

Algebraic Network Verification

Alexandra Silva

A Team Effort



Carolyn Anderson
Wellesley College



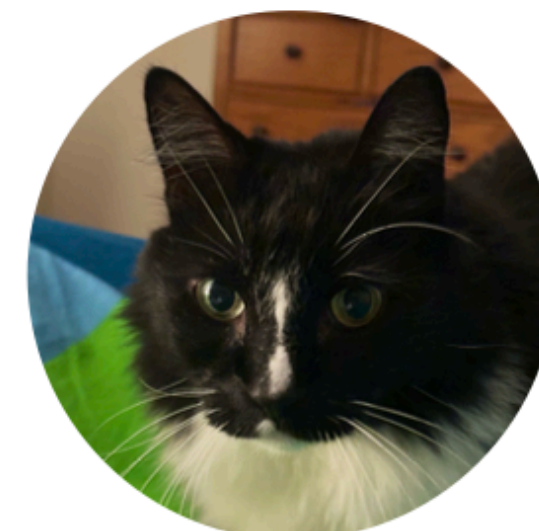
Mark Barbone
Cornell



Olivier Savary Belanger
Galois



Eric Hayden Campbell
UT Austin



Carbon
Chez Foster



David Darais
Galois



Arjun Guha
Northeastern



Jules Jacobs
Cornell



Tobias Kappé
Leiden



Dexter Kozen
Cornell



Mae Milano
Princeton



Mark Moeller
Cornell



Mark Reitblatt
Cornell



Cole Schlesinger
Galois



Alexandra Silva
Cornell



Steffen Smolka
Google



Laure Thompson
Princeton



Jana Wagemaker
Nijmegen



Dave Walker
Princeton

A Team Effort

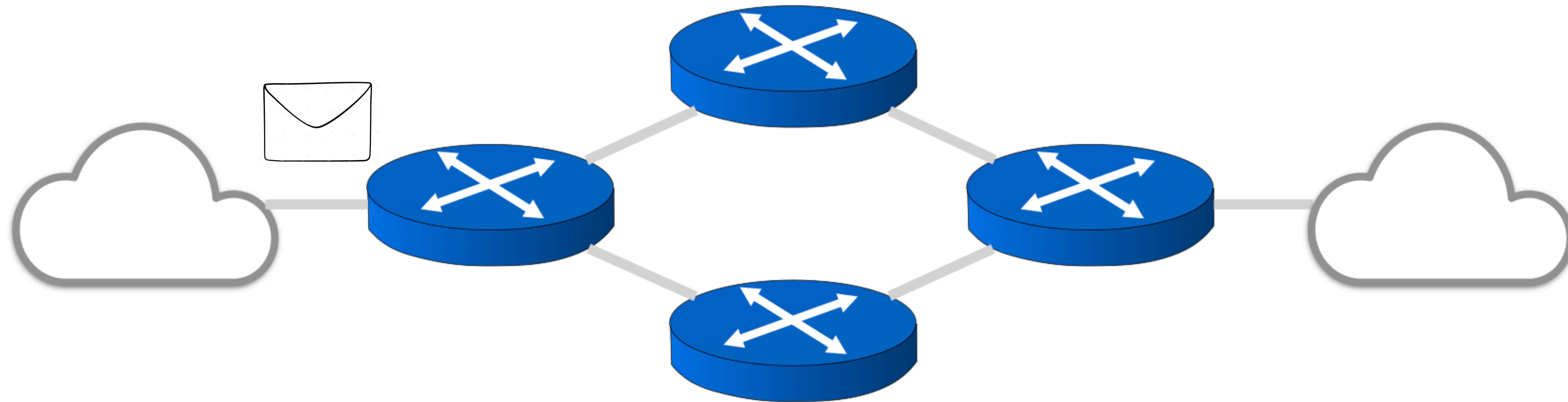


<https://netkat.org/>

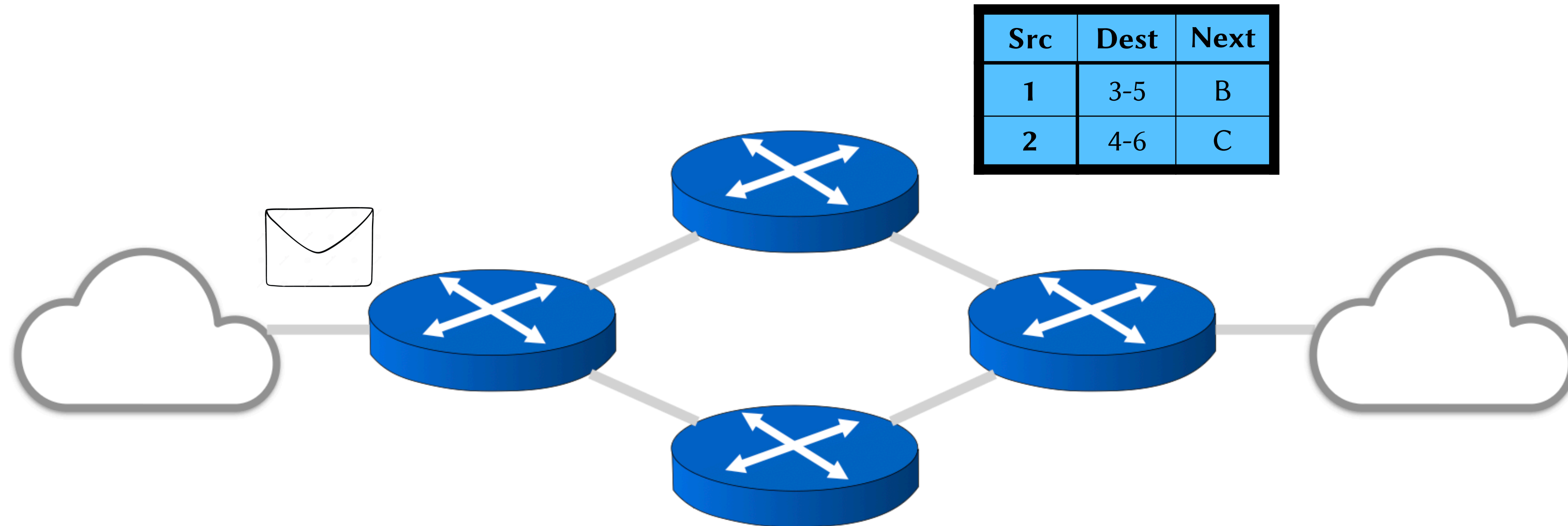
Plan

- Today: NetKAT (syntax and semantics)
- Tomorrow: Decision Procedures & Applications

Verification in Network Routing Policies



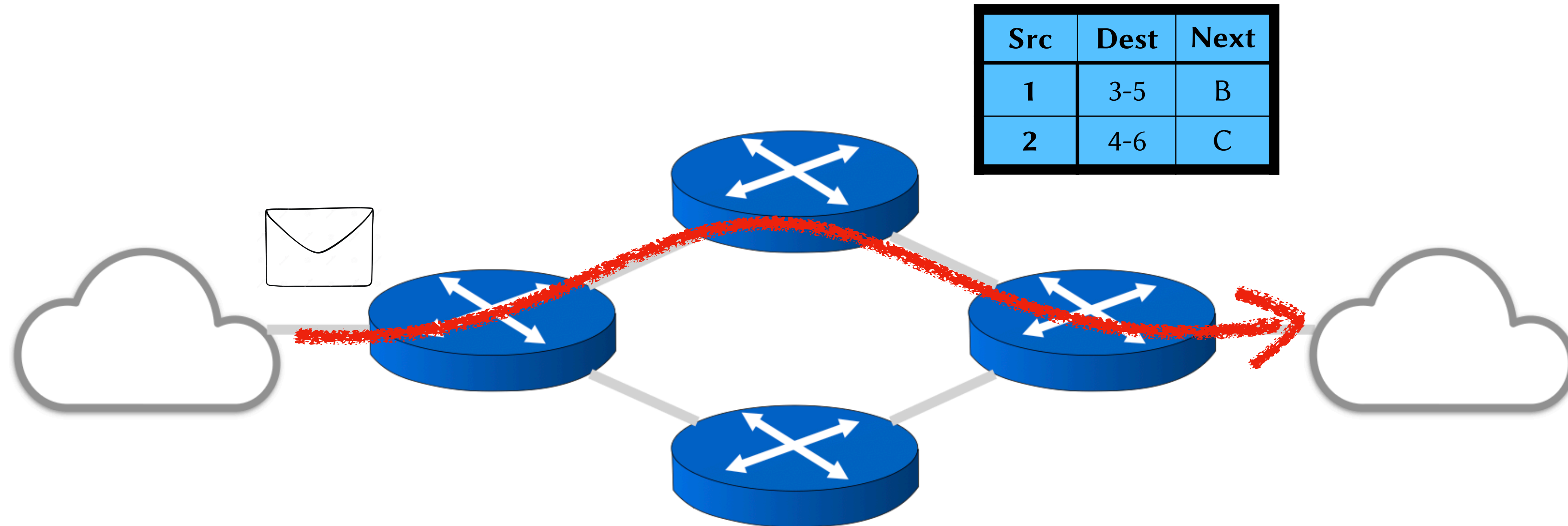
Verification in Network Routing Policies



Each table contains rules that:

- **Match** on packet headers
- Execute **Actions** that transform, forward, or drop packets

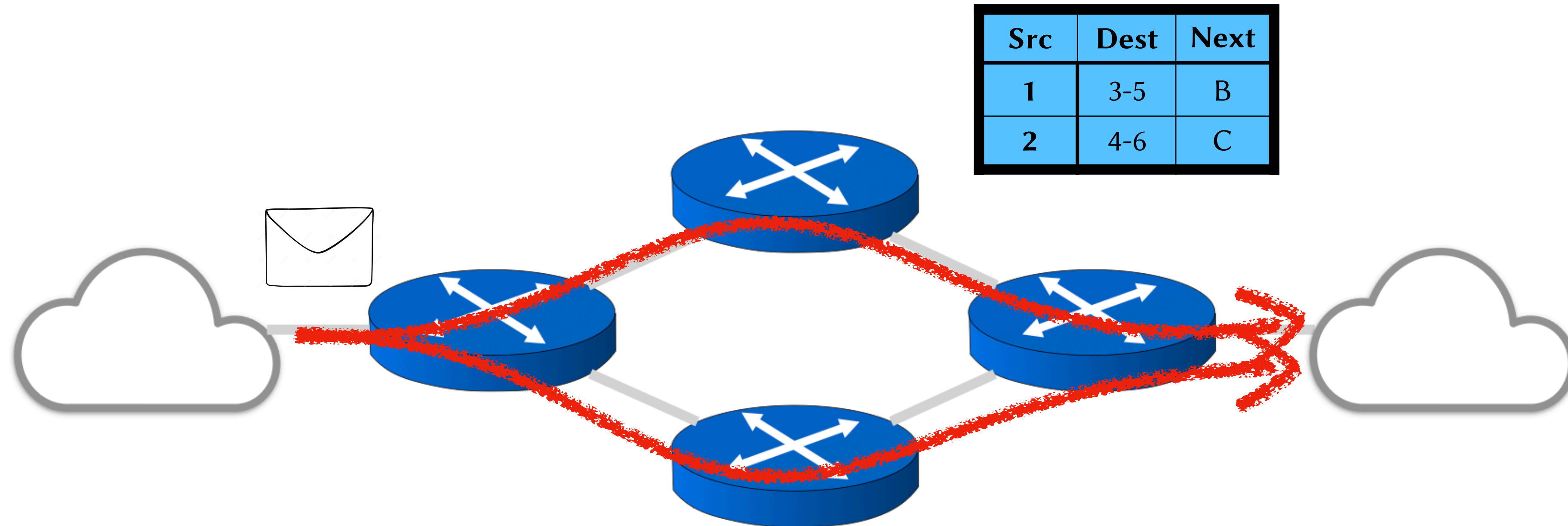
Verification in Network Routing Policies



Each table contains rules that:

- **Match** on packet headers
- Execute **Actions** that transform, forward, or drop packets

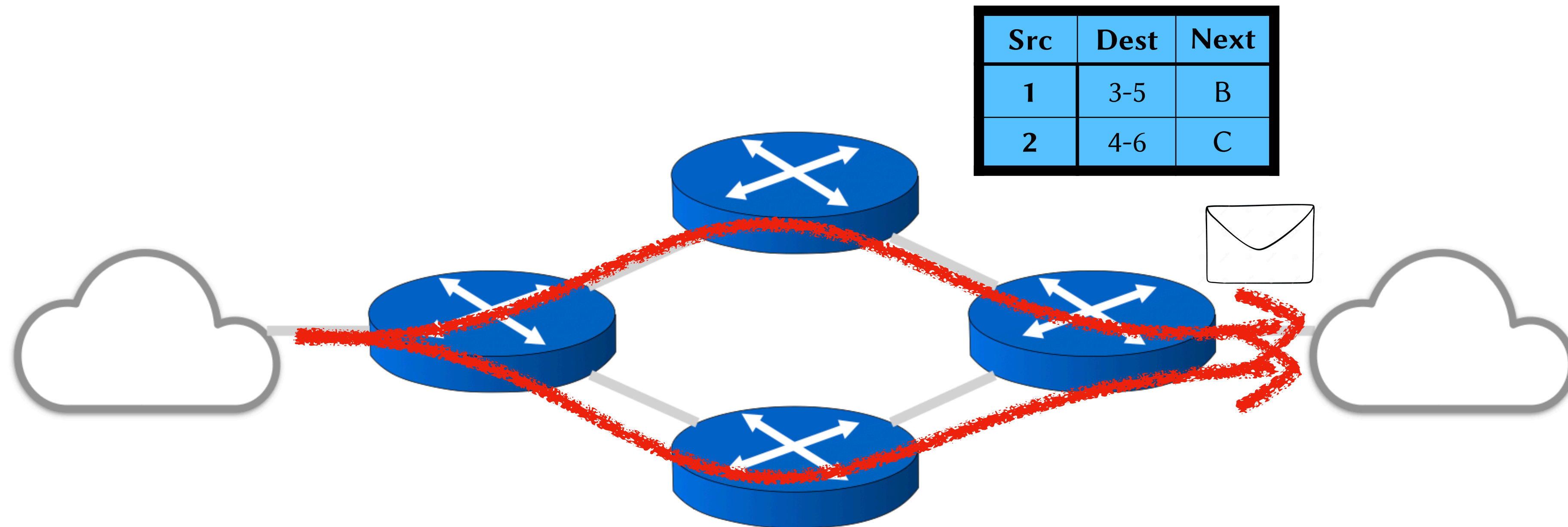
Verification in Network Routing Policies



Each table contains rules that:

- **Match** on packet headers
- Execute **Actions** that transform, forward, or drop packets

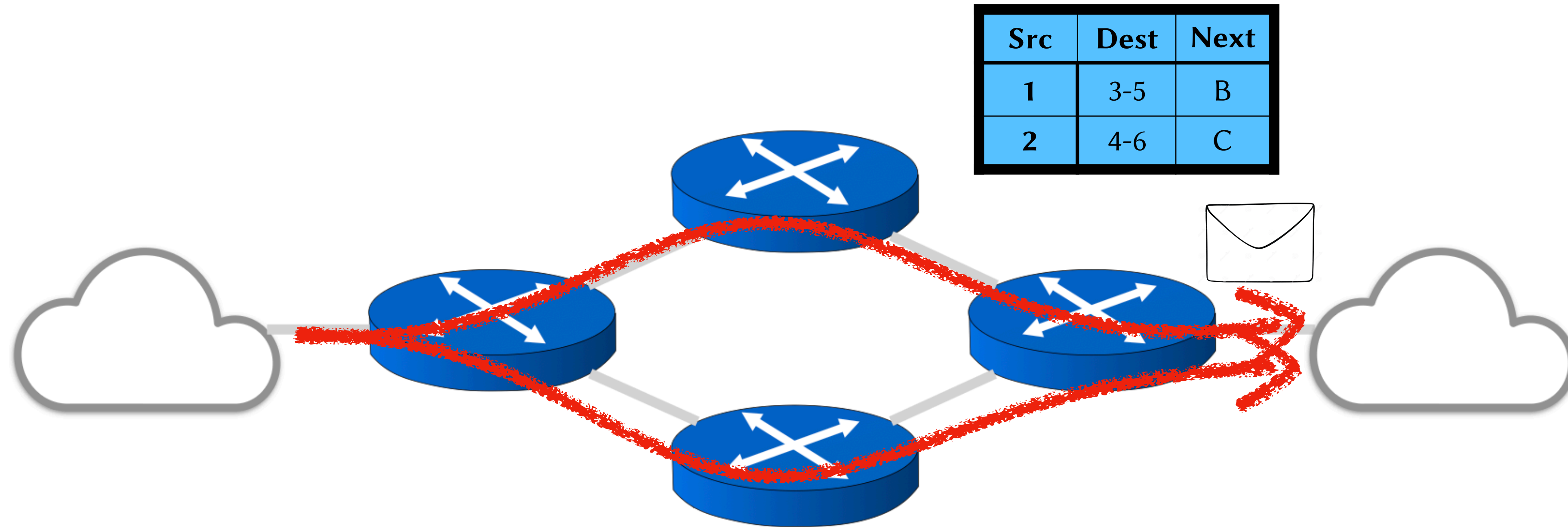
Verification in Network Routing Policies



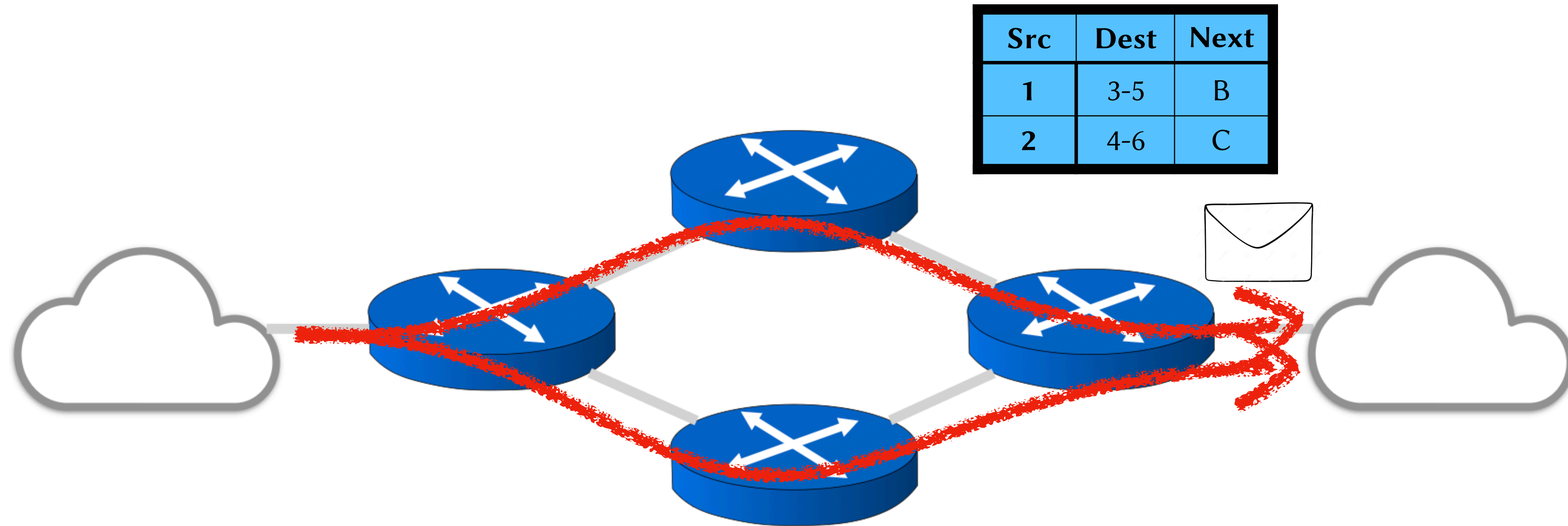
Each table contains rules that:

- **Match** on packet headers
- Execute **Actions** that transform, forward, or drop packets

Verification in Network Routing Policies

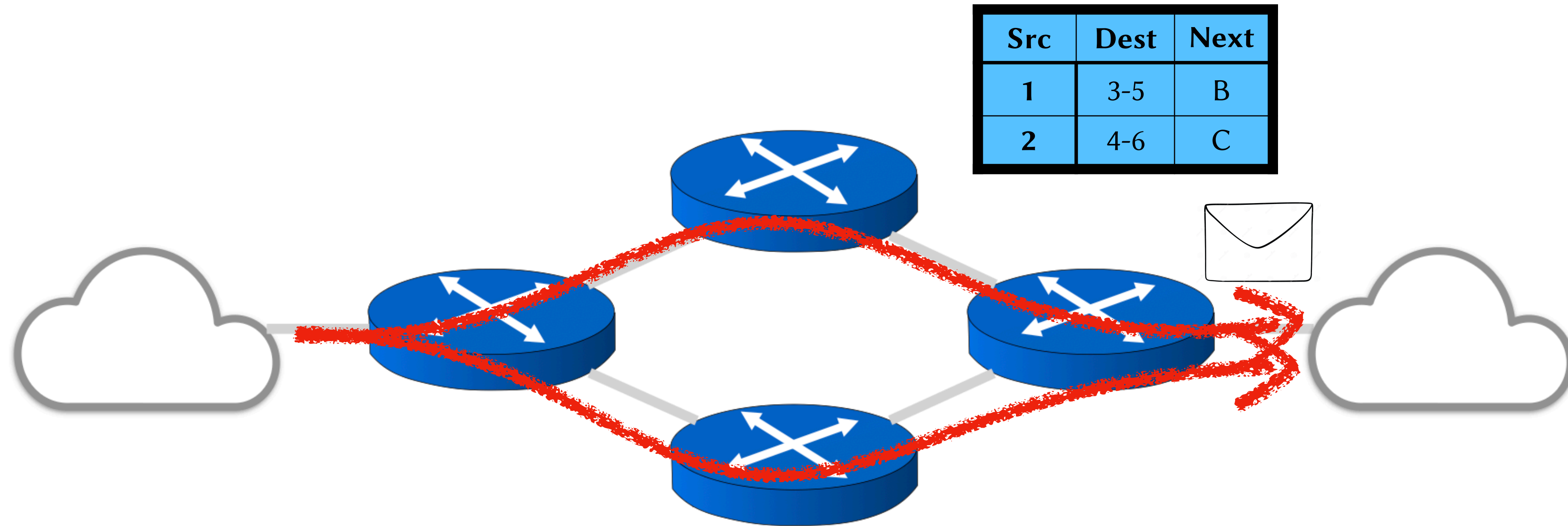


Verification in Network Routing Policies



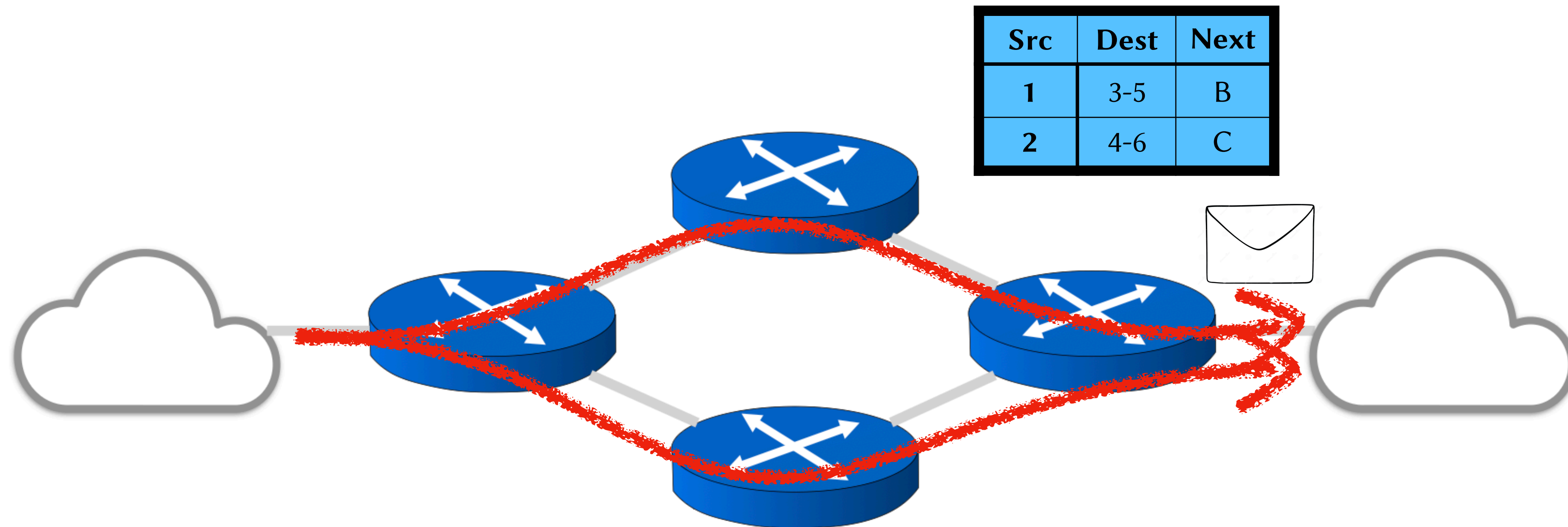
Can a packet travel from A to B?

