow n$		<i>Packet Action</i>
	<i>p + q</i>		<i>Choice</i>
	<i>p</i> [*]		<i>Iteration</i>
	<i>t</i>		<i>Packet Filter</i>
	<i>e</i>		<i>State Action</i>
	<i>p ; q</i>		<i>Sequence</i>
	<i>a</i>		<i>Packet Sets</i>

Multiple packets and parallel processing

- ✓ Multiple Packets
 - ✓ Access to a variable store
- } **Concurrency**



$\text{Prg} \ni p, q ::=$	abort	<i>Abort</i>
	o	<i>State Obs</i>
	dup	<i>Duplicate</i>
	$p \parallel q$	<i>Parallel</i>
	skip	<i>Skip</i>
	$f \leftarrow n$	<i>Packet Action</i>
	$p + q$	<i>Choice</i>
	p^*	<i>Iteration</i>
	t	<i>Packet Filter</i>
	e	<i>State Action</i>
	$p ; q$	<i>Sequence</i>
	a	<i>Packet Sets</i>

Multiple packets and parallel processing

- ✓ Multiple Packets
 - ✓ Access to a variable store
- } **Concurrency**

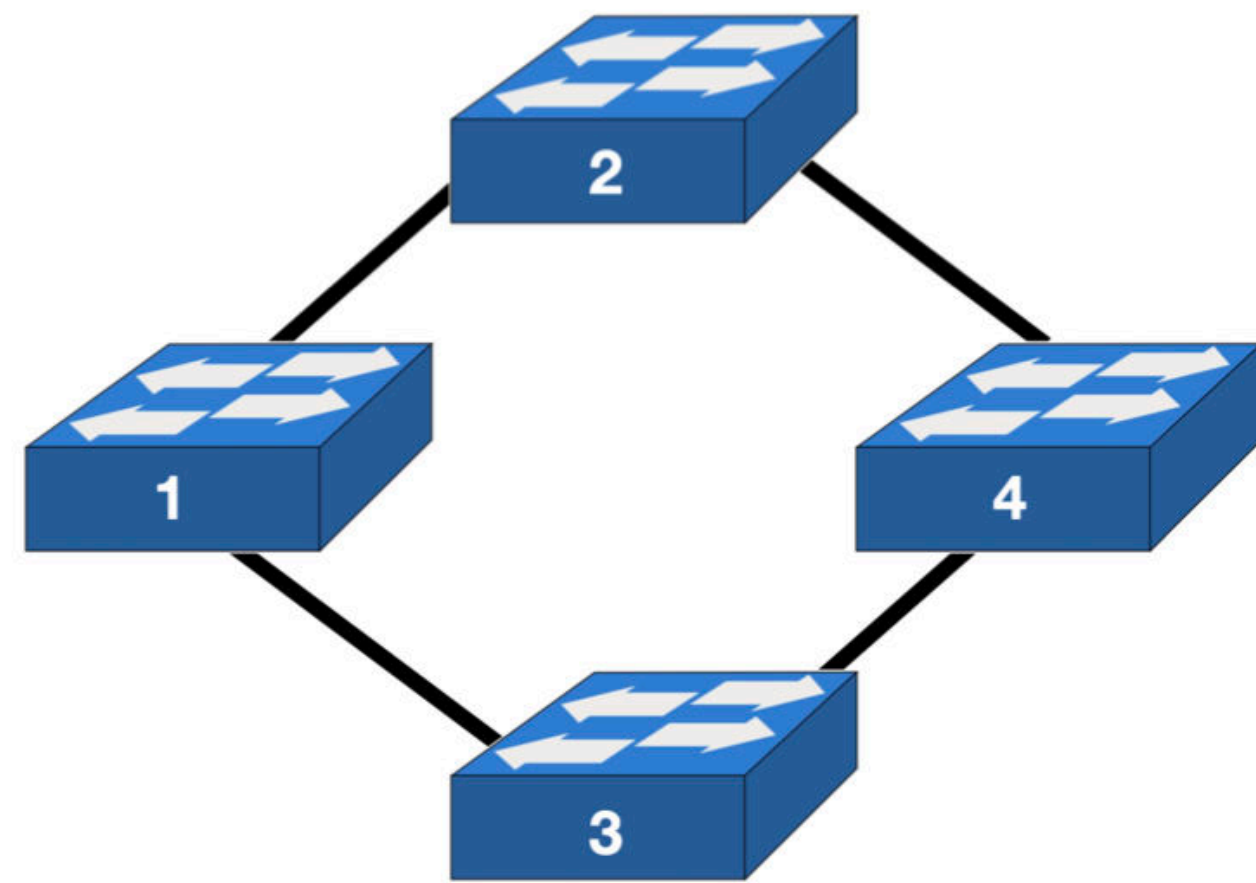
Pomset of global state evolution

$$[-] : 2^{P_k} \rightarrow 2^{\mathcal{P}_{om}} \cdot 2^{P_k}$$

Input and Output
sets of packets

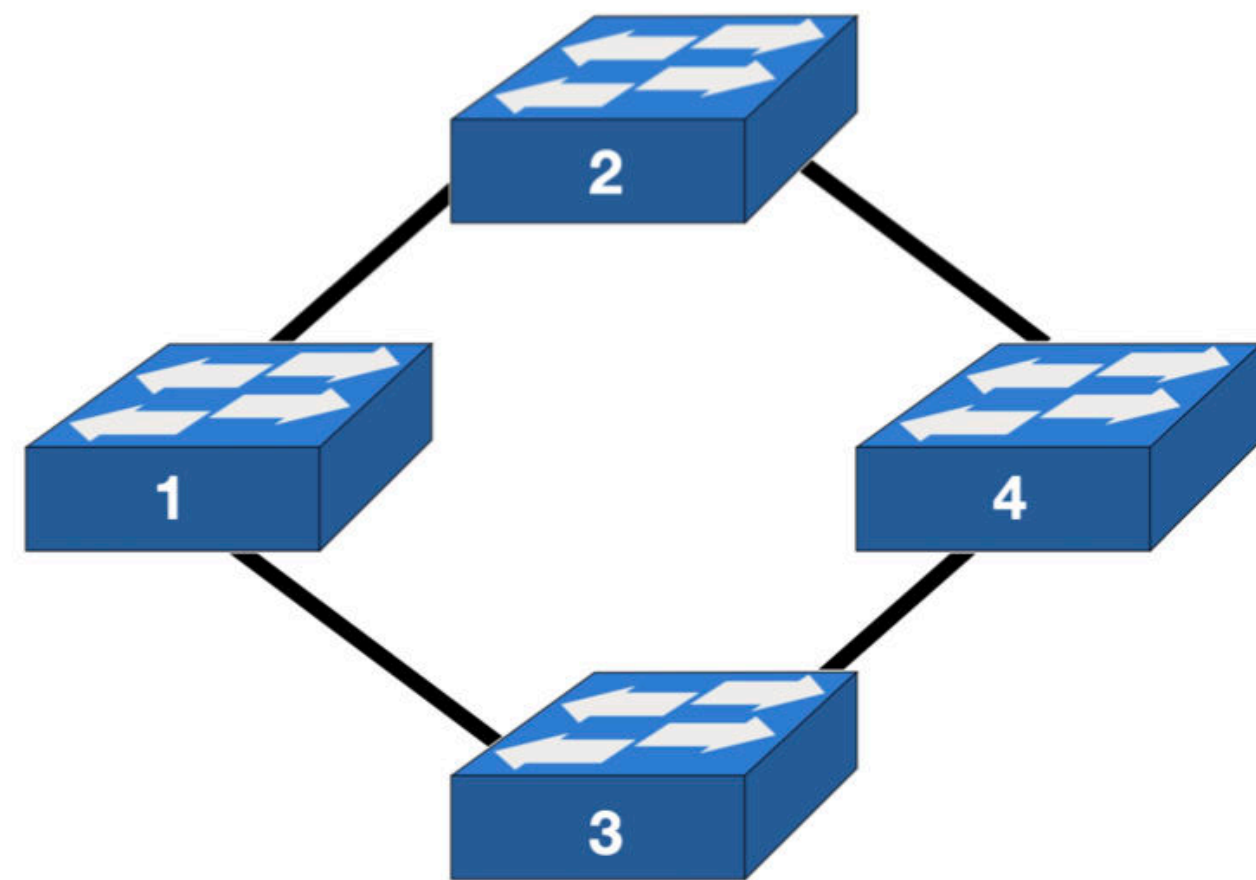
Prg $\ni p, q ::=$		abort	<i>Abort</i>
	<i>o</i>		<i>State Obs</i>
	dup		<i>Duplicate</i>
	<i>p q</i>		<i>Parallel</i>
	skip		<i>Skip</i>
	<i>f</i> $\leftarrow n$		<i>Packet Action</i>
	<i>p + q</i>		<i>Choice</i>
	<i>p</i> [*]		<i>Iteration</i>
	<i>t</i>		<i>Packet Filter</i>
	<i>e</i>		<i>State Action</i>
	<i>p ; q</i>		<i>Sequence</i>
	<i>a</i>		<i>Packet Sets</i>