Verification in Network Routing Policies



Can a packet travel from A to B?
Are all hosts reachable from every other hosts?

Verification in Network Routing Policies

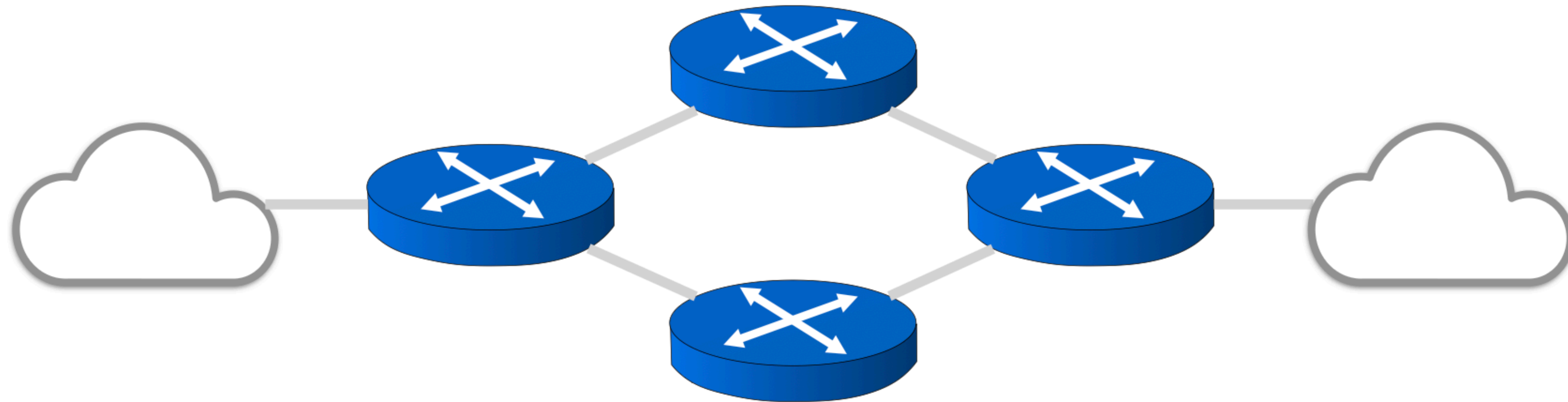


Can a packet travel from A to B?

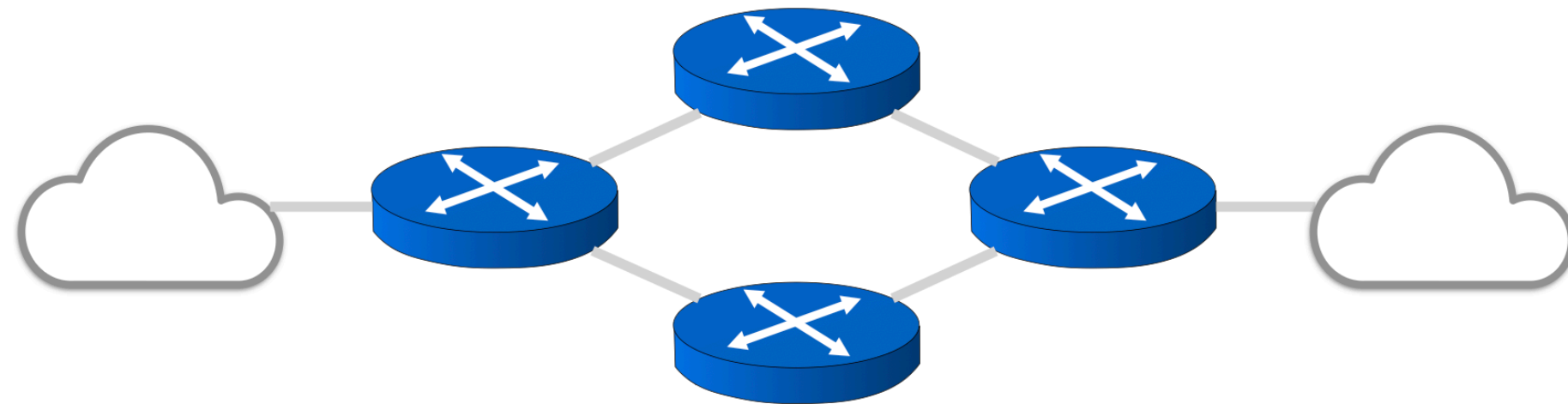
Are all hosts reachable from every other hosts?

Is subnet X isolated from subnet Y?

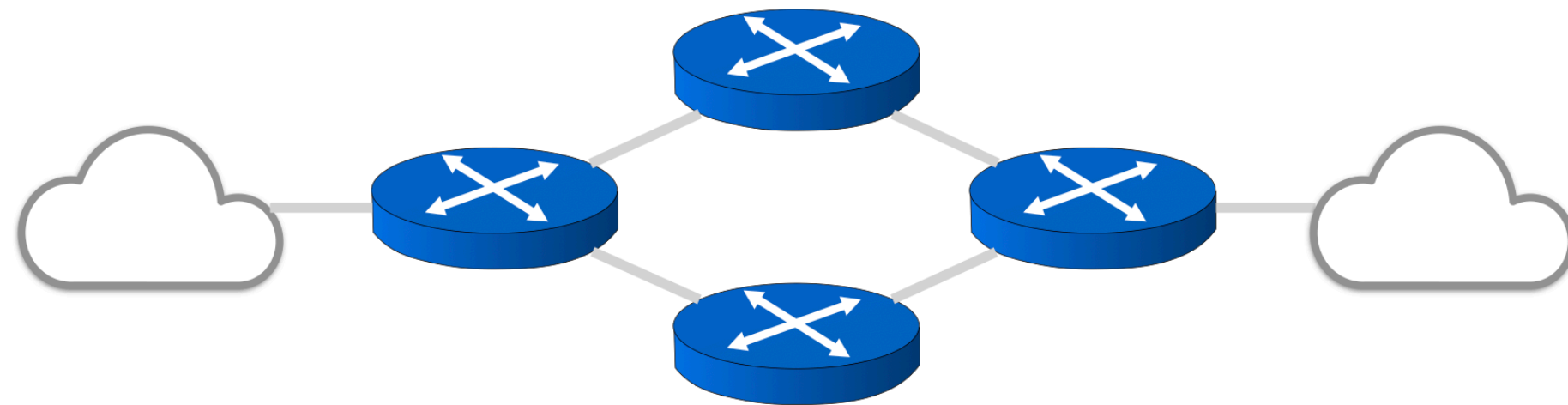
Verification in Network Routing Policies



Verification in Network Routing Policies

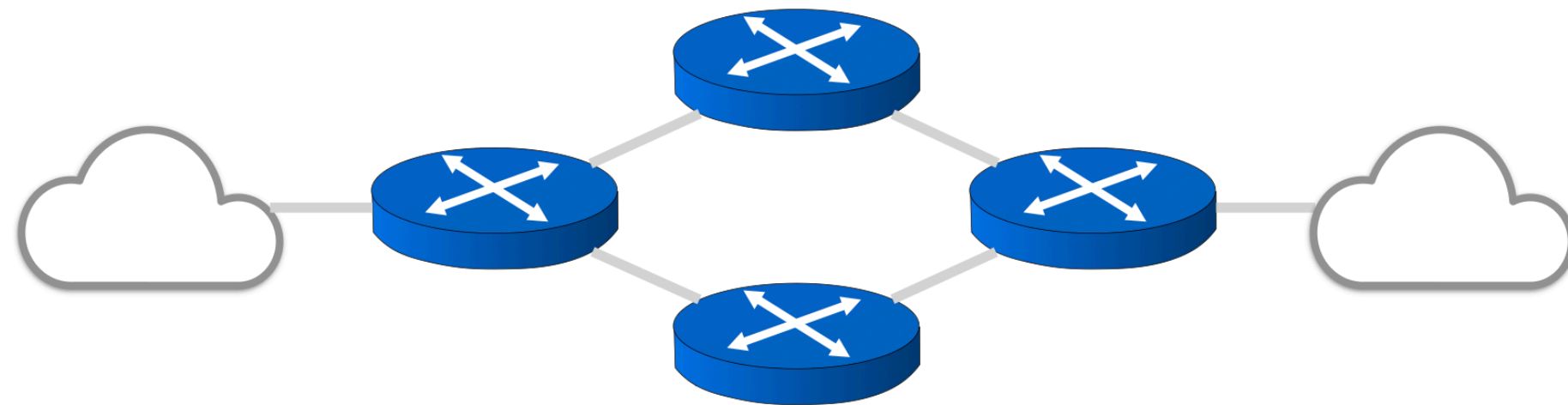


Verification in Network Routing Policies



Topology + configurations

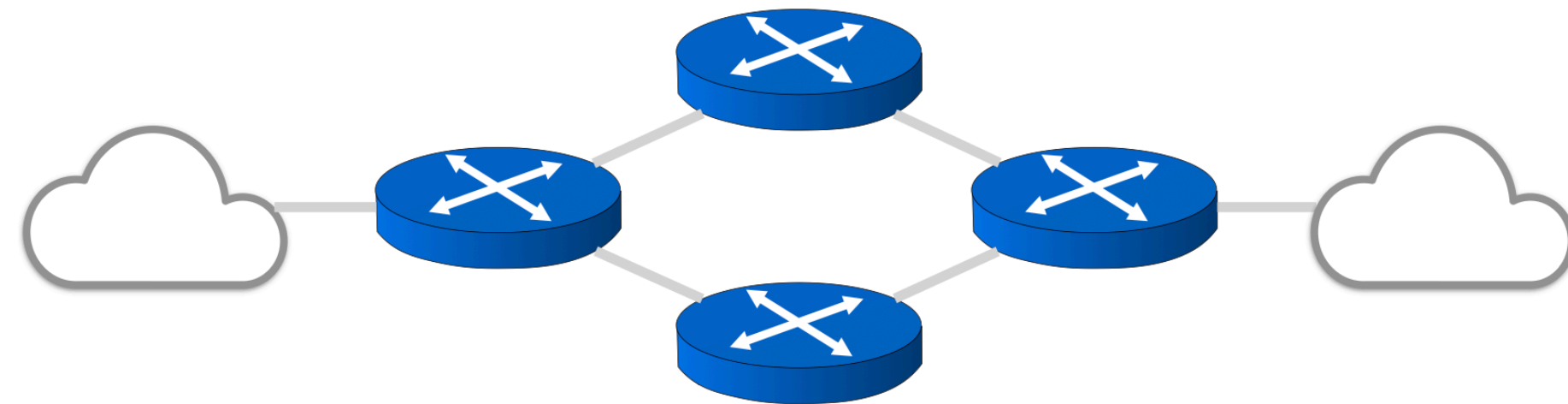
Verification in Network Routing Policies



Topology + configurations

Spec: *The network should be free of forwarding loops, and every packet should be delivered to its destination*

Verification in Network Routing Policies



Topology + configurations

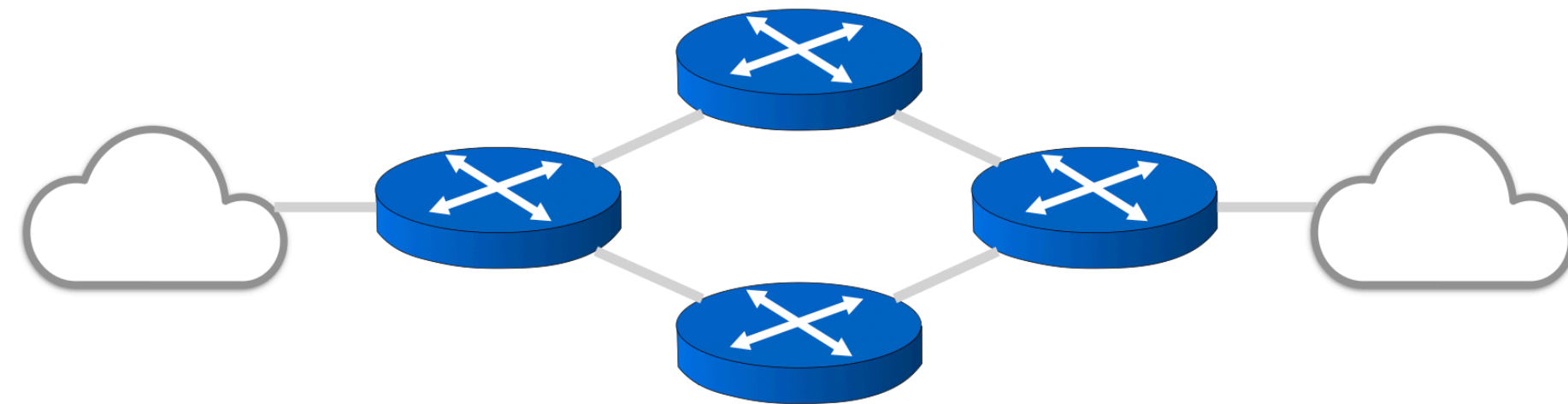
Spec: *The network should be free of forwarding loops, and every packet should be delivered to its destination*

Φ

Solver

Ψ

Verification in Network Routing Policies



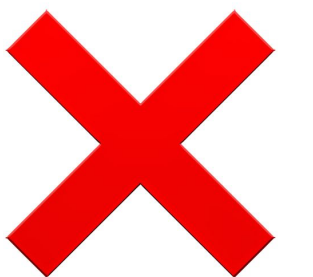
Topology + configurations

Φ

Solver

Spec: *The network should be free of forwarding loops, and every packet should be delivered to its destination*

Ψ

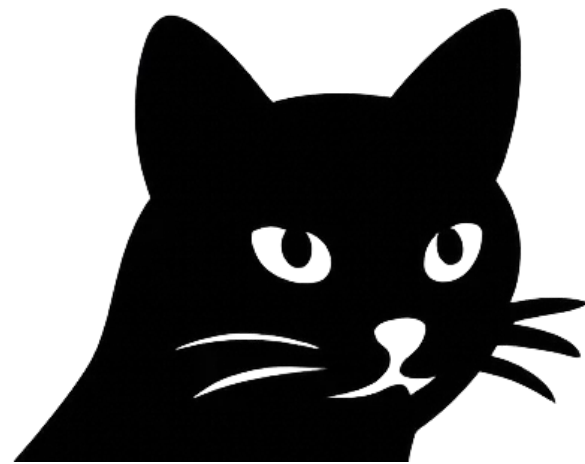


Challenging the status quo

Domain-specific languages *based on algebraic foundations* have a key role to play in formal verification of systems

Challenging the status quo

Domain-specific languages *based on algebraic foundations* have a key role to play in formal verification of systems



Design **high-level languages** that model essential network features
Develop **semantics** that enables reasoning precisely about behaviour
Build **tools** to synthesise low-level implementations automatically

What are essential network features?

What are essential network features?

Nodes



What are essential network features?

Nodes

Forwarding Policies

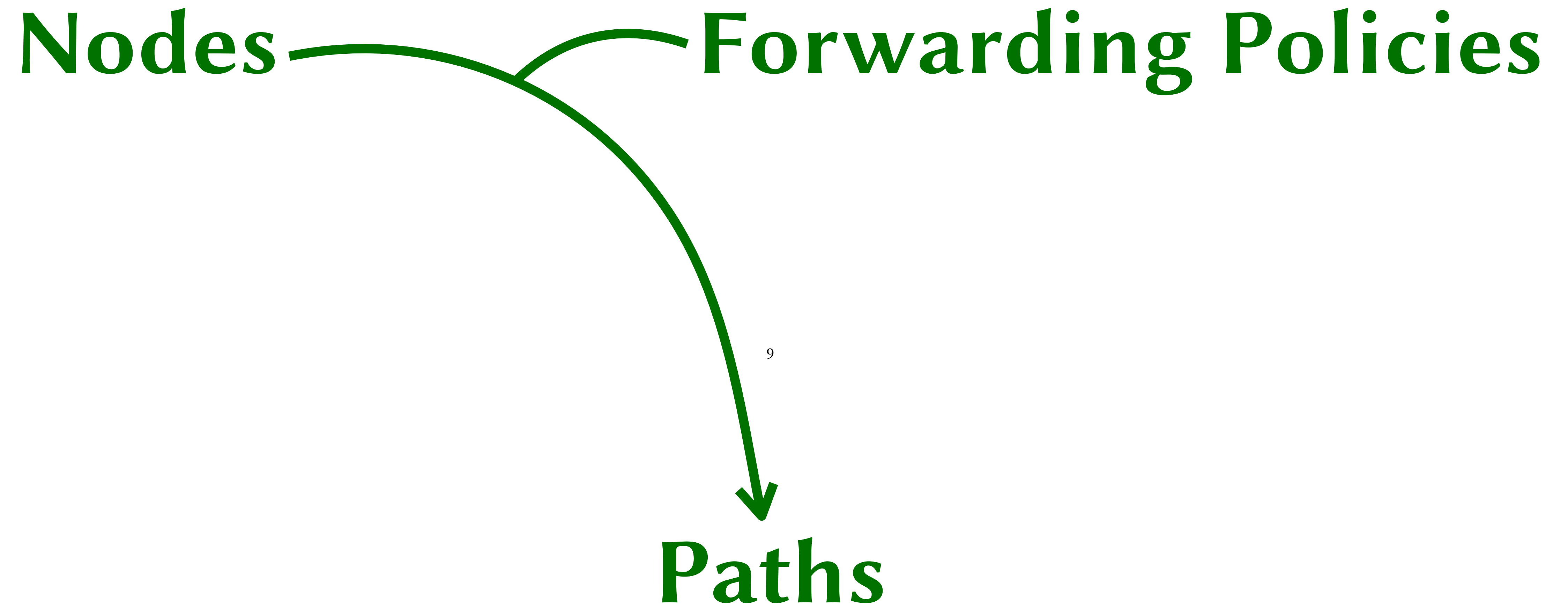
What are essential network features?

Nodes **Forwarding Policies**

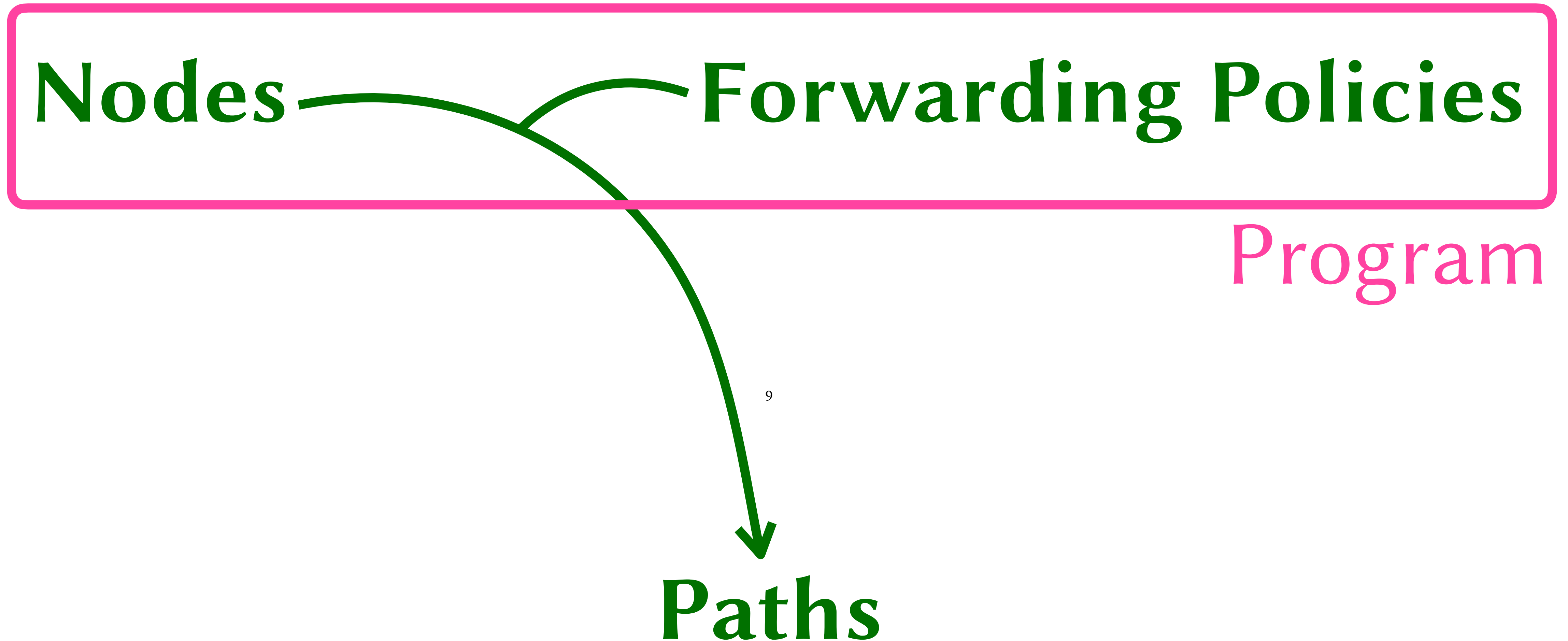


Paths

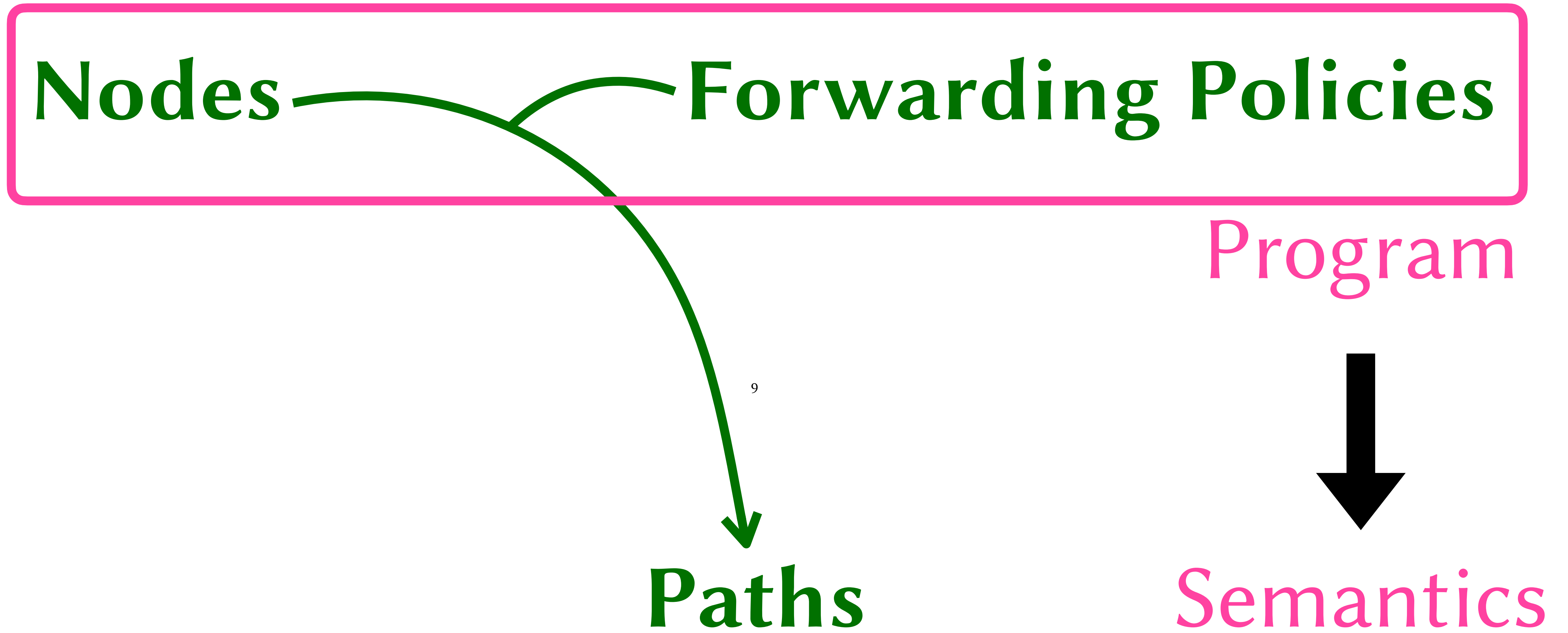
What are essential network features?



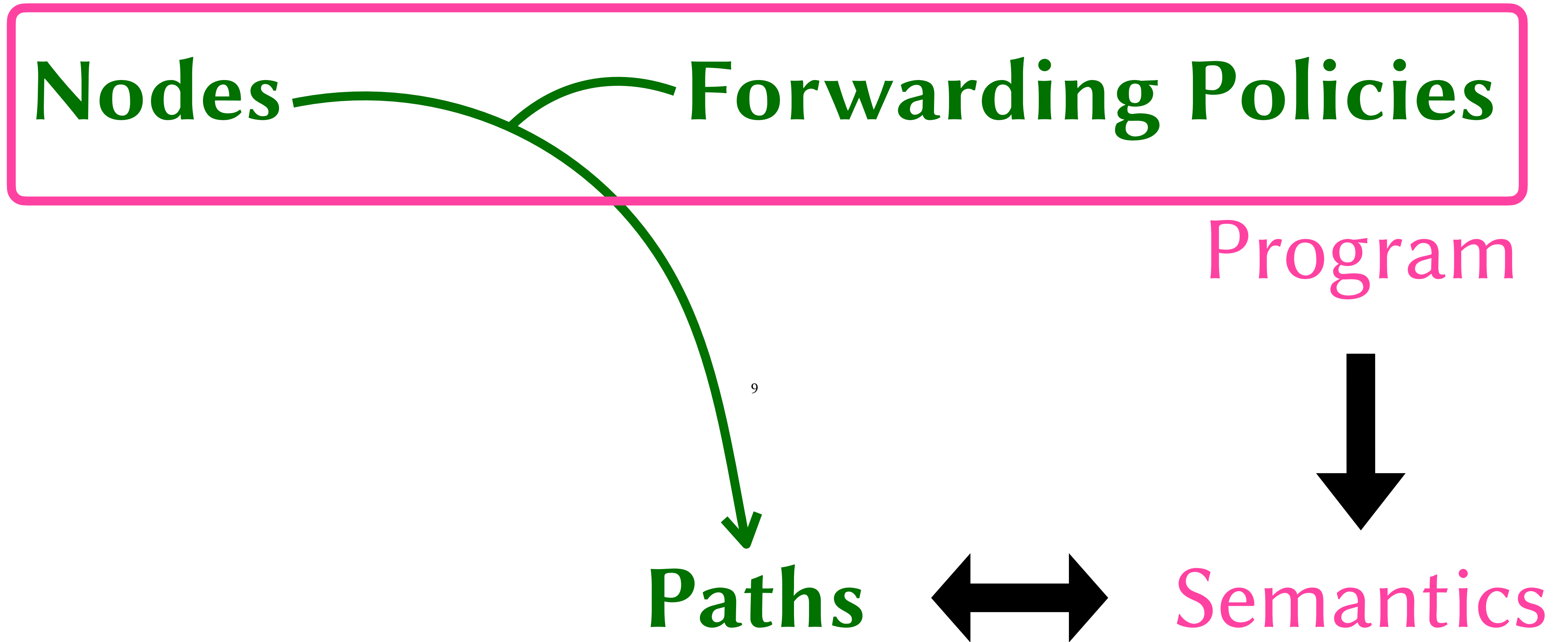
What are essential network features?



What are essential network features?



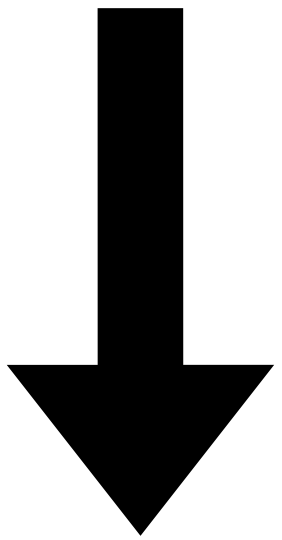
What are essential network features?



What are essential network features?

Nodes **Forwarding Policies**

Program

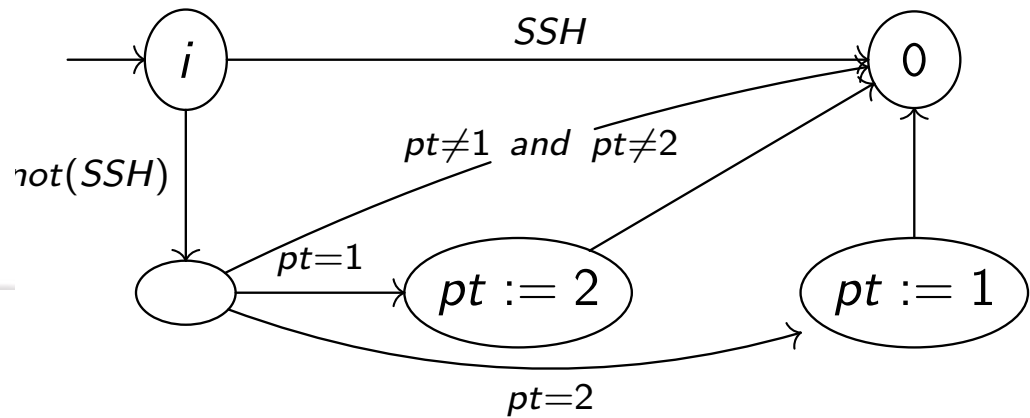


Semantics

Paths



Automata



NetKAT: a network language for reachability

Automata  Paths

NetKAT: a network language for reachability



NetKAT: a network language for reachability

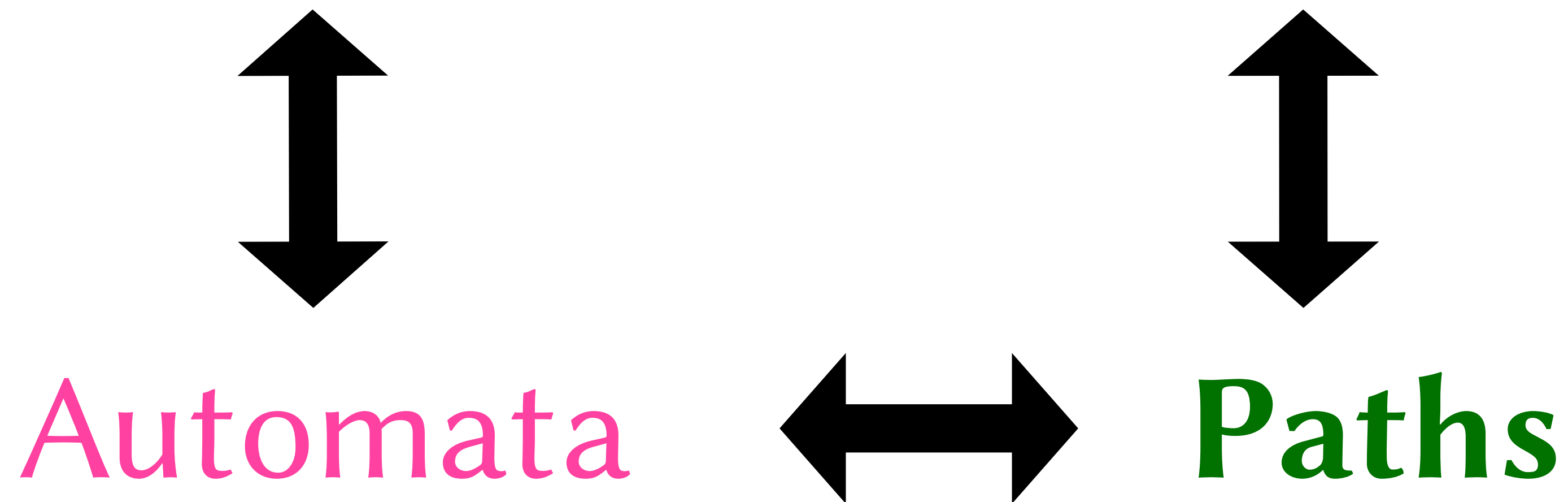
Nodes + Tests + Forwarding Policies



NetKAT: a network language for reachability

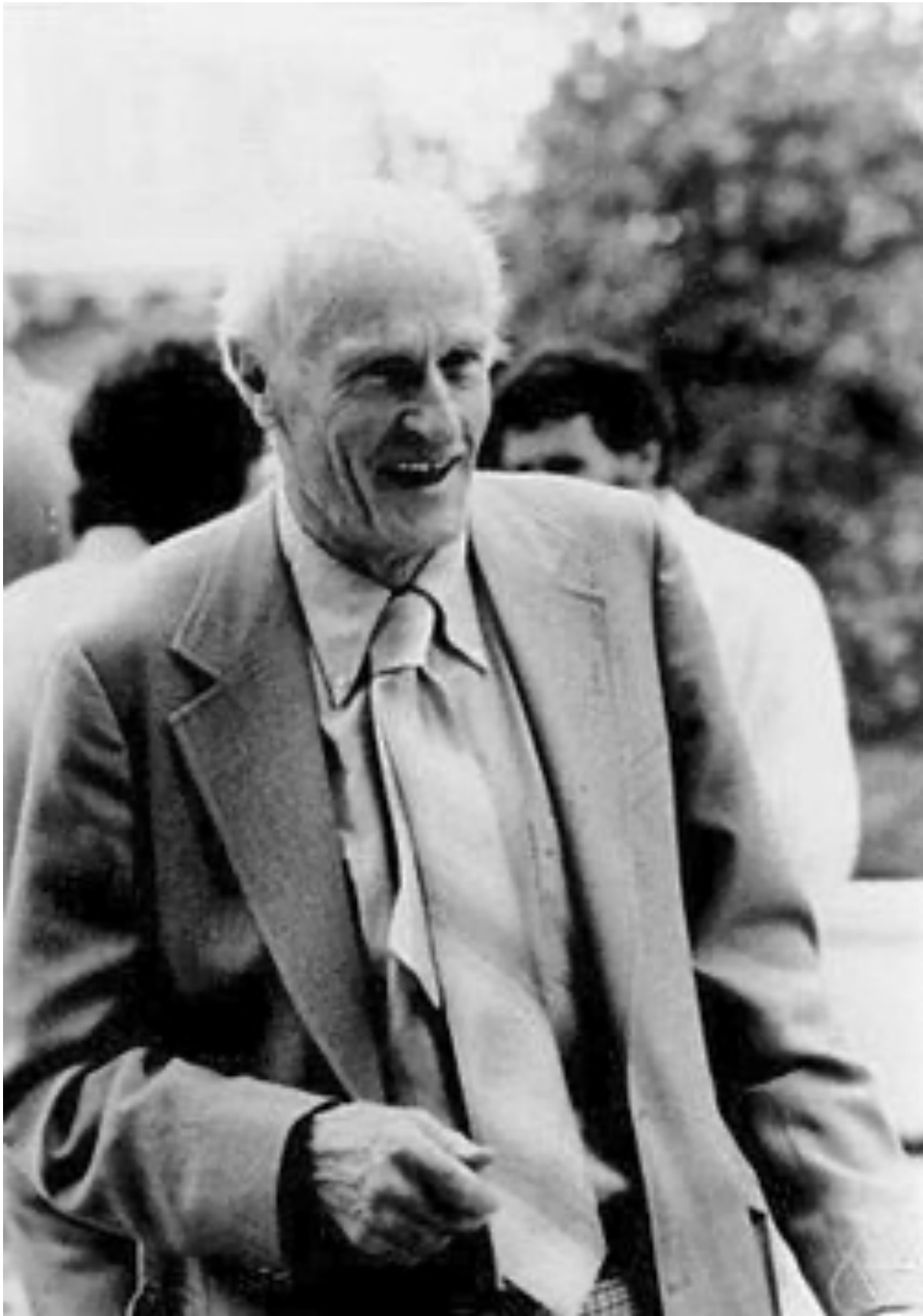
Nodes + Tests + Forwarding Policies

Generalised Regular Expressions



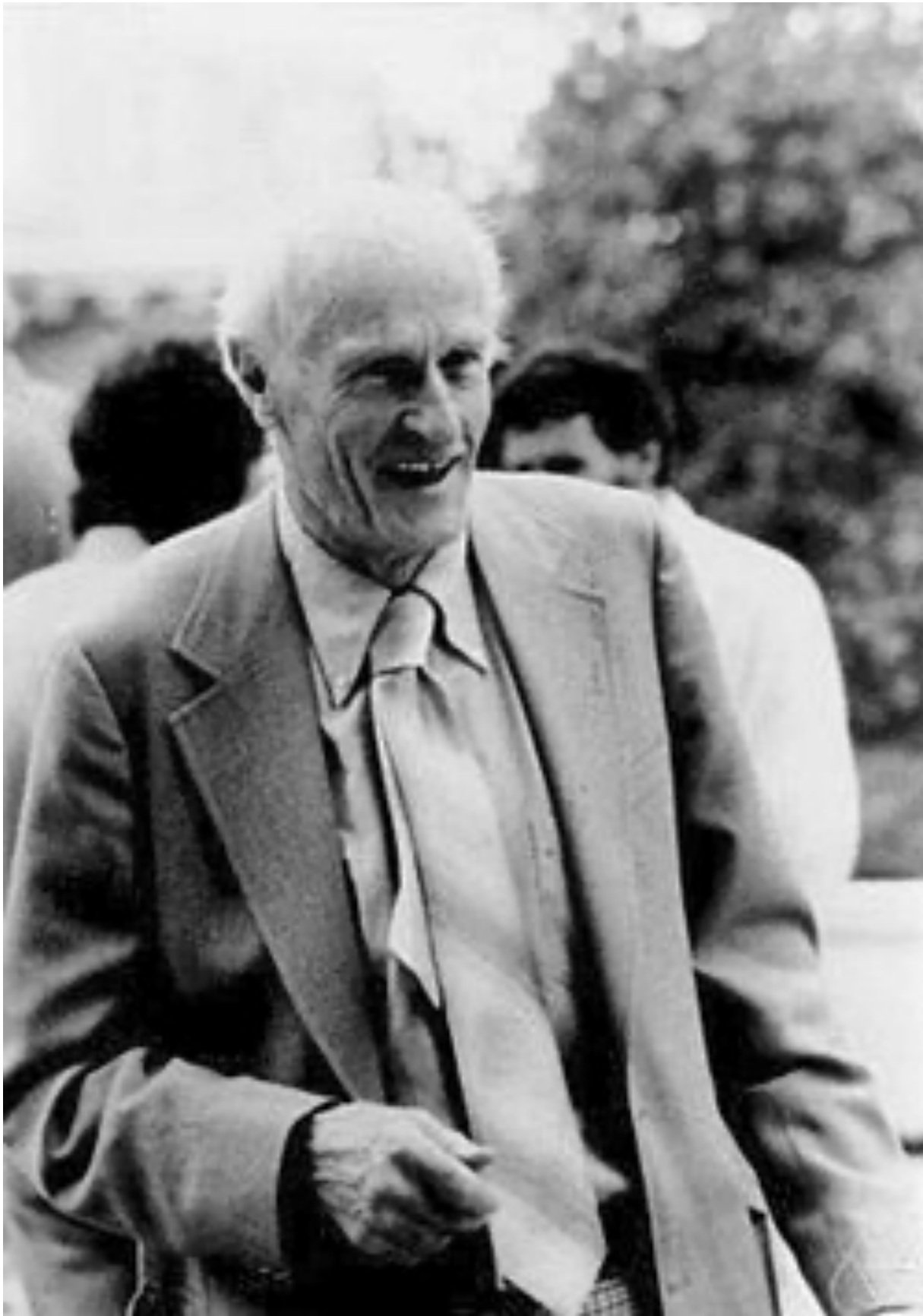
Let's go back to the beginning...

Kleene Algebra



Stephen Cole Kleene
(1909—1994)

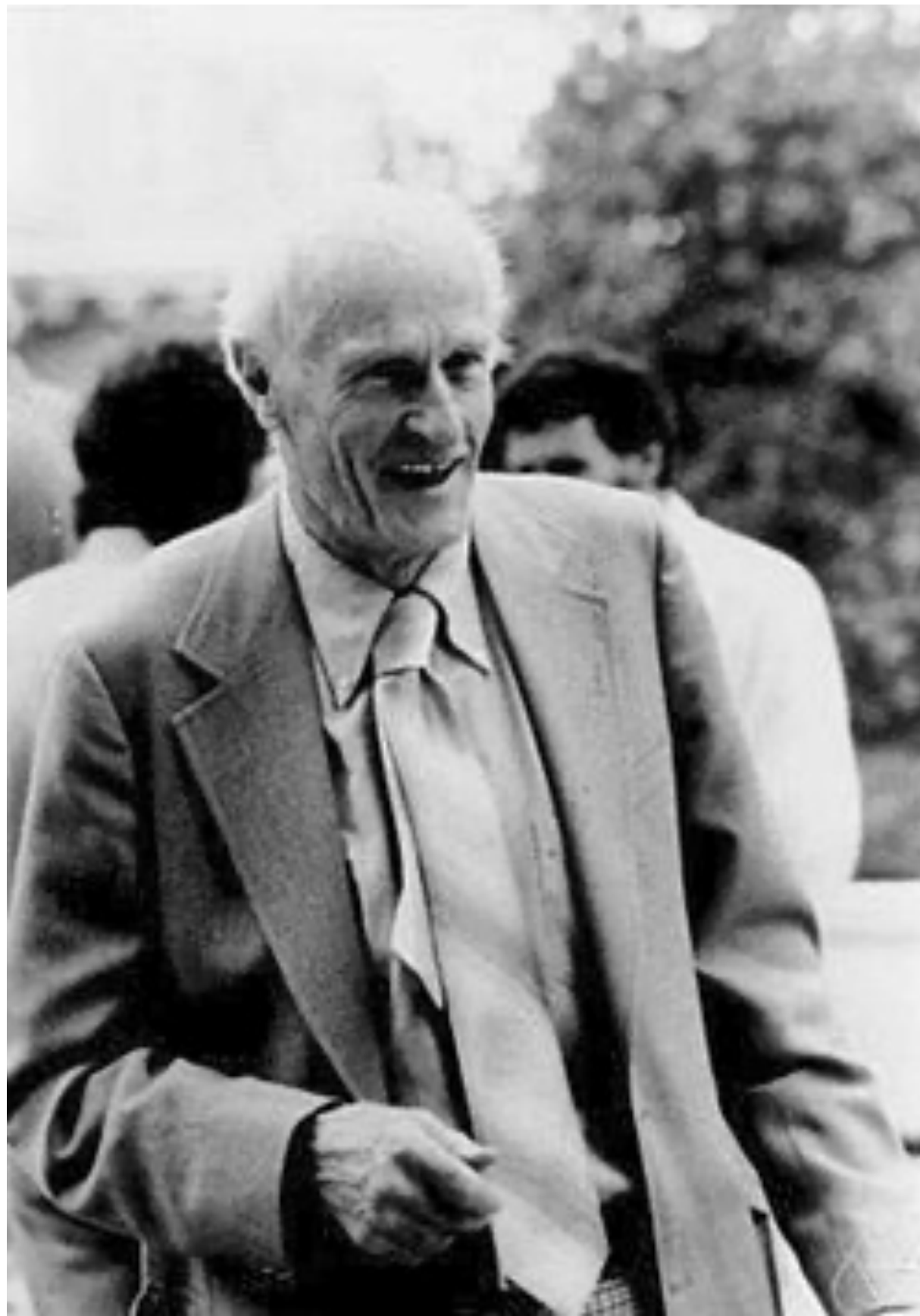
Kleene Algebra



Stephen Cole Kleene
(1909—1994)