Revisiting the example



$p_1 \triangleq \text{sw} = 1 ; ((v = 1 ; \text{tag} = \spadesuit ; \text{sw} \leftarrow 2) \parallel (\text{tag} = \heartsuit ; \text{sw} \leftarrow 3 ; v \leftarrow 1))$
 $p_2 \triangleq \text{sw} = 2 ; \text{sw} \leftarrow 4$
 $p_3 \triangleq \text{sw} = 3 ; \text{sw} \leftarrow 4$
 $p_4 \triangleq \text{sw} = 4$
 $p \triangleq v \leftarrow 0 ; (p_1 \parallel p_2 \parallel p_3 \parallel p_4)^*$

Revisiting the example

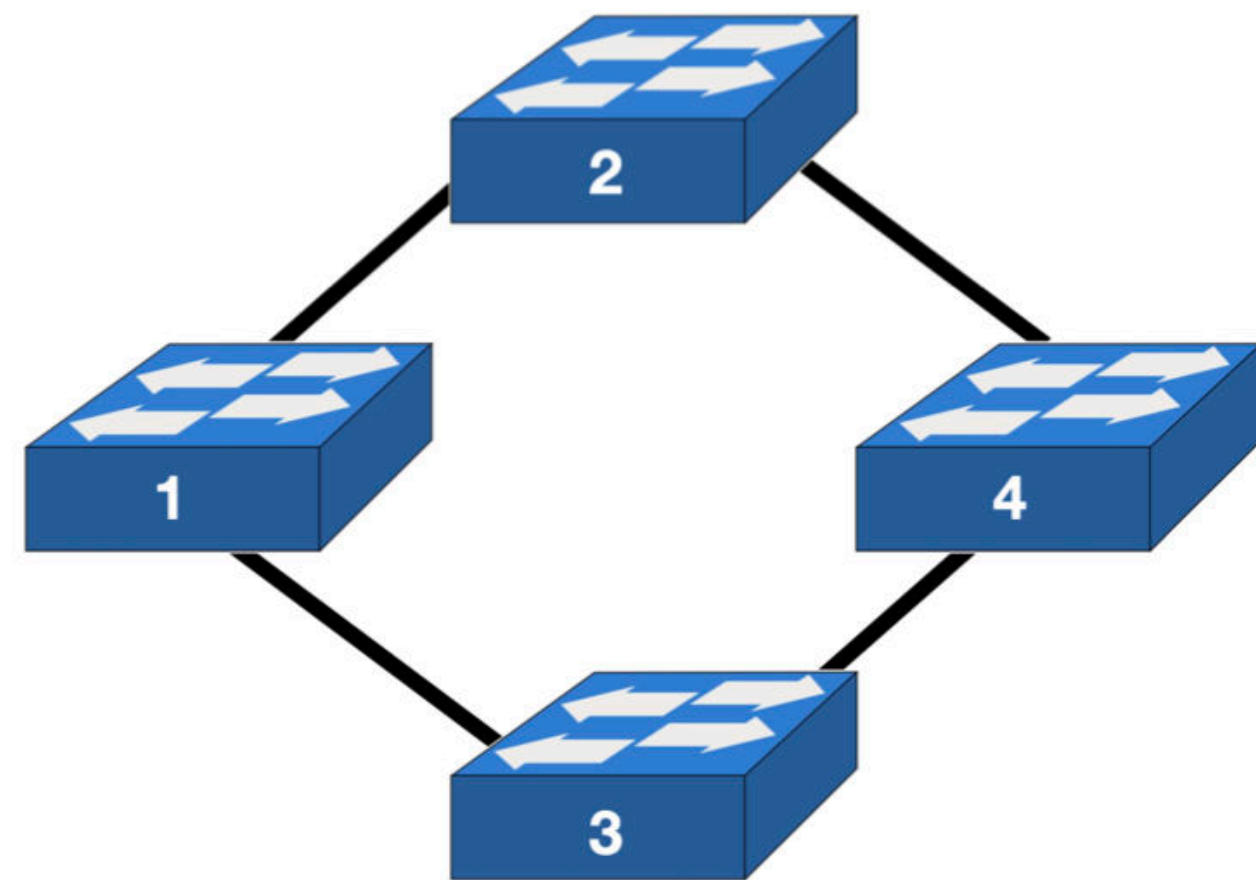


$p_1 \triangleq \text{sw} = 1 ; ((v = 1 ; \text{tag} = \spadesuit ; \text{sw} \leftarrow 2) \parallel (\text{tag} = \heartsuit ; \text{sw} \leftarrow 3 ; v \leftarrow 1))$
 $p_2 \triangleq \text{sw} = 2 ; \text{sw} \leftarrow 4$
 $p_3 \triangleq \text{sw} = 3 ; \text{sw} \leftarrow 4$
 $p_4 \triangleq \text{sw} = 4$

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

Do I always see a green packet in 3 before I see a red packet in 2?

Revisiting the example