$p, q ::=$

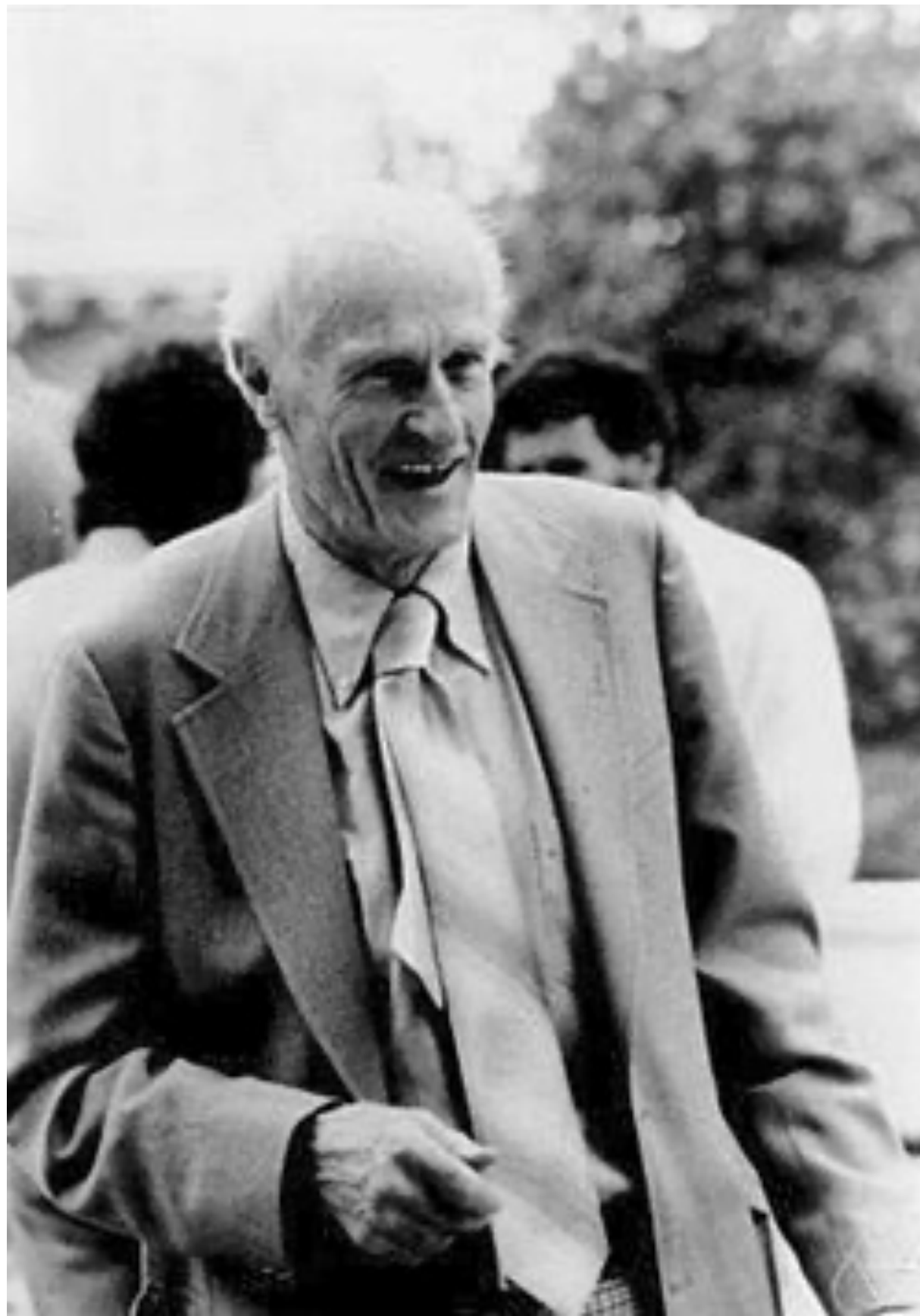
Kleene Algebra



Stephen Cole Kleene
(1909—1994)

```
p, q ::=  
      0      // abort
```

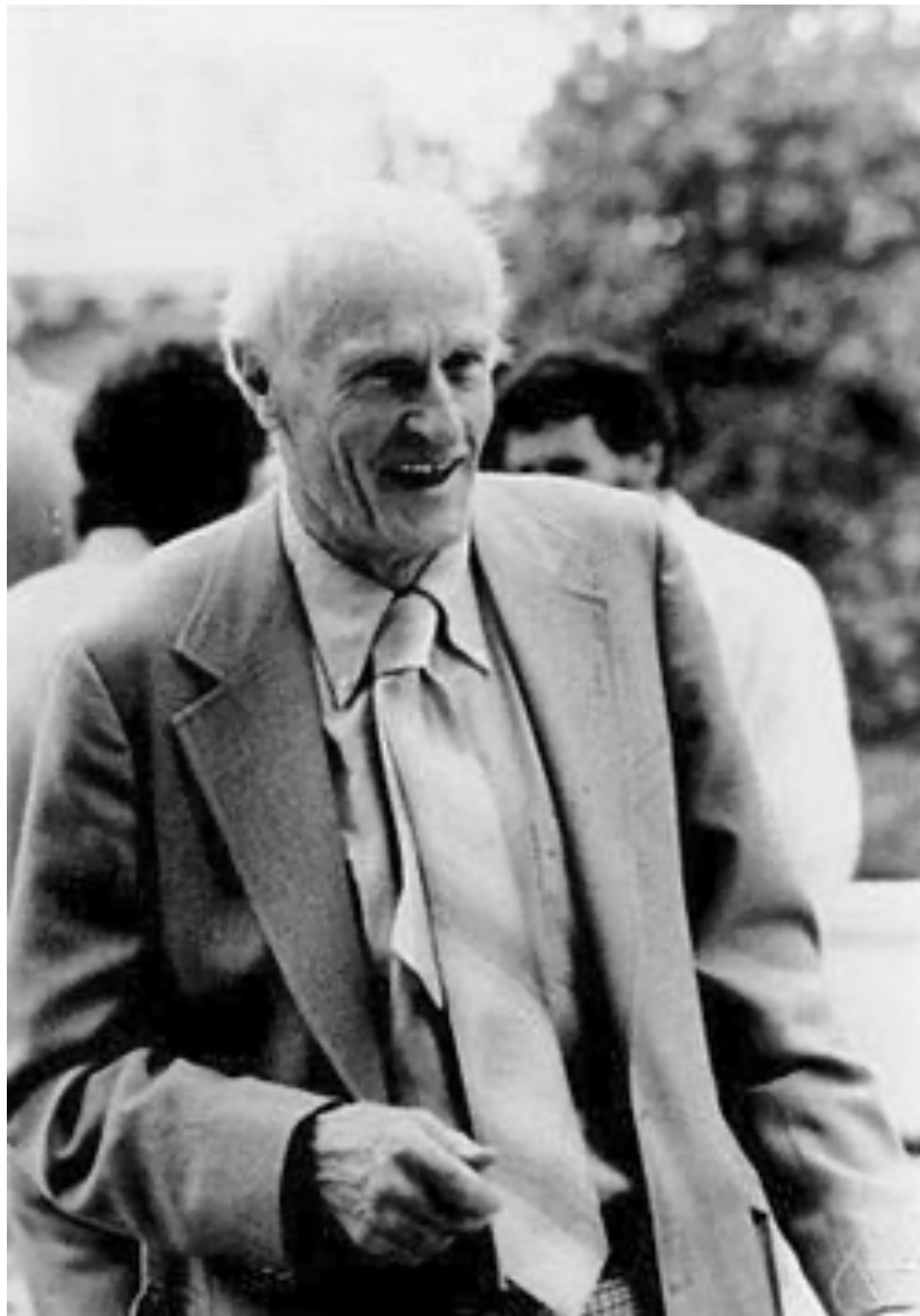
Kleene Algebra



Stephen Cole Kleene
(1909–1994)

```
p, q ::=  
      0      // abort  
    | 1      // skip
```

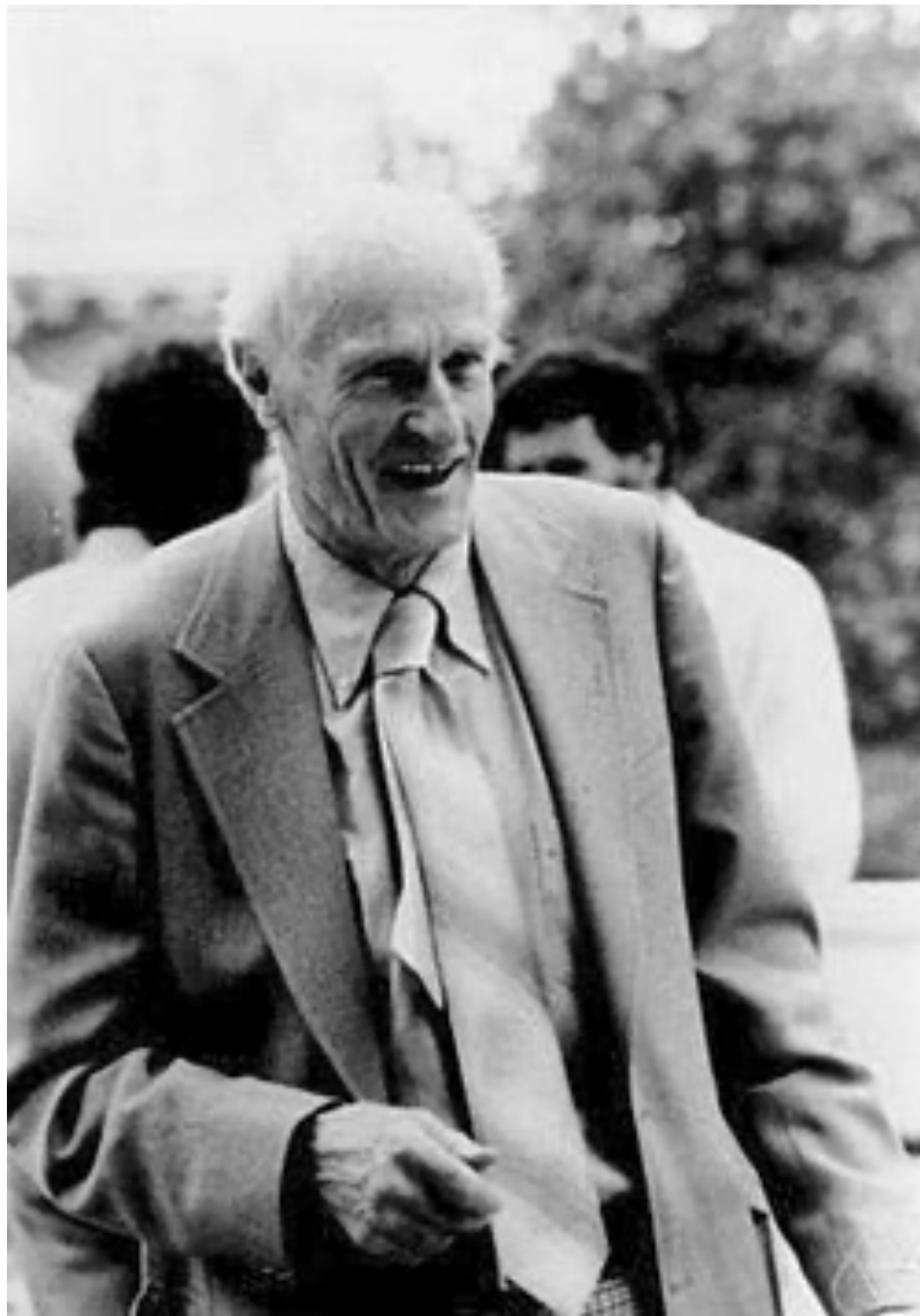
Kleene Algebra



Stephen Cole Kleene
(1909—1994)

```
p, q ::=  
      0      // abort  
    | 1      // skip  
    | a      // action
```

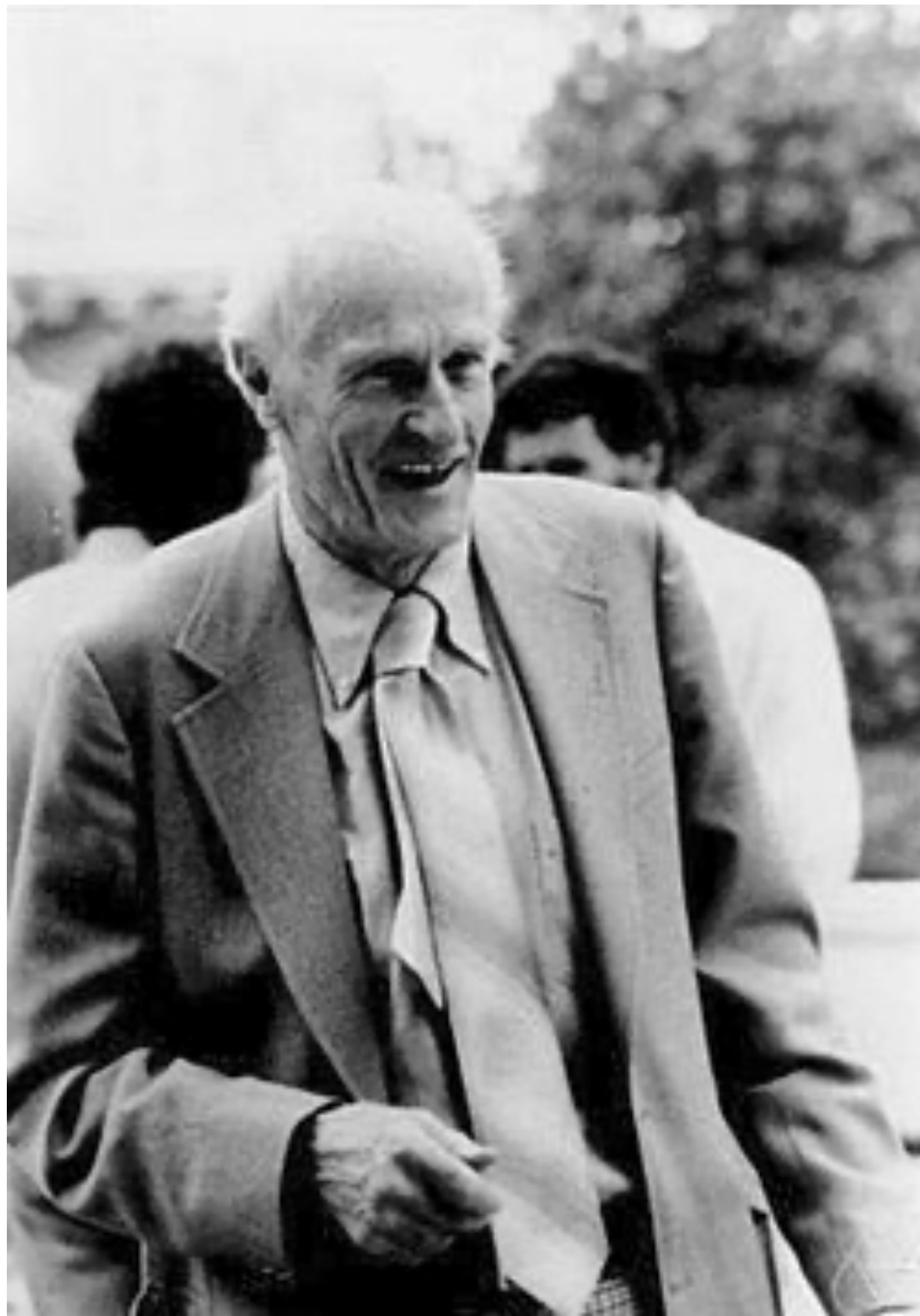
Kleene Algebra



Stephen Cole Kleene
(1909—1994)

```
p, q ::=  
      0           // abort  
    | 1           // skip  
    | a           // action  
    | p + q       // choice
```

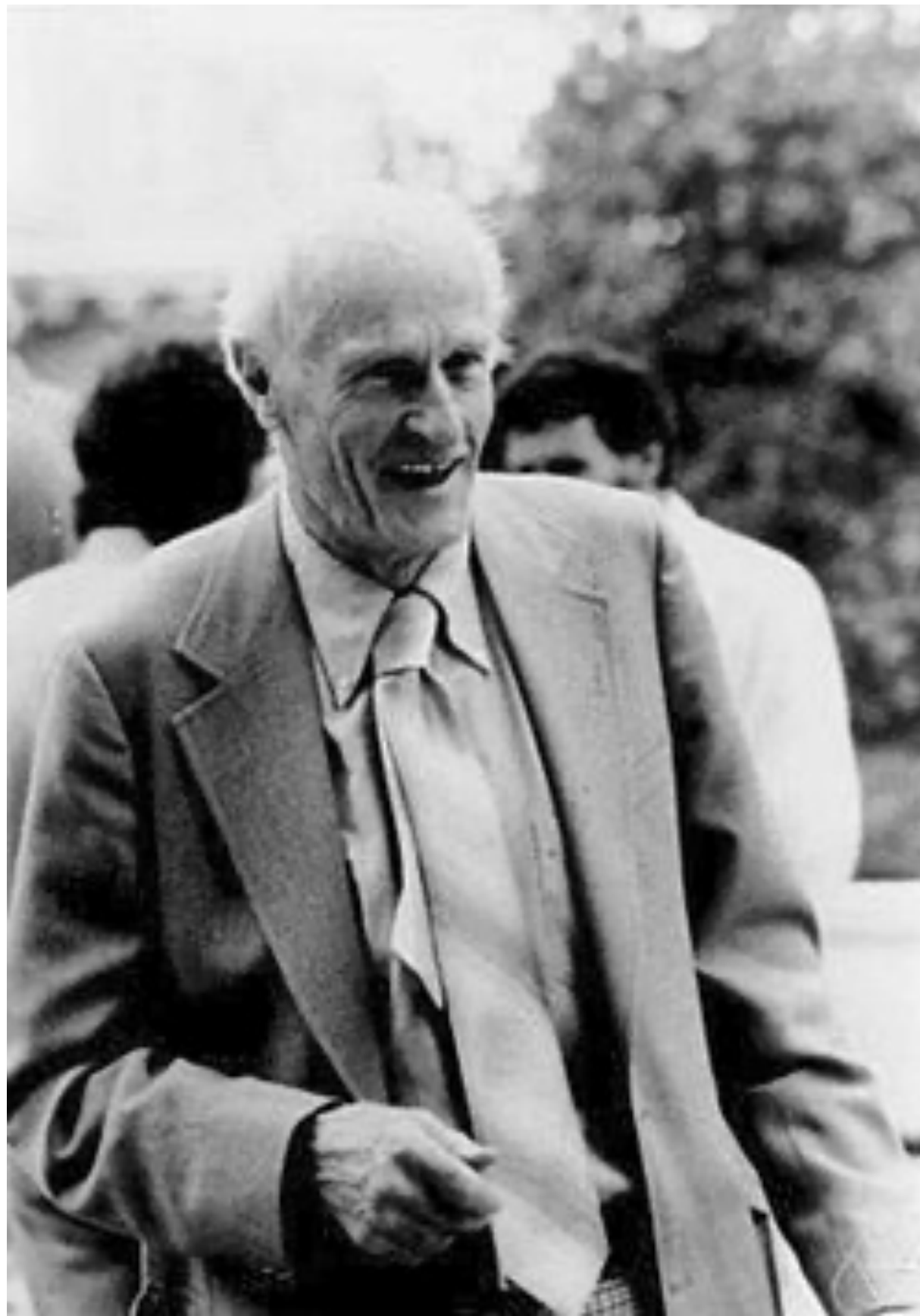
Kleene Algebra



Stephen Cole Kleene
(1909—1994)

```
p, q ::=  
      0           // abort  
    | 1           // skip  
    | a           // action  
    | p + q       // choice  
    | p · q       // sequential composition
```

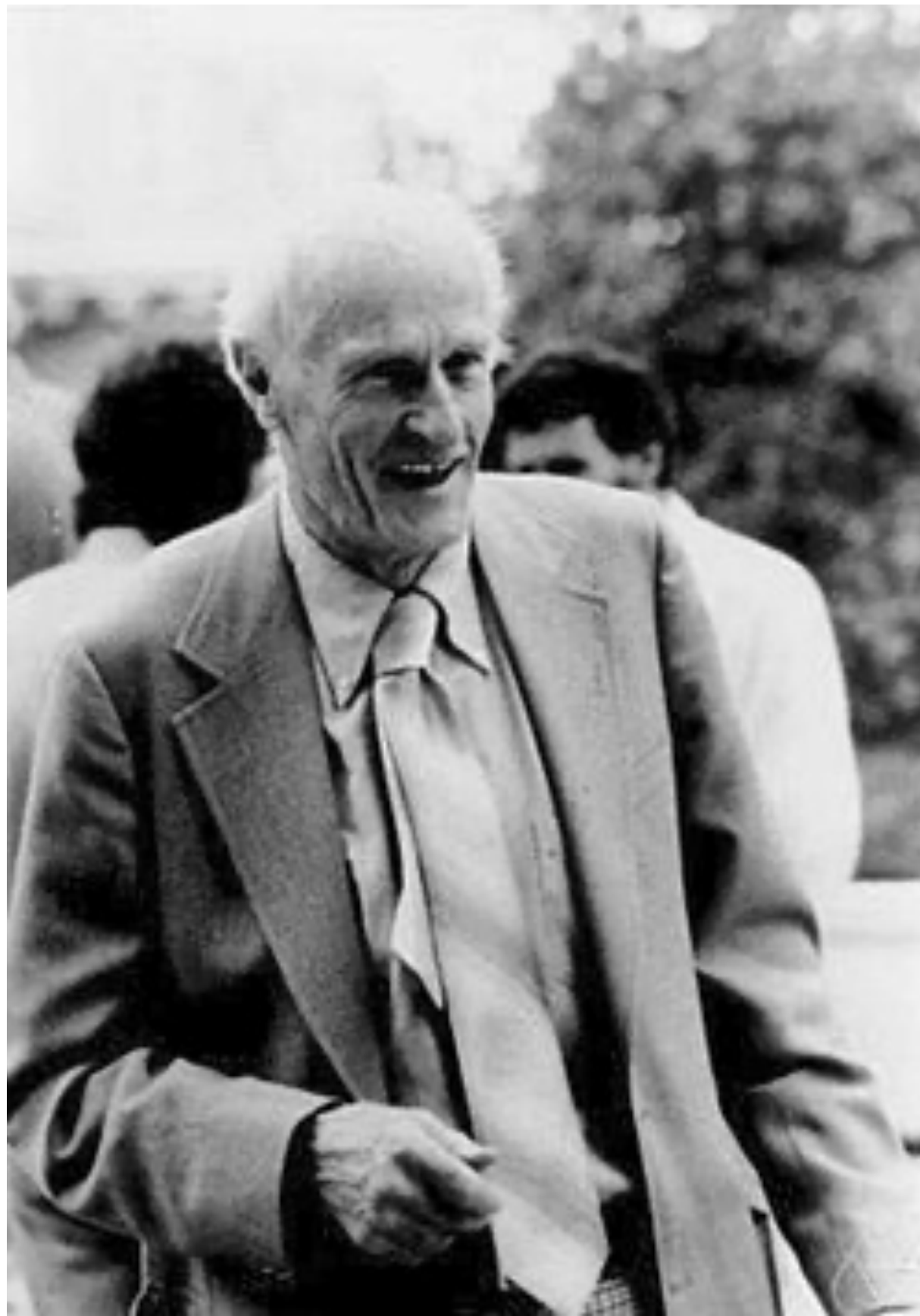
Kleene Algebra



Stephen Cole Kleene
(1909—1994)

```
p, q ::=  
    0           // abort  
    | 1         // skip  
    | a         // action  
    | p + q     // choice  
    | p · q     // sequential composition  
    | p*        // iteration
```

Kleene Algebra

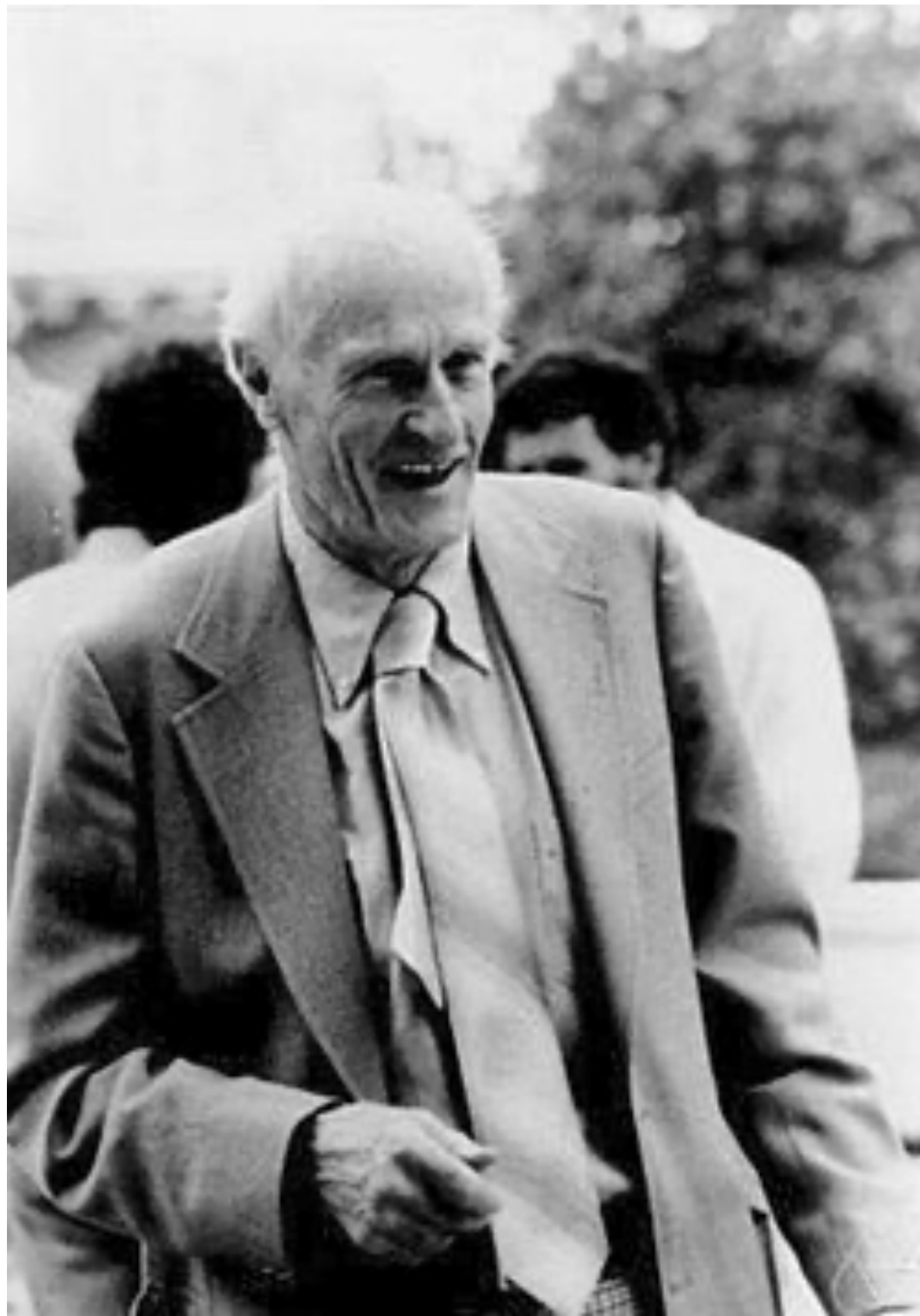


Stephen Cole Kleene
(1909—1994)

```
p, q ::=
    0           // abort
  | 1           // skip
  | a           // action
  | p + q       // choice
  | p · q       // sequential composition
  | p*          // iteration
```

```
(1+a) · (1+b) // ' ' or 'a' or 'b' or 'ab'
```

Kleene Algebra

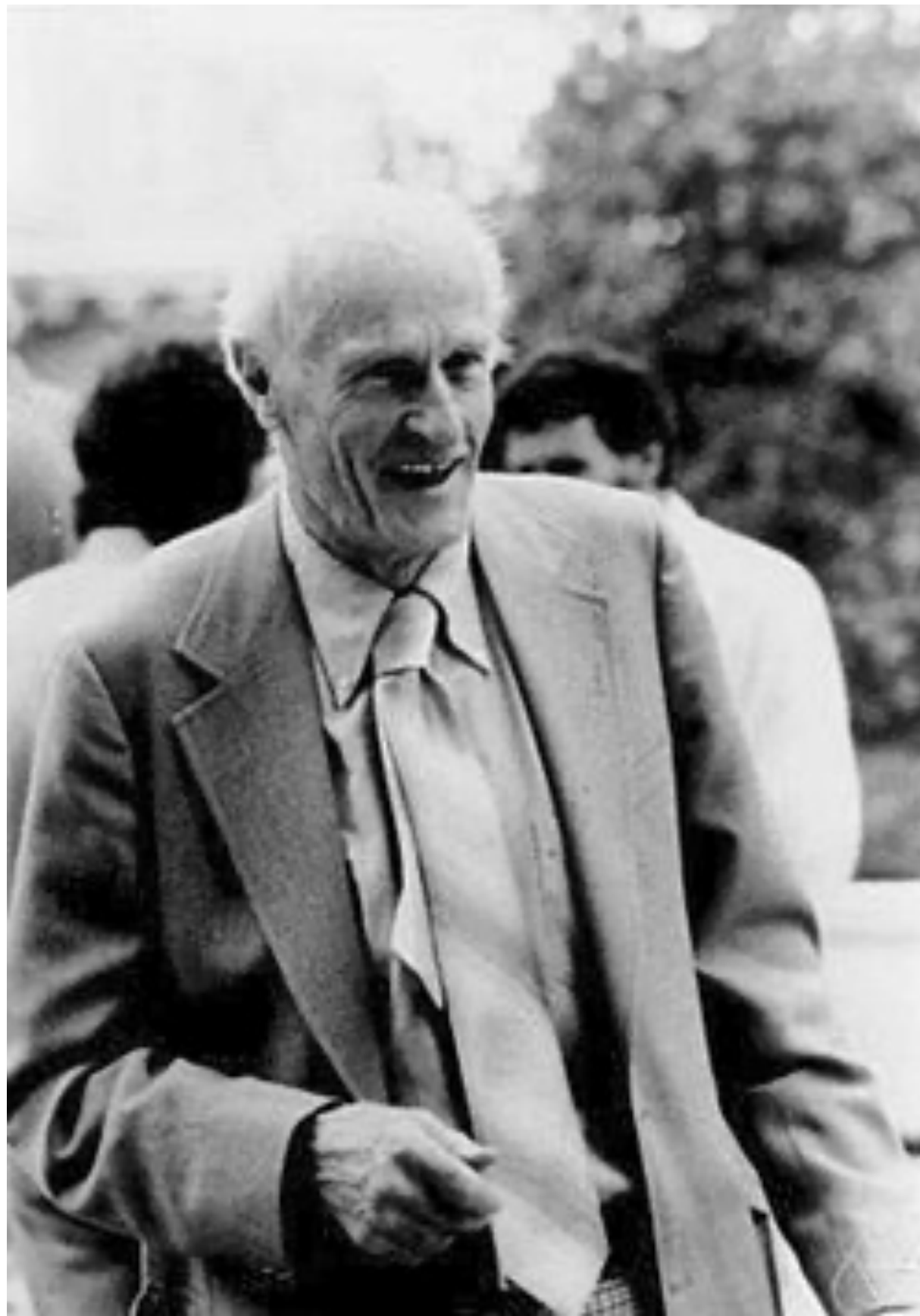


Stephen Cole Kleene
(1909—1994)

```
p, q ::=
    0           // abort
  | 1           // skip
  | a           // action
  | p + q       // choice
  | p · q       // sequential composition
  | p*          // iteration

(1+a) · (1+b) // ' ' or 'a' or 'b' or 'ab'
(a+b)*        // any sequence of 'a's and 'b's
```

Kleene Algebra

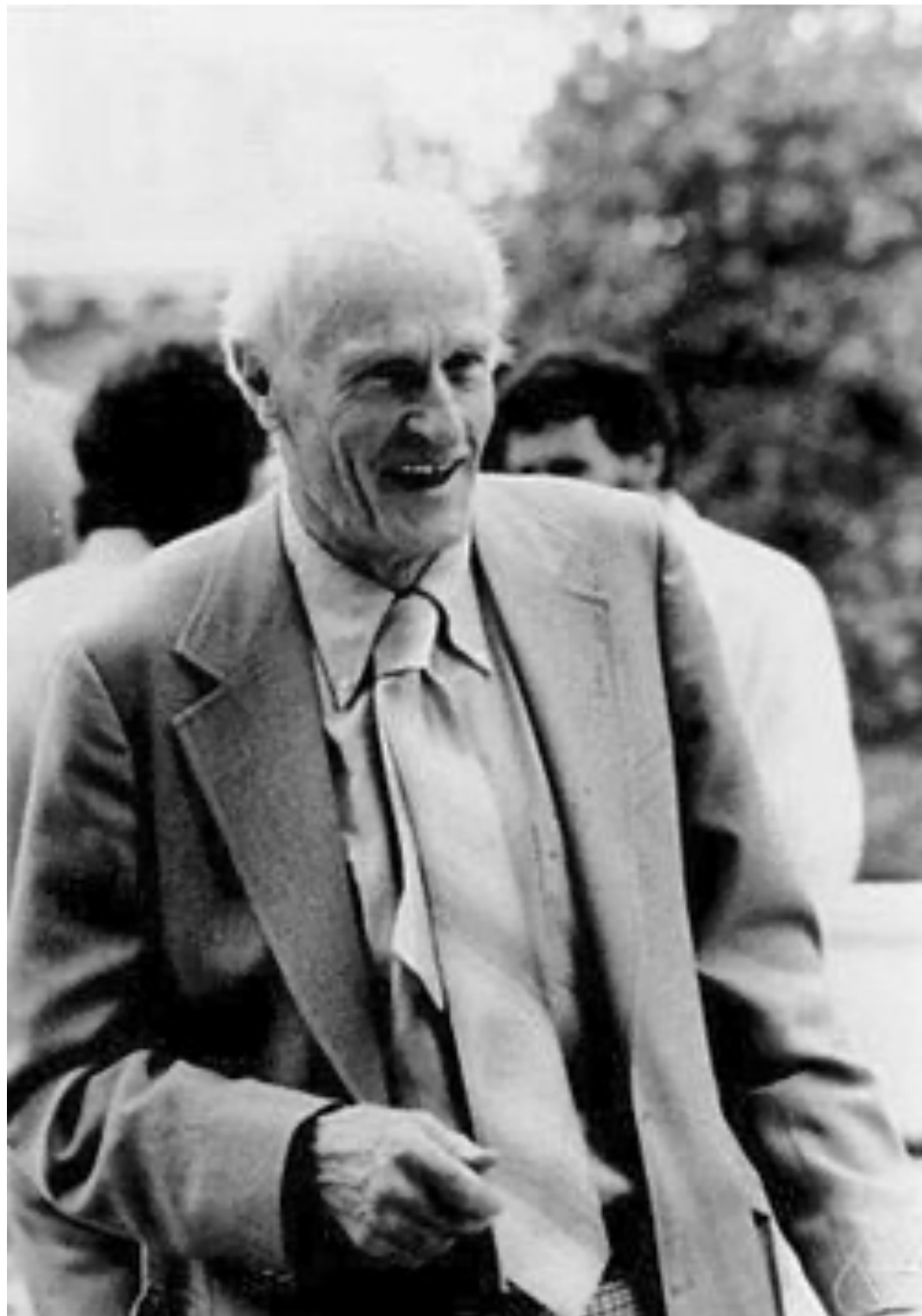


Stephen Cole Kleene
(1909—1994)

```
p, q ::=
    0           // abort
  | 1           // skip
  | a           // action
  | p + q       // choice
  | p · q       // sequential composition
  | p*          // iteration

(1+a) · (1+b) // ' ' or 'a' or 'b' or 'ab'
(a+b)*        // any sequence of 'a's and 'b's
(a*b)* · a*   // ??
```

Kleene Algebra



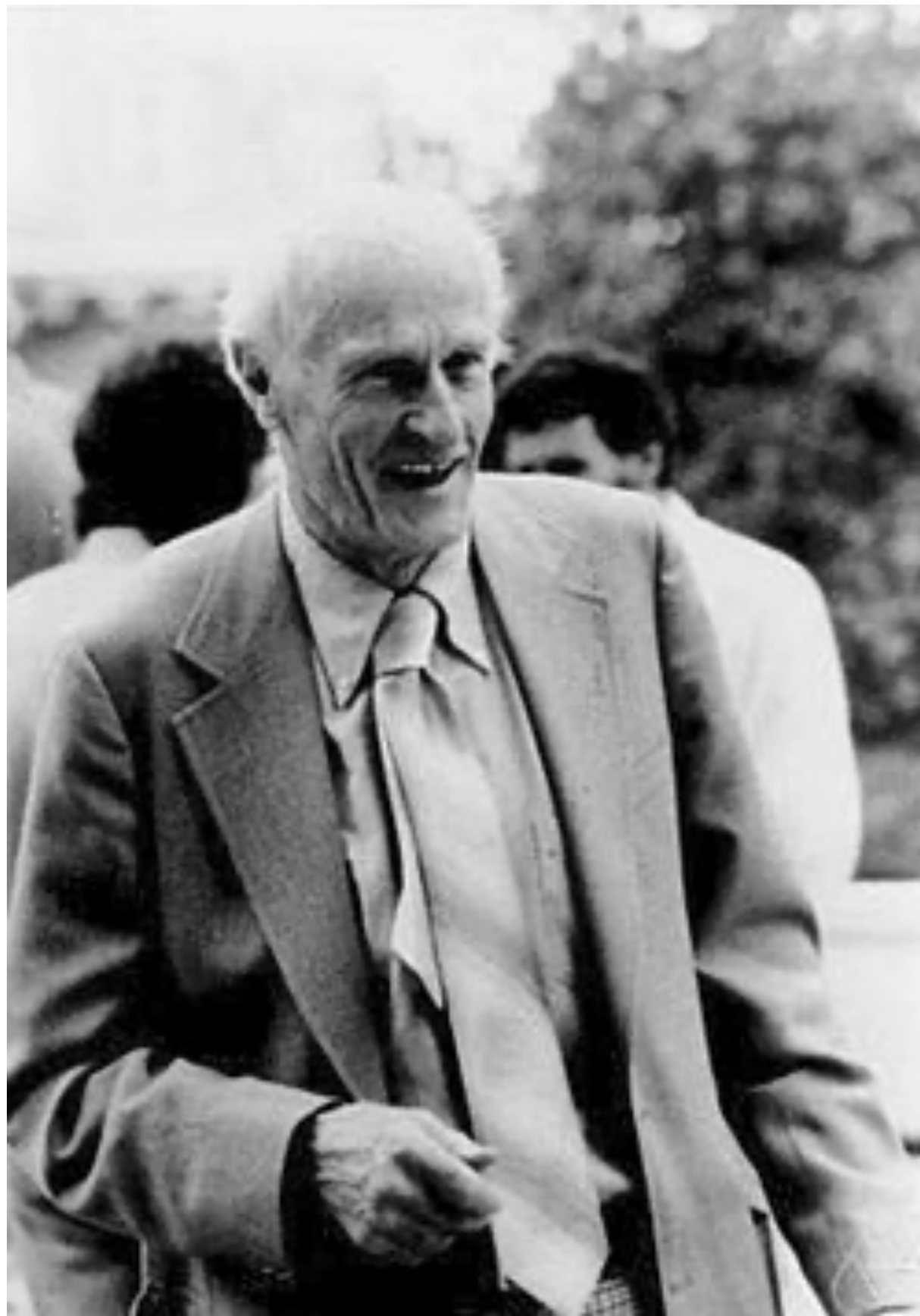
Stephen Cole Kleene
(1909—1994)

```
p, q ::=
    0           // abort
  | 1           // skip
  | a           // action
  | p + q       // choice
  | p · q       // sequential composition
  | p*          // iteration
```

```
(1+a) · (1+b) // ' or 'a' or 'b' or 'ab'
(a+b)*        // any sequence of 'a's and 'b's
(a*b)* · a*   // ??
```

Theorem. $(a+b)^* \equiv (a^*b)^* \cdot a^*$

Kleene Algebra



Stephen Cole Kleene
(1909—1994)

```
p, q ::=
    0           // abort
  | 1           // skip
  | a           // action
  | p + q       // choice
  | p · q       // sequential composition
  | p*          // iteration
```

```
(1+a) · (1+b) // ' or 'a' or 'b' or 'ab'
(a+b)*        // any sequence of 'a's and 'b's
(a*b)* · a*   // ??
```

Theorem. $(a+b)^* \equiv (a^*b)^* \cdot a^*$

A **Kleene Algebra** is an algebraic structure $\langle K, +, \cdot, *, 0, 1 \rangle$ comprising a carrier set K , along with distinguished constants and operations satisfying certain axioms (which will be defined in a few slides)

Kleene Algebra, examples

- Regular languages
 - **+ is union**
 - **; is pointwise concatenation**
 - *** is iteration**

Kleene Algebra, examples

- Regular languages
 - **+ is union**
 - **; is pointwise concatenation**
 - *** is iteration**
- Binary Relations
 - **+ is union**
 - **; is relational composition**
 - *** is reflexive transitive closure**

Kleene Algebra, examples

- Regular languages
 - **+ is union**
 - **; is pointwise concatenation**
 - *** is iteration**
- Binary Relations
 - **+ is union**
 - **; is relational composition**
 - *** is reflexive transitive closure**
- Square Matrices over a KA **K**
 - **+ and ; lifted to usual matrices ops**
 - *** iteratively from 2 x 2**

Kleene Algebra, examples

- Regular languages
 - **+ is union**
 - **; is pointwise concatenation**
 - *** is iteration**
- Binary Relations
 - **+ is union**
 - **; is relational composition**
 - *** is reflexive transitive closure**

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^* = \begin{bmatrix} (a + bd^*c)^* & (a + bd^*c)^*bd^* \\ (d + ca^*b)^*ca^* & (d + ca^*b)^* \end{bmatrix}$$

iteratively from $Z \times Z$

Kleene Algebra, examples

- Regular languages
 - **+ is union**
 - **; is pointwise concatenation**
 - *** is iteration**
- Binary Relations
 - **+ is union**
 - **; is relational composition**
 - *** is reflexive transitive closure**

Important for
relational
verification

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^* = \begin{bmatrix} (a + bd^*c)^* & (a + bd^*c)^*bd^* \\ (d + ca^*b)^*ca^* & (d + ca^*b)^* \end{bmatrix}$$

iteratively from $\mathbb{Z} \times \mathbb{Z}$

Kleene Algebra, examples

- Regular languages
 - **+** is union
 - **;** is pointwise conjunction
 - ***** is iteration

Important for
relational
verification

- Binary Relations
 - **+** is union
 - **;** is relational composition
 - ***** is reflexive transitive closure

Important for
automata
representations

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^* = \begin{bmatrix} (a + bd^*c)^* & (a + bd^*c)^*bd^* \\ (d + ca^*b)^*ca^* & (d + ca^*b)^* \end{bmatrix}$$

iteratively from $Z \times Z$

Kleene Algebra: Regular Semantics

Kleene Algebra: Regular Semantics

Language of traces
of actions

$\llbracket p \rrbracket \subseteq A^*$

Kleene Algebra: Regular Semantics

Language of traces
of actions

$$\llbracket p \rrbracket \subseteq A^*$$

$$\llbracket 0 \rrbracket = \{ \}$$

Kleene Algebra: Regular Semantics

Language of traces
of actions

$$\llbracket p \rrbracket \subseteq A^*$$

$$\llbracket 0 \rrbracket = \{ \}$$

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

Kleene Algebra: Regular Semantics

Language of traces
of actions

$$\llbracket p \rrbracket \subseteq A^*$$

$$\llbracket 0 \rrbracket = \{ \}$$

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

$$\llbracket a \rrbracket = \{ a \cdot \epsilon \}$$

Kleene Algebra: Regular Semantics

Language of traces
of actions

$$\llbracket p \rrbracket \subseteq A^*$$

$$\llbracket 0 \rrbracket = \{ \}$$

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

$$\llbracket a \rrbracket = \{ a \cdot \epsilon \}$$

$$\llbracket p \cdot q \rrbracket = \llbracket p \rrbracket \cdot \llbracket q \rrbracket$$

Kleene Algebra: Regular Semantics

Language of traces
of actions

$$\llbracket p \rrbracket \subseteq A^*$$

$$\llbracket 0 \rrbracket = \{ \}$$

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

$$\llbracket a \rrbracket = \{ a \cdot \epsilon \}$$

$$\llbracket p \cdot q \rrbracket = \llbracket p \rrbracket \cdot \llbracket q \rrbracket$$

$$\llbracket p + q \rrbracket = \llbracket p \rrbracket \cup \llbracket q \rrbracket$$

Kleene Algebra: Regular Semantics

Language of traces
of actions

$$\llbracket p \rrbracket \subseteq A^*$$

$$\llbracket 0 \rrbracket = \{ \}$$

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

$$\llbracket a \rrbracket = \{ a \cdot \epsilon \}$$

$$\llbracket p \cdot q \rrbracket = \llbracket p \rrbracket \cdot \llbracket q \rrbracket$$

$$\llbracket p + q \rrbracket = \llbracket p \rrbracket \cup \llbracket q \rrbracket$$

$$\llbracket p^* \rrbracket = \bigcup_i \llbracket p \rrbracket^i$$

Kleene Algebra: Regular Semantics

Language of traces
of actions

$$\llbracket p \rrbracket \subseteq A^*$$

$$\llbracket 0 \rrbracket = \{ \}$$

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

$$\llbracket a \rrbracket = \{ a \cdot \epsilon \}$$

$$\llbracket p \cdot q \rrbracket = \llbracket p \rrbracket \cdot \llbracket q \rrbracket$$

$$\llbracket p + q \rrbracket = \llbracket p \rrbracket \cup \llbracket q \rrbracket$$

$$\llbracket p^* \rrbracket = \bigcup_i \llbracket p \rrbracket^i$$

$$\text{where } \llbracket p \rrbracket^0 = \{ \epsilon \}$$

Kleene Algebra: Regular Semantics

Language of traces
of actions

$$\llbracket p \rrbracket \subseteq A^*$$

$$\llbracket 0 \rrbracket = \{ \}$$

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

$$\llbracket a \rrbracket = \{ a \cdot \epsilon \}$$

$$\llbracket p \cdot q \rrbracket = \llbracket p \rrbracket \cdot \llbracket q \rrbracket$$

$$\llbracket p + q \rrbracket = \llbracket p \rrbracket \cup \llbracket q \rrbracket$$

$$\llbracket p^* \rrbracket = \bigcup_i \llbracket p \rrbracket^i$$

$$\text{where } \llbracket p \rrbracket^0 = \{ \epsilon \}$$

$$\text{and } \llbracket p \rrbracket^{i+1} = \llbracket p \rrbracket \cdot \llbracket p \rrbracket^i$$

Kleene Algebra: Regular Semantics

$$\llbracket p \rrbracket \subseteq A^*$$

$$\llbracket 0 \rrbracket = \{ \}$$

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

$$\llbracket a \rrbracket = \{ a \cdot \epsilon \}$$

$$\llbracket p \cdot q \rrbracket = \llbracket p \rrbracket \cdot \llbracket q \rrbracket$$

$$\llbracket p + q \rrbracket = \llbracket p \rrbracket \cup \llbracket q \rrbracket$$

$$\llbracket p^* \rrbracket = \bigcup_i \llbracket p \rrbracket^i$$

$$\text{where } \llbracket p \rrbracket^0 = \{ \epsilon \}$$

$$\text{and } \llbracket p \rrbracket^{i+1} = \llbracket p \rrbracket \cdot \llbracket p \rrbracket^i$$

$$(a+b)^* \equiv (a^*b)^* \cdot a^*$$

def.



$$\llbracket (a+b)^* \rrbracket = \llbracket (a^*b)^* \cdot a^* \rrbracket$$

Kleene Algebra: Regular Semantics

$$\llbracket p \rrbracket \subseteq A^*$$

$$\llbracket 0 \rrbracket = \{ \}$$

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

$$\llbracket a \rrbracket = \{ a \cdot \epsilon \}$$

$$\llbracket p \cdot q \rrbracket = \llbracket p \rrbracket \cdot \llbracket q \rrbracket$$

$$\llbracket p + q \rrbracket = \llbracket p \rrbracket \cup \llbracket q \rrbracket$$

$$\llbracket p^* \rrbracket = \bigcup_i \llbracket p \rrbracket^i$$

$$\text{where } \llbracket p \rrbracket^0 = \{ \epsilon \}$$

$$\text{and } \llbracket p \rrbracket^{i+1} = \llbracket p \rrbracket \cdot \llbracket p \rrbracket^i$$

$$(a+b)^* \equiv (a^*b)^* \cdot a^*$$

def.



$$\llbracket (a+b)^* \rrbracket = \llbracket (a^*b)^* \cdot a^* \rrbracket$$

How do we prove this
efficiently?