$$p_1 \triangleq \text{sw} = 1 ; ((v = 1 ; \text{tag} = \spadesuit ; \text{sw} \leftarrow 2) \parallel (\text{tag} = \heartsuit ; \text{sw} \leftarrow 3 ; v \leftarrow 1))$$
$$p_2 \triangleq \text{sw} = 2; \text{sw} \leftarrow 4$$
$$p_3 \triangleq \text{sw} = 3; \text{sw} \leftarrow 4$$
$$p_4 \triangleq \text{sw} = 4$$
$$p \triangleq v \leftarrow 0 ; (p_1 \parallel p_2 \parallel p_3 \parallel p_4)^*$$

$$\llbracket p \rrbracket(\{\heartsuit, \spadesuit\}) \ni \begin{array}{l} (v \leftarrow 0) \rightarrow \{\heartsuit, \spadesuit\} \rightarrow \{\heartsuit\} \rightarrow \{\heartsuit[3/\text{sw}]\} \rightarrow (v \leftarrow 1) \rightarrow \beta \rightarrow \cdots \\ \cdots \rightarrow \{\spadesuit\} \rightarrow \{\spadesuit[2/\text{sw}]\} \rightarrow \begin{array}{l} \{\spadesuit[2/\text{sw}]\} \rightarrow \{\spadesuit[4/\text{sw}]\} \rightarrow \\ \{\heartsuit[3/\text{sw}]\} \rightarrow \{\heartsuit[4/\text{sw}]\} \rightarrow \end{array} \{\spadesuit[4/\text{sw}], \heartsuit[4/\text{sw}]\} \end{array}$$