Kleene Algebra: Regular Semantics

$$\llbracket p \rrbracket \subseteq A^*$$

$$\llbracket 0 \rrbracket = \{ \}$$

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

$$\llbracket a \rrbracket = \{ a \cdot \epsilon \}$$

$$\llbracket p \cdot q \rrbracket = \llbracket p \rrbracket \cdot \llbracket q \rrbracket$$

$$\llbracket p + q \rrbracket = \llbracket p \rrbracket \cup \llbracket q \rrbracket$$

$$\llbracket p^* \rrbracket = \bigcup_i \llbracket p \rrbracket^i$$

$$\text{where } \llbracket p \rrbracket^0 = \{ \epsilon \}$$

$$\text{and } \llbracket p \rrbracket^{i+1} = \llbracket p \rrbracket \cdot \llbracket p \rrbracket^i$$

$$(a+b)^* \equiv (a^*b)^* \cdot a^*$$

def.



$$\llbracket (a+b)^* \rrbracket = \llbracket (a^*b)^* \cdot a^* \rrbracket$$

How do we prove this efficiently?



Kleene's Theorem. TFAE:

1. $L = \llbracket e \rrbracket$, for some regular expression e
2. $L = L(M)$, for some deterministic finite automaton M

Kleene Algebra: Fundamental Results

Kleene Algebra: Fundamental Results

└ **Axiomatization**

Can we characterize the set of valid program equalities/refinements?

Kleene Algebra: Fundamental Results

⊢ **Axiomatization**

Can we characterize the set of valid program equalities/refinements?

✓ **Verification**

Can we automatically check whether two expressions are equivalent?

Kleene Algebra: Fundamental Results

└ **Axiomatization**

Can we characterize the set of valid program equalities/refinements?

Yes: Salomaa (1966); Kozen (1990)

✓ **Verification**

Can we automatically check whether two expressions are equivalent?

Kleene Algebra: Fundamental Results

⊢ **Axiomatization**

Can we characterize the set of valid program equalities/refinements?

Yes: Salomaa (1966); Kozen (1990)

✓ **Verification**

Can we automatically check whether two expressions are equivalent?

Yes: Stockmeyer (1974); Bonchi-Pous (2015)

Kleene Algebra: Axiomatization

Kleene Algebra: Axiomatization

$$Z: p + 0 \equiv p$$

Kleene Algebra: Axiomatization

$$Z: p + 0 \equiv p$$

$$C: p + q \equiv q + p$$

Kleene Algebra: Axiomatization

$$Z: p + \emptyset \equiv p$$

$$C: p + q \equiv q + p$$

$$A: (p + q) + r \equiv p + (q + r)$$

Kleene Algebra: Axiomatization

$$Z: p + \emptyset \equiv p$$

$$C: p + q \equiv q + p$$

$$A: (p + q) + r \equiv p + (q + r)$$

$$I: p + p \equiv p$$

Kleene Algebra: Axiomatization

$$Z: p + \emptyset \equiv p$$

$$C: p + q \equiv q + p$$

$$A: (p + q) + r \equiv p + (q + r)$$

$$I: p + p \equiv p$$

} JSL

Kleene Algebra: Axiomatization

$$Z: p + 0 \equiv p$$

$$C: p + q \equiv q + p$$

$$A: (p + q) + r \equiv p + (q + r)$$

$$I: p + p \equiv p$$

} JSL

$$SN: p \cdot 1 \equiv 1 \cdot p \equiv p$$

Kleene Algebra: Axiomatization

$$Z: p + 0 \equiv p$$

$$C: p + q \equiv q + p$$

$$A: (p + q) + r \equiv p + (q + r)$$

$$I: p + p \equiv p$$

$$SN: p \cdot 1 \equiv 1 \cdot p \equiv p$$

$$SZ: p \cdot 0 \equiv 0 \cdot p \equiv 0$$

} JSL

Kleene Algebra: Axiomatization

$$Z: p + 0 \equiv p$$

$$C: p + q \equiv q + p$$

$$A: (p + q) + r \equiv p + (q + r)$$

$$I: p + p \equiv p$$

$$SN: p \cdot 1 \equiv 1 \cdot p \equiv p$$

$$SZ: p \cdot 0 \equiv 0 \cdot p \equiv 0$$

$$SA: p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$$

} JSL

Kleene Algebra: Axiomatization

$$Z: p + 0 \equiv p$$

$$C: p + q \equiv q + p$$

$$A: (p + q) + r \equiv p + (q + r)$$

$$I: p + p \equiv p$$

$$SN: p \cdot 1 \equiv 1 \cdot p \equiv p$$

$$SZ: p \cdot 0 \equiv 0 \cdot p \equiv 0$$

$$SA: p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$$

$$SD1: p \cdot (q + r) \equiv p \cdot q + p \cdot r$$

} JSL

Kleene Algebra: Axiomatization

$$Z: p + 0 \equiv p$$

$$C: p + q \equiv q + p$$

$$A: (p + q) + r \equiv p + (q + r)$$

$$I: p + p \equiv p$$

} JSL

$$SN: p \cdot 1 \equiv 1 \cdot p \equiv p$$

$$SZ: p \cdot 0 \equiv 0 \cdot p \equiv 0$$

$$SA: p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$$

$$SDl: p \cdot (q + r) \equiv p \cdot q + p \cdot r$$

$$SDr: (p + q) \cdot r \equiv p \cdot r + q \cdot r$$

Kleene Algebra: Axiomatization

Z: $p + 0 \equiv p$
C: $p + q \equiv q + p$
A: $(p + q) + r \equiv p + (q + r)$
I: $p + p \equiv p$

SN: $p \cdot 1 \equiv 1 \cdot p \equiv p$
SZ: $p \cdot 0 \equiv 0 \cdot p \equiv 0$
SA: $p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$
SDl: $p \cdot (q + r) \equiv p \cdot q + p \cdot r$
SDr: $(p + q) \cdot r \equiv p \cdot r + q \cdot r$

$\left. \begin{array}{l} \text{Z} \\ \text{C} \\ \text{A} \\ \text{I} \end{array} \right\} \text{JSL}$
 $\left. \begin{array}{l} \text{SN} \\ \text{SZ} \\ \text{SA} \\ \text{SDl} \\ \text{SDr} \end{array} \right\}$

Idempotent
Semiring

Kleene Algebra: Axiomatization

$$Z: p + 0 \equiv p$$

$$C: p + q \equiv q + p$$

$$A: (p + q) + r \equiv p + (q + r)$$

$$I: p + p \equiv p$$

$$SN: p \cdot 1 \equiv 1 \cdot p \equiv p$$

$$SZ: p \cdot 0 \equiv 0 \cdot p \equiv 0$$

$$SA: p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$$

$$SDl: p \cdot (q + r) \equiv p \cdot q + p \cdot r$$

$$SDr: (p + q) \cdot r \equiv p \cdot r + q \cdot r$$

$$FP: 1 + p \cdot p^* \leq p^*$$

$\left. \begin{array}{l} Z \\ C \\ A \\ I \end{array} \right\} \text{JSL}$

$\left. \begin{array}{l} SN \\ SZ \\ SA \\ SDl \\ SDr \end{array} \right\} \text{Idempotent Semiring}$

Kleene Algebra: Axiomatization

Z: $p + 0 \equiv p$
C: $p + q \equiv q + p$
A: $(p + q) + r \equiv p + (q + r)$
I: $p + p \equiv p$

SN: $p \cdot 1 \equiv 1 \cdot p \equiv p$
SZ: $p \cdot 0 \equiv 0 \cdot p \equiv 0$
SA: $p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$
SDl: $p \cdot (q + r) \equiv p \cdot q + p \cdot r$
SDr: $(p + q) \cdot r \equiv p \cdot r + q \cdot r$

FP: $1 + p \cdot p^* \leq p^*$
 $1 + p^* \cdot p \leq p^*$

} JSL }

Idempotent
Semiring

Kleene Algebra: Axiomatization

Z: $p + 0 \equiv p$
C: $p + q \equiv q + p$
A: $(p + q) + r \equiv p + (q + r)$
I: $p + p \equiv p$

SN: $p \cdot 1 \equiv 1 \cdot p \equiv p$
SZ: $p \cdot 0 \equiv 0 \cdot p \equiv 0$
SA: $p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$
SDl: $p \cdot (q + r) \equiv p \cdot q + p \cdot r$
SDr: $(p + q) \cdot r \equiv p \cdot r + q \cdot r$

FP: $1 + p \cdot p^* \leq p^*$
 $1 + p^* \cdot p \leq p^*$
LFP: $q + p \cdot x \leq x \rightarrow p^* \cdot q \leq x$

} JSL }

Idempotent
Semiring

Kleene Algebra: Axiomatization

Z: $p + 0 \equiv p$
C: $p + q \equiv q + p$
A: $(p + q) + r \equiv p + (q + r)$
I: $p + p \equiv p$

SN: $p \cdot 1 \equiv 1 \cdot p \equiv p$
SZ: $p \cdot 0 \equiv 0 \cdot p \equiv 0$
SA: $p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$
SDl: $p \cdot (q + r) \equiv p \cdot q + p \cdot r$
SDr: $(p + q) \cdot r \equiv p \cdot r + q \cdot r$

FP: $1 + p \cdot p^* \leq p^*$
 $1 + p^* \cdot p \leq p^*$
LFP: $q + p \cdot x \leq x \rightarrow p^* \cdot q \leq x$
 $q + x \cdot p \leq x \rightarrow q \cdot p^* \leq x$

} JSL }

Idempotent
Semiring

Kleene Algebra: Axiomatization

Z: $p + 0 \equiv p$
C: $p + q \equiv q + p$
A: $(p + q) + r \equiv p + (q + r)$
I: $p + p \equiv p$

SN: $p \cdot 1 \equiv 1 \cdot p \equiv p$
SZ: $p \cdot 0 \equiv 0 \cdot p \equiv 0$
SA: $p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$
SDl: $p \cdot (q + r) \equiv p \cdot q + p \cdot r$
SDr: $(p + q) \cdot r \equiv p \cdot r + q \cdot r$

FP: $1 + p \cdot p^* \leq p^*$
 $1 + p^* \cdot p \leq p^*$
LFP: $q + p \cdot x \leq x \rightarrow p^* \cdot q \leq x$
 $q + x \cdot p \leq x \rightarrow q \cdot p^* \leq x$

$\left. \begin{array}{l} \text{Z} \\ \text{C} \\ \text{A} \\ \text{I} \end{array} \right\} \text{JSL}$
 $\left. \begin{array}{l} \text{SN} \\ \text{SZ} \\ \text{SA} \\ \text{SDl} \\ \text{SDr} \end{array} \right\} \text{Idempotent Semiring}$

$\left. \begin{array}{l} \text{FP} \\ 1 + p^* \cdot p \leq p^* \\ \text{LFP} \end{array} \right\} \text{Least Fixpoint}$

Examples

Examples

The following can be proved from the KA axioms:

Examples

The following can be proved from the KA axioms:

- $p^* \equiv p^* \cdot p^*$

Examples

The following can be proved from the KA axioms:

- $p^* \equiv p^* \cdot p^*$
- $p^* \equiv p^{**}$

Examples

The following can be proved from the KA axioms:

- $p^* \equiv p^* \cdot p^*$
- $p^* \equiv p^{**}$
- $(p + q)^* \equiv p^* \cdot (q \cdot p^*)^* \quad // \text{ denesting}$

Examples

The following can be proved from the KA axioms:

- $p^* \equiv p^* \cdot p^*$
- $p^* \equiv p^{**}$
- $(p + q)^* \equiv p^* \cdot (q \cdot p^*)^* \quad // \text{ denesting}$
- $(p \cdot q)^* \cdot p \equiv p \cdot (q \cdot p)^* \quad // \text{ sliding}$

Examples

The following can be proved from the KA axioms:

- $p^* \equiv p^* \cdot p^*$
- $p^* \equiv p^{**}$
- $(p + q)^* \equiv p^* \cdot (q \cdot p^*)^* \quad // \text{ denesting}$
- $(p \cdot q)^* \cdot p \equiv p \cdot (q \cdot p)^* \quad // \text{ sliding}$
- $p \cdot q \equiv q \cdot r \implies p^* \cdot q \equiv q \cdot r^* \quad // \text{ bisimulation}$

Denesting proof (easy direction)

Denesting proof (easy direction)

$$(p^* \cdot q)^* \cdot p^*$$

Denesting proof (easy direction)

$(p^* \cdot q)^* \cdot p^*$
 $\leq \{ \text{monotonicity: } p \leq (p + q) \text{ and } q \leq (p + q) \}$

Denesting proof (easy direction)

$$\begin{aligned} & (p^* \cdot q)^* \cdot p^* \\ & \leq \{ \text{monotonicity: } p \leq (p + q) \text{ and } q \leq (p + q) \} \\ & ((p + q)^* \cdot (p + q))^* \cdot (p + q)^* \end{aligned}$$

Denesting proof (easy direction)

$$\begin{aligned} & (p^* \cdot q)^* \cdot p^* \\ & \leq \{ \text{monotonicity: } p \leq (p + q) \text{ and } q \leq (p + q) \} \\ & ((p + q)^* \cdot (p + q))^* \cdot (p + q)^* \\ & \leq \{ p^* \cdot p \leq p^* \} \end{aligned}$$

Denesting proof (easy direction)

$$\begin{aligned} & (p^* \cdot q)^* \cdot p^* \\ & \leq \{ \text{monotonicity: } p \leq (p + q) \text{ and } q \leq (p + q) \} \\ & ((p + q)^* \cdot (p + q))^* \cdot (p + q)^* \\ & \leq \{ p^* \cdot p \leq p^* \} \\ & (p + q)^{**} \cdot (p + q)^* \end{aligned}$$

Denesting proof (easy direction)

$$\begin{aligned} & (p^* \cdot q)^* \cdot p^* \\ & \leq \{ \text{monotonicity: } p \leq (p + q) \text{ and } q \leq (p + q) \} \\ & ((p + q)^* \cdot (p + q))^* \cdot (p + q)^* \\ & \leq \{ p^* \cdot p \leq p^* \} \\ & (p + q)^{**} \cdot (p + q)^* \\ & \leq \{ p^{**} \equiv p^* \} \end{aligned}$$

Denesting proof (easy direction)

$$\begin{aligned} & (p^* \cdot q)^* \cdot p^* \\ & \leq \{ \text{monotonicity: } p \leq (p + q) \text{ and } q \leq (p + q) \} \\ & ((p + q)^* \cdot (p + q))^* \cdot (p + q)^* \\ & \leq \{ p^* \cdot p \leq p^* \} \\ & (p + q)^{**} \cdot (p + q)^* \\ & \leq \{ p^{**} \equiv p^* \} \\ & (p + q)^* \cdot (p + q)^* \end{aligned}$$

Denesting proof (easy direction)

$$\begin{aligned} & (p^* \cdot q)^* \cdot p^* \\ & \leq \{ \text{monotonicity: } p \leq (p + q) \text{ and } q \leq (p + q) \} \\ & ((p + q)^* \cdot (p + q))^* \cdot (p + q)^* \\ & \leq \{ p^* \cdot p \leq p^* \} \\ & (p + q)^{**} \cdot (p + q)^* \\ & \leq \{ p^{**} \equiv p^* \} \\ & (p + q)^* \cdot (p + q)^* \\ & \leq \{ p^* \cdot p^* \equiv p^* \} \end{aligned}$$

Denesting proof (easy direction)

$$\begin{aligned} & (p^* \cdot q)^* \cdot p^* \\ & \leq \{ \text{monotonicity: } p \leq (p + q) \text{ and } q \leq (p + q) \} \\ & ((p + q)^* \cdot (p + q))^* \cdot (p + q)^* \\ & \leq \{ p^* \cdot p \leq p^* \} \\ & (p + q)^{**} \cdot (p + q)^* \\ & \leq \{ p^{**} \equiv p^* \} \\ & (p + q)^* \cdot (p + q)^* \\ & \leq \{ p^* \cdot p^* \equiv p^* \} \\ & (p + q)^* \end{aligned}$$

Break exercises

1. $x^*x^* = x^*$
2. $x^* = x^{**}$
3. $(x+y)^* = (x^*y)^*x^*$ **denesting**
4. $x(yx)^* = (xy)^*x$ **sliding**
5. $xy=yz \Rightarrow x^*y = yz^*$

<https://netkat.org/ssft25/lab1.html>

Program equivalence

```
while b {  
    p;  
    while c do q  
}
```

?
≡

```
if b then {  
    p;  
    while b or c do  
        if c then q else p  
}
```

Program equivalence

```
while b {  
    p;  
    while c do q  
}
```

?
≡

```
if b then {  
    p;  
    while b or c do  
        if c then q else p  
}
```

Idea: think of programs as generating traces of **actions** **p** and **q**

Program equivalence

```
while b {  
    p;  
    while c do q  
}
```

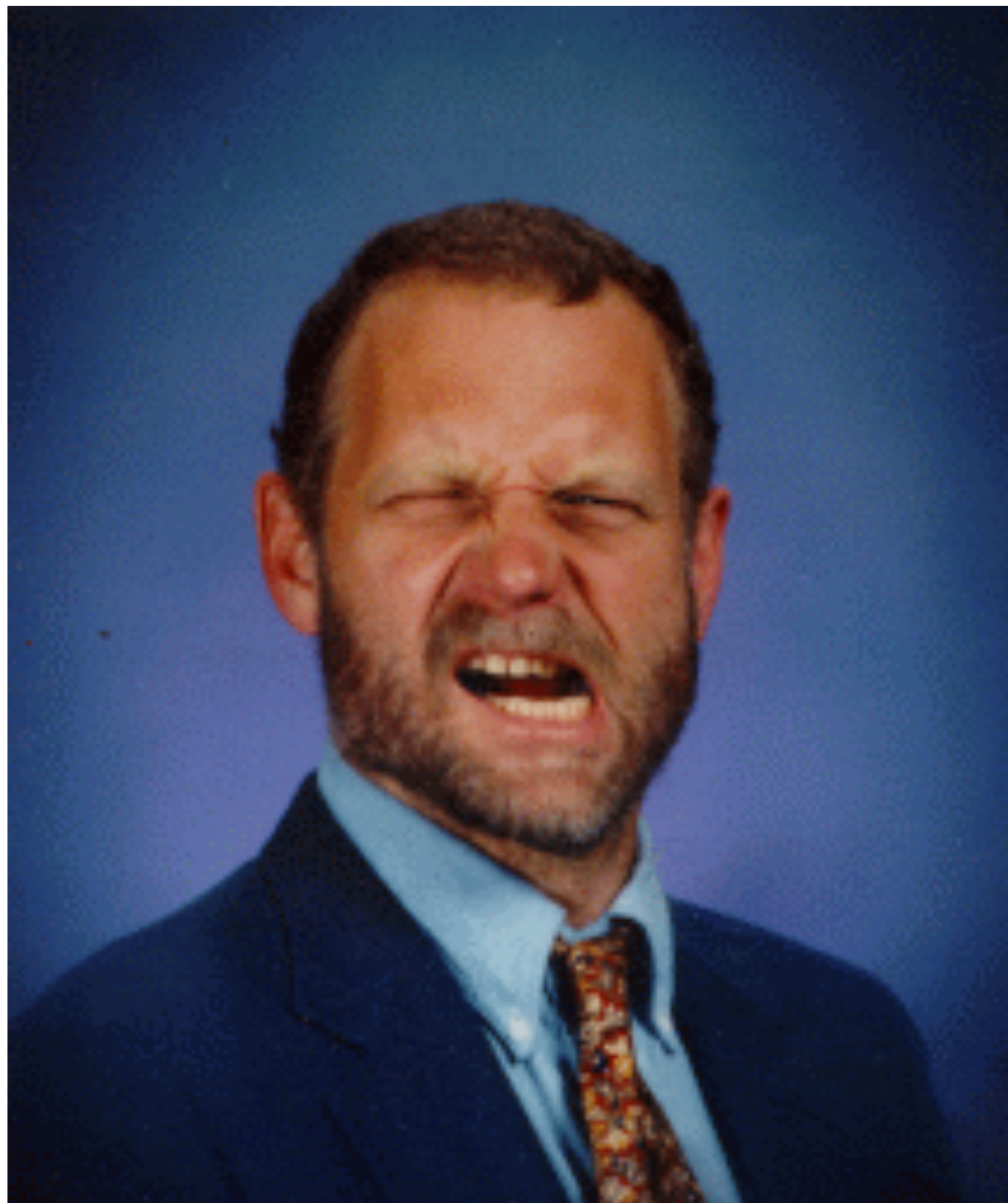
?
≡

```
if b then {  
    p;  
    while b or c do  
        if c then q else p  
}
```

Idea: think of programs as generating traces of **actions** **p** and **q**

Missing feature: **Boolean** tests for control flow!