Multiple packets : non-determinism vs multicast



$$f = n + g = m$$

Multiple packets : non-determinism vs multicast



$$f = n + g = m$$



Multiple packets : non-determinism vs multicast



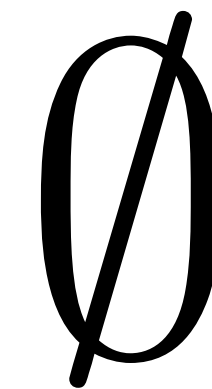
$$f = n + g = m$$



Multiple packets : non-determinism vs multicast



$$f = n + g = m$$



Multiple packets : non-determinism vs multicast



$$f = n + g = m$$

Multiple packets : non-determinism vs multicast



$$f = n + g = m$$



Multiple packets : non-determinism vs multicast



$$f = n + g = m$$



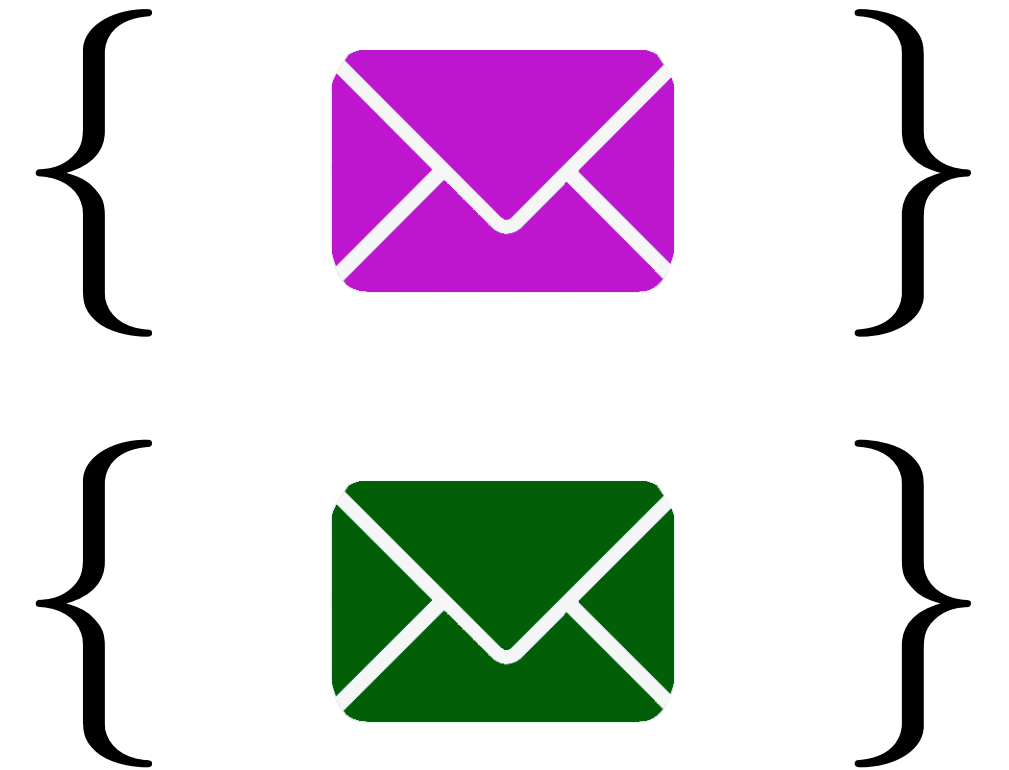
Multiple packets : non-determinism vs multicast

$$f = n + g = m$$



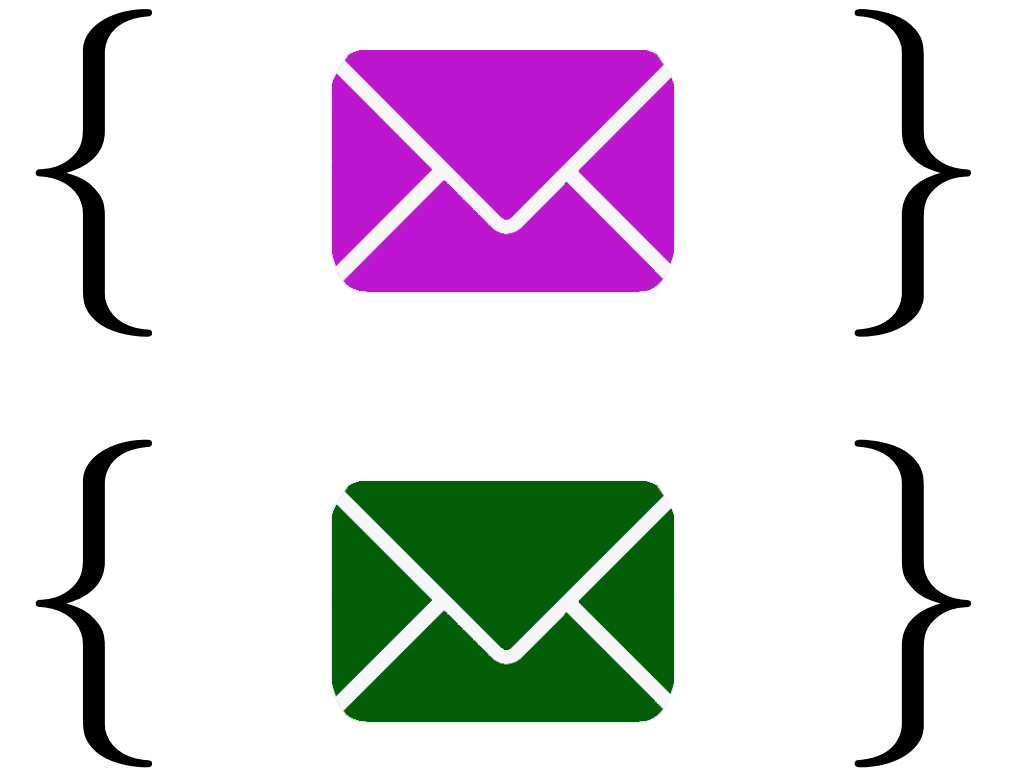
Multiple packets : non-determinism vs multicast

$$f = n + g = m$$



Multiple packets : non-determinism vs multicast

$$f = n + g = m$$



$$f = n \parallel g = m$$



CNetKAT axioms

Boolean Algebra

PCDL (pseudo-complement distributive lattice)