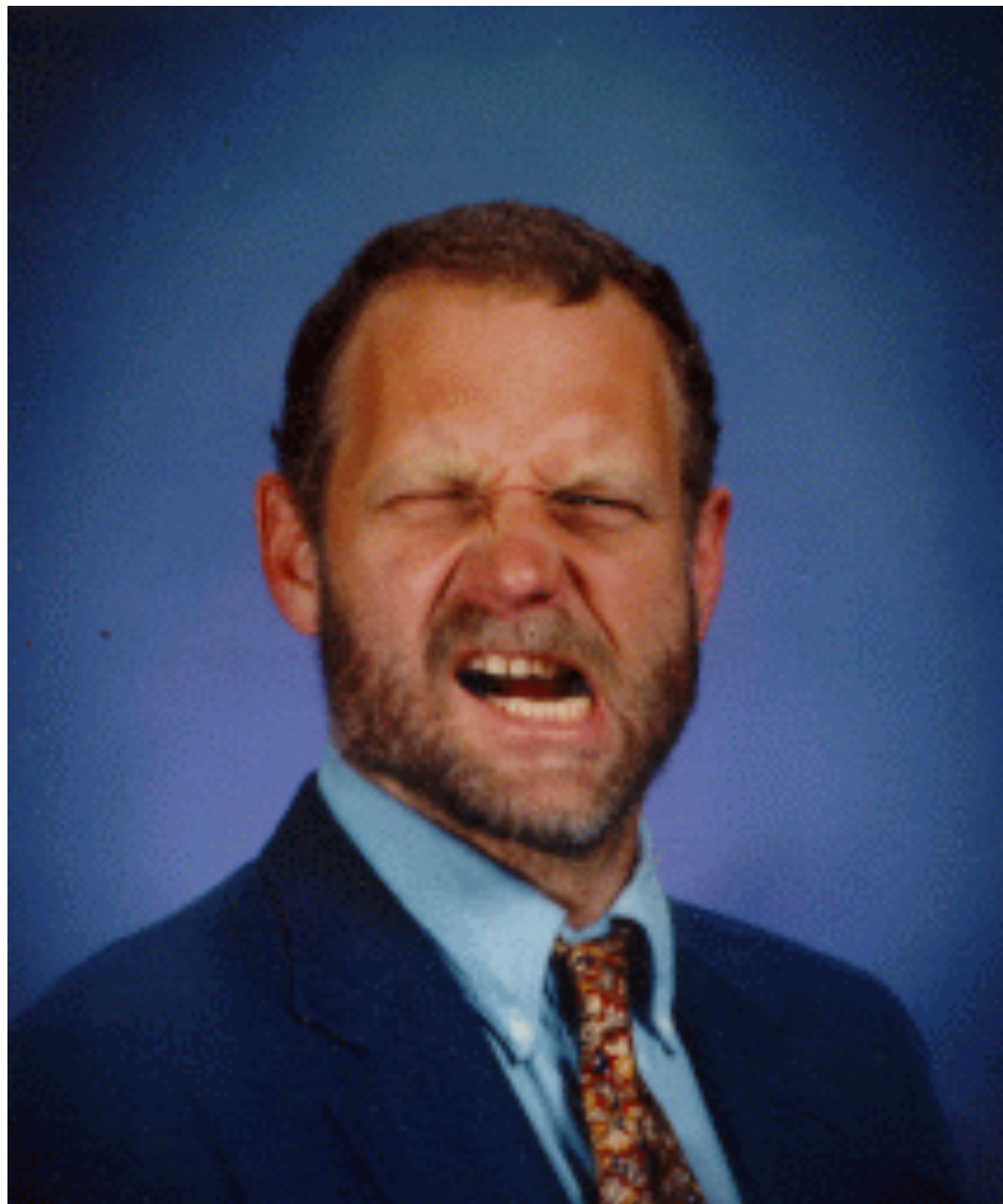
Kleene Algebra with Tests



Dexter Kozen

```
p, q ::=
    0           // abort
  | 1           // skip
    a           // action
  | p + q       // choice
  | p · q       // sequential composition
  | p*          // iteration
```

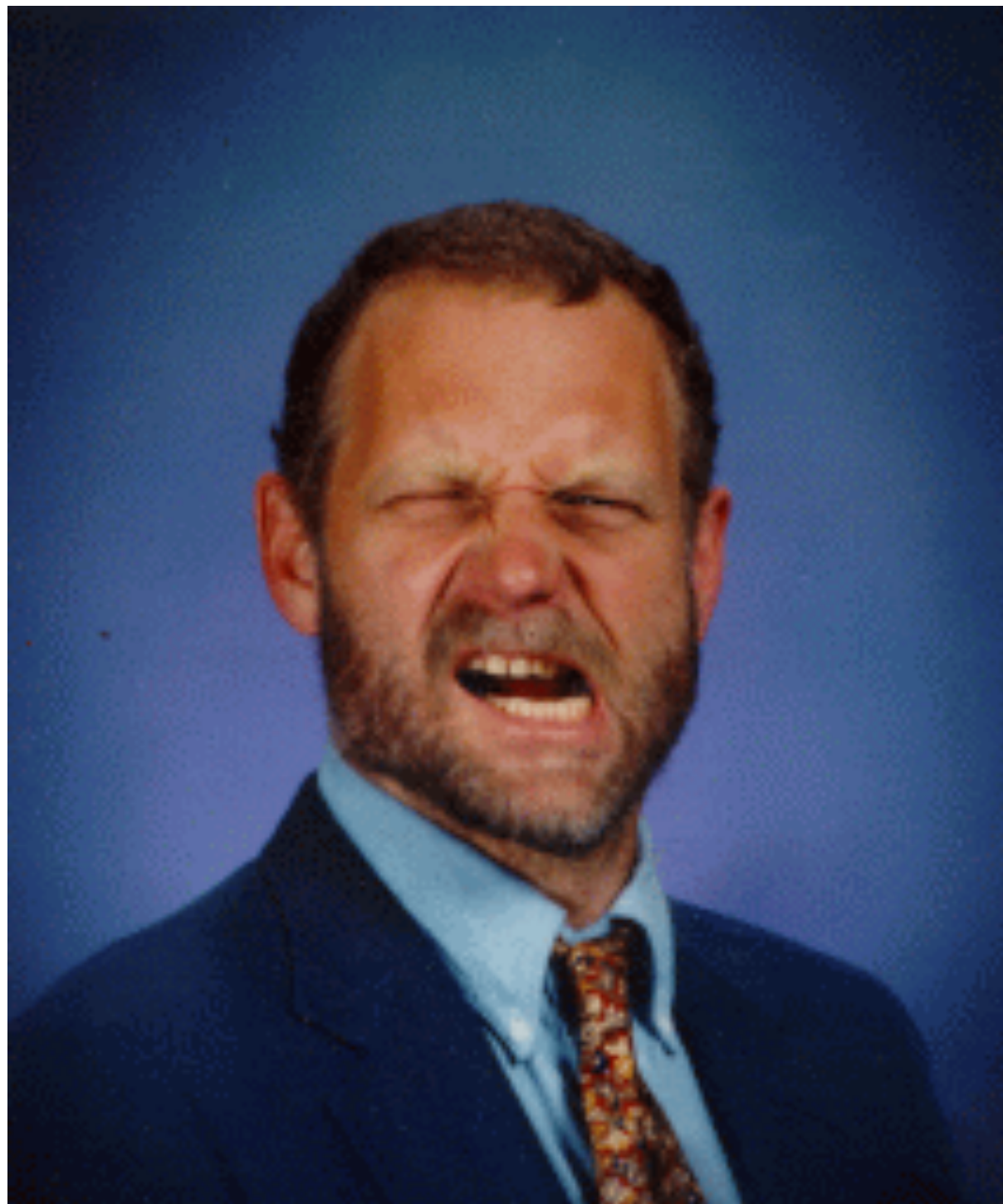
Kleene Algebra with Tests



Dexter Kozen

```
p, q ::=
    0           // abort
  | 1           // skip
  | b, ~b       // Boolean tests
  | a           // action
  | p + q       // choice
  | p · q       // sequential composition
  | p*          // iteration
```

Kleene Algebra with Tests

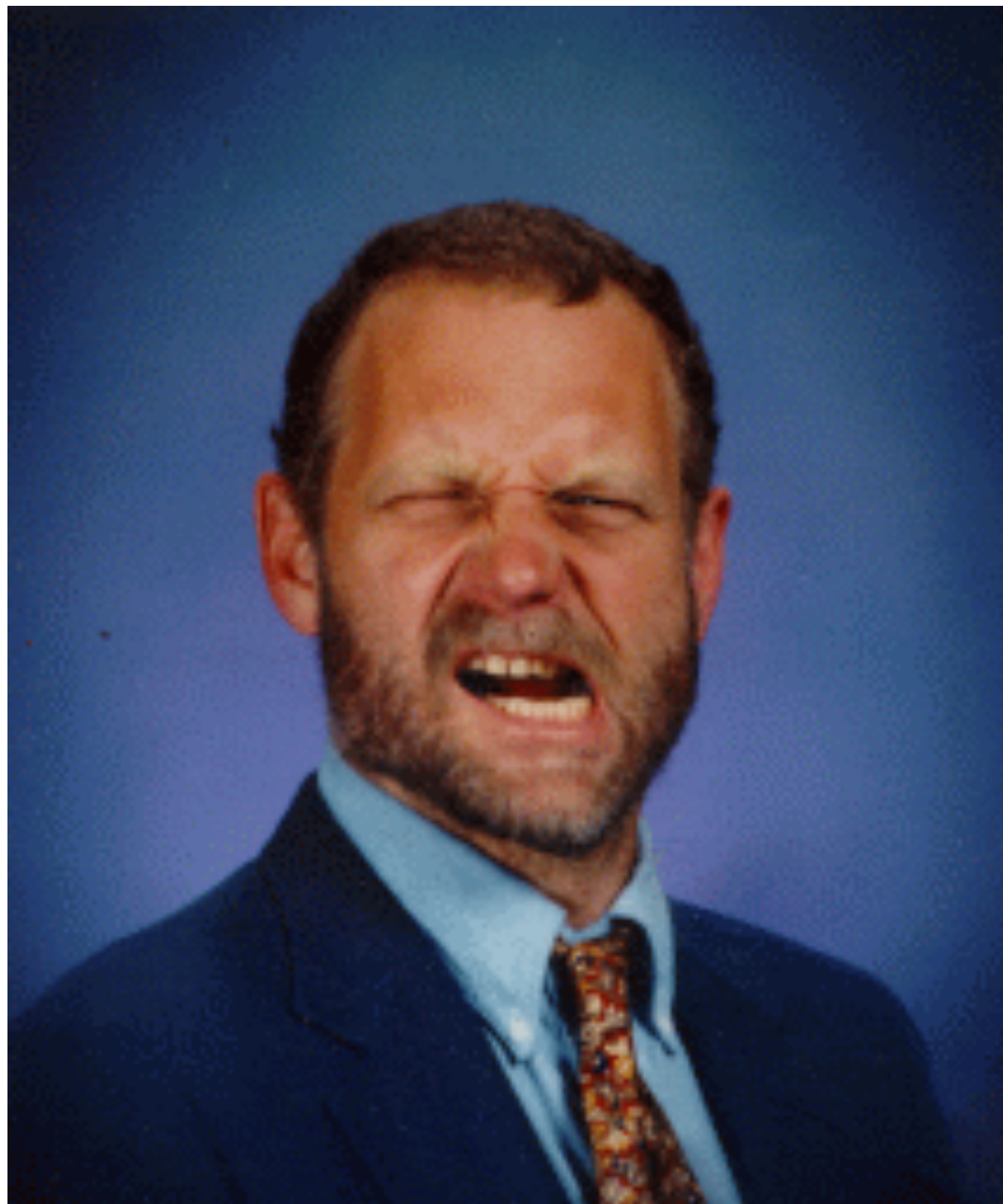


Dexter Kozen

```
p, q ::=
    0           // abort
  | 1           // skip
  | b, ~b       // Boolean tests
  | a           // action
  | p + q       // choice
  | p · q       // sequential composition
  | p*          // iteration
```

Double meaning
+ = disjunction
· = conjunction

Kleene Algebra with Tests

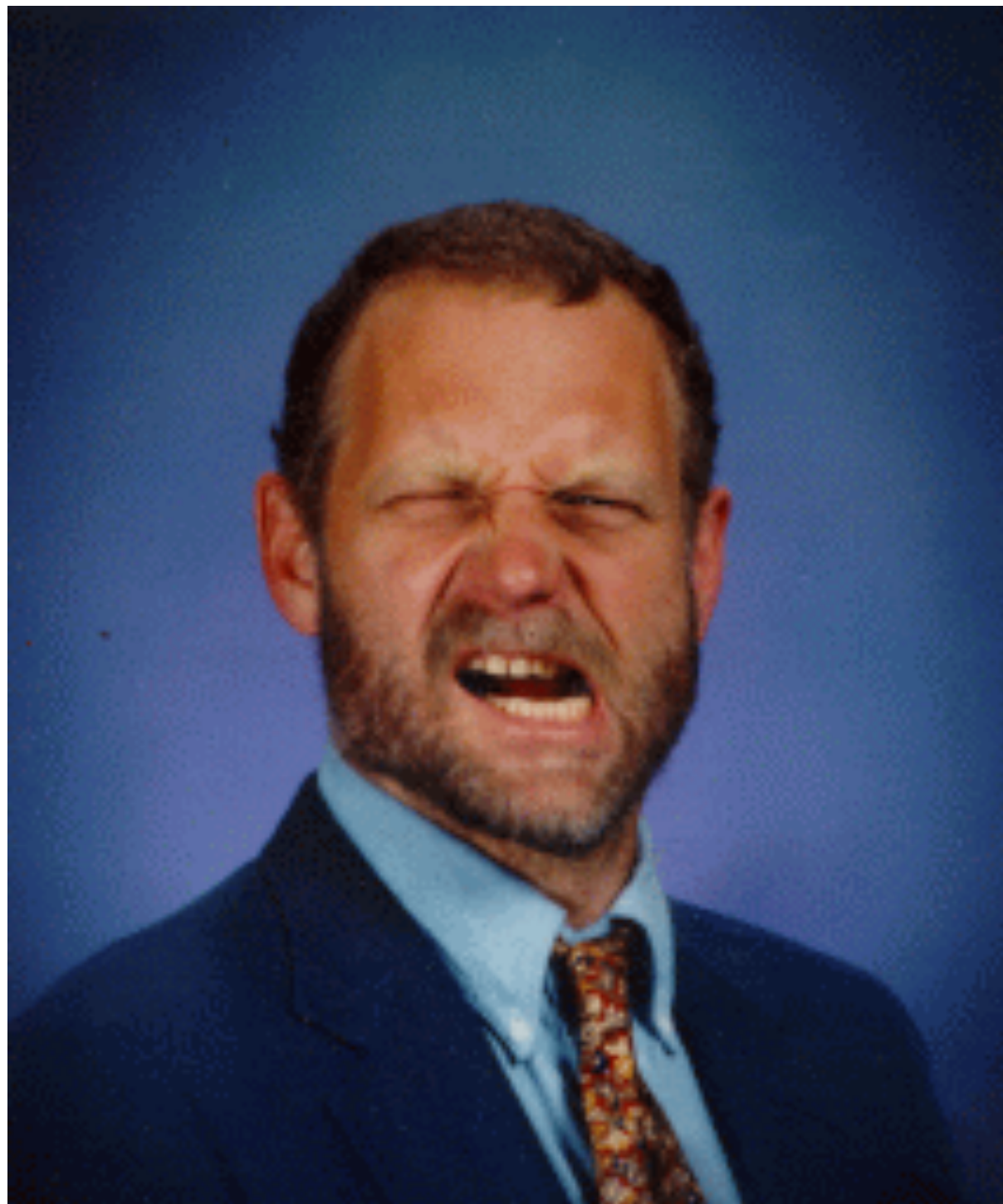


Dexter Kozen

```
p, q ::=
    0           // abort
  | 1           // skip
  | b, ~b       // Boolean tests
  | a           // action
  | p + q       // choice
  | p · q       // sequential composition
  | p*          // iteration
```

```
if b then p else q = b · p + ~b · q
```

Kleene Algebra with Tests



Dexter Kozen

```
p, q ::=
    0           // abort
  | 1           // skip
  | b, ~b       // Boolean tests
  | a           // action
  | p + q       // choice
  | p · q       // sequential composition
  | p*          // iteration
```

```
if b then p else q = b · p + ~b · q
while b do p = (b · p)* · ~b
```

Kleene Algebra with Tests

Kleene Algebra with Tests

A Kleene Algebra with Tests is an algebraic structure

Kleene Algebra with Tests

A Kleene Algebra with Tests is an algebraic structure
 $\langle K, B, +, \cdot, *, \sim, 0, 1 \rangle$

Kleene Algebra with Tests

A Kleene Algebra with Tests is an algebraic structure

$\langle K, B, +, \cdot, *, \sim, 0, 1 \rangle$

where

Kleene Algebra with Tests

A Kleene Algebra with Tests is an algebraic structure

$\langle K, B, +, \cdot, *, \sim, 0, 1 \rangle$

where

- $B \subseteq K$

Kleene Algebra with Tests

A Kleene Algebra with Tests is an algebraic structure

$$\langle K, B, +, \cdot, *, \sim, 0, 1 \rangle$$

where

- $B \subseteq K$
- $\langle K, +, \cdot, *, 0, 1 \rangle$ is a Kleene Algebra

Kleene Algebra with Tests

A Kleene Algebra with Tests is an algebraic structure

$$\langle K, B, +, \cdot, *, \sim, 0, 1 \rangle$$

where

- $B \subseteq K$
- $\langle K, +, \cdot, *, 0, 1 \rangle$ is a Kleene Algebra
- $\langle B, +, \cdot, \sim, 0, 1 \rangle$ is a Boolean Algebra

Kleene Algebra with Tests: Fundamental Results

Kleene Algebra with Tests: Fundamental Results

└ **Axiomatization**

Can we characterize the set of valid program equalities/refinements?

Kleene Algebra with Tests: Fundamental Results

⊢ **Axiomatization**

Can we characterize the set of valid program equalities/refinements?

✓ **Verification**

Can we automatically check whether two expressions are equivalent?

Kleene Algebra with Tests: Fundamental Results

└ Axiomatization

Can we characterize the set of valid program equalities/refinements?

Yes: Kozen (1997, 2001)

✓ Verification

Can we automatically check whether two expressions are equivalent?

Kleene Algebra with Tests: Fundamental Results

└ Axiomatization

Can we characterize the set of valid program equalities/refinements?

Yes: Kozen (1997, 2001)

✓ Verification

Can we automatically check whether two expressions are equivalent?