+

Kleene Algebra

+

Packet Axioms

+

Parallel Axioms

+

Interface axioms

CNetKAT axioms

~~Boolean Algebra~~

PCDL (pseudo-complement distributive lattice)

+

Kleene Algebra

+

Packet Axioms

+

Parallel Axioms

+

Interface axioms

CNetKAT axioms

~~Boolean Algebra~~

PCDL (pseudo-complement distributive lattice)

+

Kleene Algebra

+

Packet Axioms

+

Parallel Axioms

+

Interface axioms

$$p \parallel (q \parallel r) \equiv (p \parallel q) \parallel r$$

$$p \parallel \text{abort} \equiv \text{abort}$$

$$\text{drop} \parallel p \equiv p$$

$$p \parallel (q + r) \equiv p \parallel q + p \parallel r$$

$$p \parallel q \equiv q \parallel p$$

CNetKAT axioms

~~Boolean Algebra~~

PCDL (pseudo-complement distributive lattice)

+

Kleene Algebra

+

Packet Axioms

+

Parallel Axioms

+

Interface axioms

$$p \parallel (q \parallel r) \equiv (p \parallel q) \parallel r$$

$$\underline{p \parallel \text{abort}} \equiv \text{abort}$$

$$\text{drop} \parallel p \equiv p$$

$$p \parallel (q + r) \equiv p \parallel q + p \parallel r$$

$$p \parallel q \equiv q \parallel p$$

CNetKAT axioms

~~Boolean Algebra~~

PCDL (pseudo-complement distributive lattice)

+

Kleene Algebra

+

Packet Axioms

+

Parallel Axioms

+

Interface axioms

$$p \parallel (q \parallel r) \equiv (p \parallel q) \parallel r$$

$$\underline{p \parallel \text{abort}} \equiv \text{abort}$$

$$\underline{\text{drop} \parallel p} \equiv p$$

$$p \parallel (q + r) \equiv p \parallel q + p \parallel r$$

$$p \parallel q \equiv q \parallel p$$

Soundness and Completeness

Soundness

For all $p, q \in \text{Prg}$, if $p \equiv q$, then $\llbracket p \rrbracket \downarrow = \llbracket q \rrbracket \downarrow$.

Completeness

Let $p, q \in \text{Prg}$ and $a \in 2^{\text{Pk}}$. If $\llbracket p \rrbracket \downarrow (a) = \llbracket q \rrbracket \downarrow (a)$, then $\Pi_a ; p \equiv \Pi_a ; q$.

Final Remarks

- ✓ Language to specify and verify network behaviour
- ✓ Compositional semantics
- ✓ Multiple Packets
- ✓ Access to a variable store

Final Remarks

- NetKAT** {
- ✓ Language to specify and verify network behaviour
 - ✓ Compositional semantics
 - ✓ Multiple Packets
 - ✓ Access to a variable store

Final Remarks

- NetKAT** {
- ✓ Language to specify and verify network behaviour
 - ✓ Compositional semantics
 - ✓ Multiple Packets
 - ✓ Access to a variable store
- } **Concurrency**

Final Remarks

- NetKAT** {
- ✓ Language to specify and verify network behaviour
 - ✓ Compositional semantics
 - ✓ Multiple Packets
 - ✓ Access to a variable store
- } **Concurrency**

- ❖ Concurrency is hard
- ❖ Global vs local state makes reasoning subtle
- ❖ Encoding examples can be tricky

Final Remarks

- NetKAT** {
- ✓ Language to specify and verify network behaviour
 - ✓ Compositional semantics
 - ✓ Multiple Packets
 - ✓ Access to a variable store
- } **Concurrency**

- ❖ Concurrency is hard
- ❖ Global vs local state makes reasoning subtle
- ❖ Encoding examples can be tricky

- ➡ Case studies
- ➡ Better reasoning support
- ➡ Decision procedure
- ➡ Library of theorems