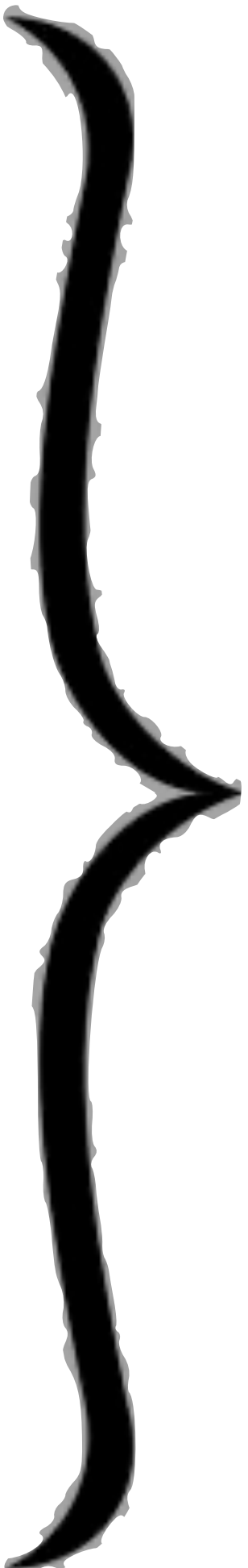
Yes: Chen and Pucella (2004); Kozen (2010); Pous (2016)

Kleene Algebra with Tests: Axiomatization

Z: $p + 0 \equiv p$
C: $p + q \equiv q + p$
A: $(p + q) + r \equiv p + (q + r)$
I: $p + p \equiv p$

SN: $p \cdot 1 \equiv 1 \cdot p \equiv p$
SZ: $p \cdot 0 \equiv 0 \cdot p \equiv 0$
SA: $p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$
SDl: $p \cdot (q + r) \equiv p \cdot q + p \cdot r$
SDr: $(p + q) \cdot r \equiv p \cdot r + q \cdot r$

FP: $1 + p \cdot p^* \leq p^*$
 $1 + p^* \cdot p \leq p^*$
LFP: $q + p \cdot x \leq x \rightarrow p^* \cdot q \leq x$
 $q + x \cdot p \leq x \rightarrow q \cdot p^* \leq x$



KA

Kleene Algebra with Tests: Axiomatization

Z: $p + 0 \equiv p$
C: $p + q \equiv q + p$
A: $(p + q) + r \equiv p + (q + r)$
I: $p + p \equiv p$

SN: $p \cdot 1 \equiv 1 \cdot p \equiv p$
SZ: $p \cdot 0 \equiv 0 \cdot p \equiv 0$
SA: $p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$
SDl: $p \cdot (q + r) \equiv p \cdot q + p \cdot r$
SDr: $(p + q) \cdot r \equiv p \cdot r + q \cdot r$

FP: $1 + p \cdot p^* \leq p^*$
 $1 + p^* \cdot p \leq p^*$
LFP: $q + p \cdot x \leq x \rightarrow p^* \cdot q \leq x$
 $q + x \cdot p \leq x \rightarrow q \cdot p^* \leq x$

KA

$a + 0 \equiv a$

Kleene Algebra with Tests: Axiomatization

Z: $p + 0 \equiv p$
C: $p + q \equiv q + p$
A: $(p + q) + r \equiv p + (q + r)$
I: $p + p \equiv p$

SN: $p \cdot 1 \equiv 1 \cdot p \equiv p$
SZ: $p \cdot 0 \equiv 0 \cdot p \equiv 0$
SA: $p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$
SDl: $p \cdot (q + r) \equiv p \cdot q + p \cdot r$
SDr: $(p + q) \cdot r \equiv p \cdot r + q \cdot r$

FP: $1 + p \cdot p^* \leq p^*$
 $1 + p^* \cdot p \leq p^*$
LFP: $q + p \cdot x \leq x \rightarrow p^* \cdot q \leq x$
 $q + x \cdot p \leq x \rightarrow q \cdot p^* \leq x$

} KA

$a + 0 \equiv a$
 $a + 1 \equiv 1$

Kleene Algebra with Tests: Axiomatization

Z: $p + 0 \equiv p$
C: $p + q \equiv q + p$
A: $(p + q) + r \equiv p + (q + r)$
I: $p + p \equiv p$

SN: $p \cdot 1 \equiv 1 \cdot p \equiv p$
SZ: $p \cdot 0 \equiv 0 \cdot p \equiv 0$
SA: $p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$
SDl: $p \cdot (q + r) \equiv p \cdot q + p \cdot r$
SDr: $(p + q) \cdot r \equiv p \cdot r + q \cdot r$

FP: $1 + p \cdot p^* \leq p^*$
 $1 + p^* \cdot p \leq p^*$
LFP: $q + p \cdot x \leq x \rightarrow p^* \cdot q \leq x$
 $q + x \cdot p \leq x \rightarrow q \cdot p^* \leq x$

} KA

$a + 0 \equiv a$
 $a + 1 \equiv 1$
 $a + b \equiv b + a$

Kleene Algebra with Tests: Axiomatization

Z: $p + 0 \equiv p$
C: $p + q \equiv q + p$
A: $(p + q) + r \equiv p + (q + r)$
I: $p + p \equiv p$

SN: $p \cdot 1 \equiv 1 \cdot p \equiv p$
SZ: $p \cdot 0 \equiv 0 \cdot p \equiv 0$
SA: $p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$
SDl: $p \cdot (q + r) \equiv p \cdot q + p \cdot r$
SDr: $(p + q) \cdot r \equiv p \cdot r + q \cdot r$

FP: $1 + p \cdot p^* \leq p^*$
 $1 + p^* \cdot p \leq p^*$
LFP: $q + p \cdot x \leq x \rightarrow p^* \cdot q \leq x$
 $q + x \cdot p \leq x \rightarrow q \cdot p^* \leq x$

} KA

$a + 0 \equiv a$
 $a + 1 \equiv 1$
 $a + b \equiv b + a$
 $a + a \equiv a$

Kleene Algebra with Tests: Axiomatization

Z: $p + 0 \equiv p$
C: $p + q \equiv q + p$
A: $(p + q) + r \equiv p + (q + r)$
I: $p + p \equiv p$

SN: $p \cdot 1 \equiv 1 \cdot p \equiv p$
SZ: $p \cdot 0 \equiv 0 \cdot p \equiv 0$
SA: $p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$
SDl: $p \cdot (q + r) \equiv p \cdot q + p \cdot r$
SDr: $(p + q) \cdot r \equiv p \cdot r + q \cdot r$

FP: $1 + p \cdot p^* \leq p^*$
 $1 + p^* \cdot p \leq p^*$
LFP: $q + p \cdot x \leq x \rightarrow p^* \cdot q \leq x$
 $q + x \cdot p \leq x \rightarrow q \cdot p^* \leq x$

} KA

$a + 0 \equiv a$
 $a + 1 \equiv 1$
 $a + b \equiv b + a$
 $a + a \equiv a$
 $(a + b) + c \equiv a + (b + c)$

Kleene Algebra with Tests: Axiomatization

Z: $p + 0 \equiv p$
C: $p + q \equiv q + p$
A: $(p + q) + r \equiv p + (q + r)$
I: $p + p \equiv p$

SN: $p \cdot 1 \equiv 1 \cdot p \equiv p$
SZ: $p \cdot 0 \equiv 0 \cdot p \equiv 0$
SA: $p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$
SDl: $p \cdot (q + r) \equiv p \cdot q + p \cdot r$
SDr: $(p + q) \cdot r \equiv p \cdot r + q \cdot r$

FP: $1 + p \cdot p^* \leq p^*$
 $1 + p^* \cdot p \leq p^*$
LFP: $q + p \cdot x \leq x \rightarrow p^* \cdot q \leq x$
 $q + x \cdot p \leq x \rightarrow q \cdot p^* \leq x$

KA

$a + 0 \equiv a$
 $a + 1 \equiv 1$
 $a + b \equiv b + a$
 $a + a \equiv a$
 $(a + b) + c \equiv a + (b + c)$
 $\sim(b+c) \equiv \sim b \cdot \sim c$

Kleene Algebra with Tests: Axiomatization

Z: $p + 0 \equiv p$
C: $p + q \equiv q + p$
A: $(p + q) + r \equiv p + (q + r)$
I: $p + p \equiv p$

SN: $p \cdot 1 \equiv 1 \cdot p \equiv p$
SZ: $p \cdot 0 \equiv 0 \cdot p \equiv 0$
SA: $p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$
SDl: $p \cdot (q + r) \equiv p \cdot q + p \cdot r$
SDr: $(p + q) \cdot r \equiv p \cdot r + q \cdot r$

FP: $1 + p \cdot p^* \leq p^*$
 $1 + p^* \cdot p \leq p^*$
LFP: $q + p \cdot x \leq x \rightarrow p^* \cdot q \leq x$
 $q + x \cdot p \leq x \rightarrow q \cdot p^* \leq x$

KA

$a + 0 \equiv a$
 $a + 1 \equiv 1$
 $a + b \equiv b + a$
 $a + a \equiv a$
 $(a + b) + c \equiv a + (b + c)$
 $\sim(b+c) \equiv \sim b \cdot \sim c$
 $\sim\sim a \equiv a$

Kleene Algebra with Tests: Axiomatization

Z: $p + 0 \equiv p$
C: $p + q \equiv q + p$
A: $(p + q) + r \equiv p + (q + r)$
I: $p + p \equiv p$

SN: $p \cdot 1 \equiv 1 \cdot p \equiv p$
SZ: $p \cdot 0 \equiv 0 \cdot p \equiv 0$
SA: $p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$
SDl: $p \cdot (q + r) \equiv p \cdot q + p \cdot r$
SDr: $(p + q) \cdot r \equiv p \cdot r + q \cdot r$

FP: $1 + p \cdot p^* \leq p^*$
 $1 + p^* \cdot p \leq p^*$
LFP: $q + p \cdot x \leq x \rightarrow p^* \cdot q \leq x$
 $q + x \cdot p \leq x \rightarrow q \cdot p^* \leq x$

KA

$a + 0 \equiv a$
 $a + 1 \equiv 1$
 $a + b \equiv b + a$
 $a + a \equiv a$
 $(a + b) + c \equiv a + (b + c)$
 $\sim(b+c) \equiv \sim b \cdot \sim c$
 $\sim\sim a \equiv a$
 $\sim(a \cdot b) \equiv \sim a + \sim b$

Kleene Algebra with Tests: Axiomatization

Z: $p + 0 \equiv p$
C: $p + q \equiv q + p$
A: $(p + q) + r \equiv p + (q + r)$
I: $p + p \equiv p$

SN: $p \cdot 1 \equiv 1 \cdot p \equiv p$
SZ: $p \cdot 0 \equiv 0 \cdot p \equiv 0$
SA: $p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$
SDl: $p \cdot (q + r) \equiv p \cdot q + p \cdot r$
SDr: $(p + q) \cdot r \equiv p \cdot r + q \cdot r$

FP: $1 + p \cdot p^* \leq p^*$
 $1 + p^* \cdot p \leq p^*$
LFP: $q + p \cdot x \leq x \rightarrow p^* \cdot q \leq x$
 $q + x \cdot p \leq x \rightarrow q \cdot p^* \leq x$

KA

$a + 0 \equiv a$
 $a + 1 \equiv 1$
 $a + b \equiv b + a$
 $a + a \equiv a$
 $(a + b) + c \equiv a + (b + c)$
 $\sim(b+c) \equiv \sim b \cdot \sim c$
 $\sim\sim a \equiv a$
 $\sim(a \cdot b) \equiv \sim a + \sim b$

$a \cdot 1 \equiv a$

Kleene Algebra with Tests: Axiomatization

Z: $p + 0 \equiv p$
C: $p + q \equiv q + p$
A: $(p + q) + r \equiv p + (q + r)$
I: $p + p \equiv p$

SN: $p \cdot 1 \equiv 1 \cdot p \equiv p$
SZ: $p \cdot 0 \equiv 0 \cdot p \equiv 0$
SA: $p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$
SDl: $p \cdot (q + r) \equiv p \cdot q + p \cdot r$
SDr: $(p + q) \cdot r \equiv p \cdot r + q \cdot r$

FP: $1 + p \cdot p^* \leq p^*$
 $1 + p^* \cdot p \leq p^*$
LFP: $q + p \cdot x \leq x \rightarrow p^* \cdot q \leq x$
 $q + x \cdot p \leq x \rightarrow q \cdot p^* \leq x$

KA

$a + 0 \equiv a$
 $a + 1 \equiv 1$
 $a + b \equiv b + a$
 $a + a \equiv a$
 $(a + b) + c \equiv a + (b + c)$
 $\sim(b+c) \equiv \sim b \cdot \sim c$
 $\sim\sim a \equiv a$
 $\sim(a \cdot b) \equiv \sim a + \sim b$

$a \cdot 1 \equiv a$
 $a \cdot 0 \equiv 0$

Kleene Algebra with Tests: Axiomatization

Z: $p + 0 \equiv p$
C: $p + q \equiv q + p$
A: $(p + q) + r \equiv p + (q + r)$
I: $p + p \equiv p$

SN: $p \cdot 1 \equiv 1 \cdot p \equiv p$
SZ: $p \cdot 0 \equiv 0 \cdot p \equiv 0$
SA: $p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$
SDl: $p \cdot (q + r) \equiv p \cdot q + p \cdot r$
SDr: $(p + q) \cdot r \equiv p \cdot r + q \cdot r$

FP: $1 + p \cdot p^* \leq p^*$
 $1 + p^* \cdot p \leq p^*$
LFP: $q + p \cdot x \leq x \rightarrow p^* \cdot q \leq x$
 $q + x \cdot p \leq x \rightarrow q \cdot p^* \leq x$

KA

$a + 0 \equiv a$
 $a + 1 \equiv 1$
 $a + b \equiv b + a$
 $a + a \equiv a$
 $(a + b) + c \equiv a + (b + c)$
 $\sim(b+c) \equiv \sim b \cdot \sim c$
 $\sim\sim a \equiv a$
 $\sim(a \cdot b) \equiv \sim a + \sim b$

$a \cdot 1 \equiv a$
 $a \cdot 0 \equiv 0$
 $a \cdot b \equiv b \cdot a$

Kleene Algebra with Tests: Axiomatization

Z: $p + 0 \equiv p$
 C: $p + q \equiv q + p$
 A: $(p + q) + r \equiv p + (q + r)$
 I: $p + p \equiv p$

SN: $p \cdot 1 \equiv 1 \cdot p \equiv p$
 SZ: $p \cdot 0 \equiv 0 \cdot p \equiv 0$
 SA: $p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$
 SDl: $p \cdot (q + r) \equiv p \cdot q + p \cdot r$
 SDr: $(p + q) \cdot r \equiv p \cdot r + q \cdot r$

FP: $1 + p \cdot p^* \leq p^*$
 $1 + p^* \cdot p \leq p^*$
 LFP: $q + p \cdot x \leq x \rightarrow p^* \cdot q \leq x$
 $q + x \cdot p \leq x \rightarrow q \cdot p^* \leq x$

} KA

$a + 0 \equiv a$
 $a + 1 \equiv 1$
 $a + b \equiv b + a$
 $a + a \equiv a$
 $(a + b) + c \equiv a + (b + c)$
 $\sim(b+c) \equiv \sim b \cdot \sim c$
 $\sim\sim a \equiv a$
 $\sim(a \cdot b) \equiv \sim a + \sim b$

$a \cdot 1 \equiv a$
 $a \cdot 0 \equiv 0$
 $a \cdot b \equiv b \cdot a$
 $a \cdot (b \cdot c) \equiv (a \cdot b) \cdot c$

Kleene Algebra with Tests: Axiomatization

Z: $p + 0 \equiv p$
 C: $p + q \equiv q + p$
 A: $(p + q) + r \equiv p + (q + r)$
 I: $p + p \equiv p$

SN: $p \cdot 1 \equiv 1 \cdot p \equiv p$
 SZ: $p \cdot 0 \equiv 0 \cdot p \equiv 0$
 SA: $p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$
 SDl: $p \cdot (q + r) \equiv p \cdot q + p \cdot r$
 SDr: $(p + q) \cdot r \equiv p \cdot r + q \cdot r$

FP: $1 + p \cdot p^* \leq p^*$
 $1 + p^* \cdot p \leq p^*$
 LFP: $q + p \cdot x \leq x \rightarrow p^* \cdot q \leq x$
 $q + x \cdot p \leq x \rightarrow q \cdot p^* \leq x$

} KA

$a + 0 \equiv a$
 $a + 1 \equiv 1$
 $a + b \equiv b + a$
 $a + a \equiv a$
 $(a + b) + c \equiv a + (b + c)$
 $\sim(b+c) \equiv \sim b \cdot \sim c$
 $\sim\sim a \equiv a$
 $\sim(a \cdot b) \equiv \sim a + \sim b$

$a \cdot 1 \equiv a$
 $a \cdot 0 \equiv 0$
 $a \cdot b \equiv b \cdot a$
 $a \cdot (b \cdot c) \equiv (a \cdot b) \cdot c$
 $a \cdot (b + c) \equiv a \cdot b + a \cdot c$

Kleene Algebra with Tests: Axiomatization

Z: $p + 0 \equiv p$
C: $p + q \equiv q + p$
A: $(p + q) + r \equiv p + (q + r)$
I: $p + p \equiv p$

SN: $p \cdot 1 \equiv 1 \cdot p \equiv p$
SZ: $p \cdot 0 \equiv 0 \cdot p \equiv 0$
SA: $p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$
SDl: $p \cdot (q + r) \equiv p \cdot q + p \cdot r$
SDr: $(p + q) \cdot r \equiv p \cdot r + q \cdot r$

FP: $1 + p \cdot p^* \leq p^*$
 $1 + p^* \cdot p \leq p^*$
LFP: $q + p \cdot x \leq x \rightarrow p^* \cdot q \leq x$
 $q + x \cdot p \leq x \rightarrow q \cdot p^* \leq x$

KA

$a + 0 \equiv a$
 $a + 1 \equiv 1$
 $a + b \equiv b + a$
 $a + a \equiv a$
 $(a + b) + c \equiv a + (b + c)$
 $\sim(b+c) \equiv \sim b \cdot \sim c$
 $\sim\sim a \equiv a$
 $\sim(a \cdot b) \equiv \sim a + \sim b$

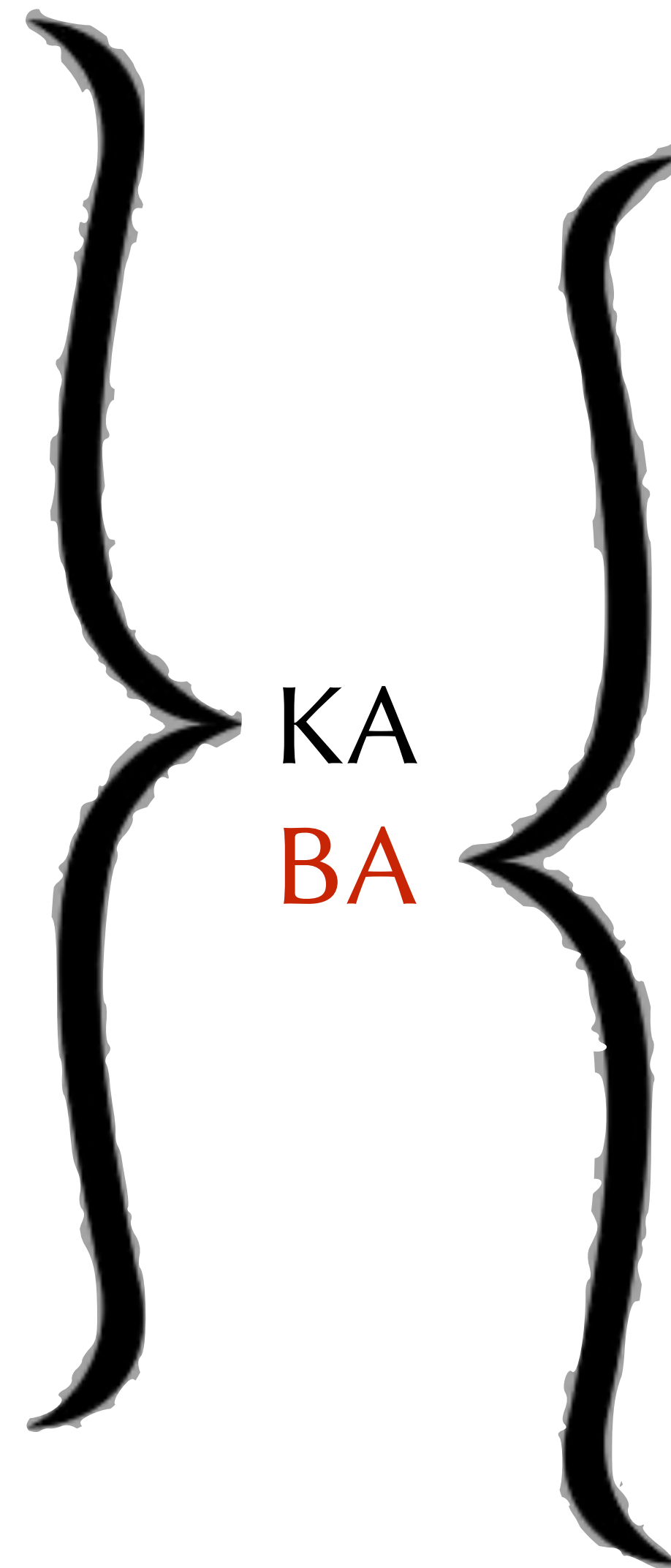
$a \cdot 1 \equiv a$
 $a \cdot 0 \equiv 0$
 $a \cdot b \equiv b \cdot a$
 $a \cdot (b \cdot c) \equiv (a \cdot b) \cdot c$
 $a \cdot (b + c) \equiv a \cdot b + a \cdot c$
 $(a + b) \cdot c \equiv a \cdot c + b \cdot c$

Kleene Algebra with Tests: Axiomatization

$Z: p + 0 \equiv p$
 $C: p + q \equiv q + p$
 $A: (p + q) + r \equiv p + (q + r)$
 $I: p + p \equiv p$

$SN: p \cdot 1 \equiv 1 \cdot p \equiv p$
 $SZ: p \cdot 0 \equiv 0 \cdot p \equiv 0$
 $SA: p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$
 $SDl: p \cdot (q + r) \equiv p \cdot q + p \cdot r$
 $SDr: (p + q) \cdot r \equiv p \cdot r + q \cdot r$

$FP: 1 + p \cdot p^* \leq p^*$
 $1 + p^* \cdot p \leq p^*$
 $LFP: q + p \cdot x \leq x \rightarrow p^* \cdot q \leq x$
 $q + x \cdot p \leq x \rightarrow q \cdot p^* \leq x$



$a + 0 \equiv a$
 $a + 1 \equiv 1$
 $a + b \equiv b + a$
 $a + a \equiv a$
 $(a + b) + c \equiv a + (b + c)$
 $\sim(b+c) \equiv \sim b \cdot \sim c$
 $\sim\sim a \equiv a$
 $\sim(a \cdot b) \equiv \sim a + \sim b$

$a \cdot 1 \equiv a$
 $a \cdot 0 \equiv 0$
 $a \cdot b \equiv b \cdot a$
 $a \cdot (b \cdot c) \equiv (a \cdot b) \cdot c$
 $a \cdot (b + c) \equiv a \cdot b + a \cdot c$
 $(a + b) \cdot c \equiv a \cdot c + b \cdot c$

Kleene Algebra with Tests: Semantics

Kleene Algebra with Tests: Semantics

Idea: strings alternating between tests and actions

Kleene Algebra with Tests: Semantics

Idea: strings alternating between tests and actions

Formally:

Kleene Algebra with Tests: Semantics

Idea: strings alternating between tests and actions

Formally:

- Σ : action symbols

Kleene Algebra with Tests: Semantics

Idea: strings alternating between tests and actions

Formally:

- Σ : action symbols
- T : test symbols

Kleene Algebra with Tests: Semantics

Idea: strings alternating between tests and actions

Formally:

- Σ : action symbols
- T : test symbols
- At : atoms of free Boolean algebra generated by T

Kleene Algebra with Tests: Semantics

Idea: strings alternating between tests and actions

Formally:

- Σ : action symbols
- T : test symbols
- At : atoms of free Boolean algebra generated by T
- Guarded strings: elements of $(At \cdot \Sigma)^* \cdot At$

Kleene Algebra with Tests: Semantics

Idea: strings alternating between tests and actions

Formally:

- Σ : action symbols
- T : test symbols
- At : atoms of free Boolean algebra generated by T
- Guarded strings: elements of $(At \cdot \Sigma)^* \cdot At$
- Example: $\alpha_0 \cdot p_0 \cdot \alpha_1 \cdot p_1 \cdot \dots \cdot p_{n-1} \cdot \alpha_n$

Kleene Algebra with Tests: Semantics

Idea: strings alternating between tests and actions

Formally:

- Σ : action symbols
- T : test symbols
- At : atoms of free Boolean algebra generated by T
- Guarded strings: elements of $(At \cdot \Sigma)^* \cdot At$
- Example: $\alpha_0 \cdot p_0 \cdot \alpha_1 \cdot p_1 \cdot \dots \cdot p_{n-1} \cdot \alpha_n$

state of program
variables

Kleene Algebra with Tests: Semantics

Kleene Algebra with Tests: Semantics

$$\llbracket p \rrbracket \subseteq (At \cdot \Sigma)^* \cdot At$$

Kleene Algebra with Tests: Semantics

$$\llbracket p \rrbracket \subseteq (At \cdot \Sigma)^* \cdot At$$

Kleene Algebra with Tests: Semantics

$$\llbracket p \rrbracket \subseteq (At \cdot \Sigma)^* \cdot At$$

$$\llbracket 0 \rrbracket = \{ \}$$

Kleene Algebra with Tests: Semantics

$$\llbracket p \rrbracket \subseteq (\text{At} \cdot \Sigma)^* \cdot \text{At}$$

$$\llbracket 0 \rrbracket = \{ \}$$

$$\llbracket 1 \rrbracket = \text{At}$$

Kleene Algebra with Tests: Semantics

$$\llbracket p \rrbracket \subseteq (\text{At} \cdot \Sigma)^* \cdot \text{At}$$

$$\llbracket 0 \rrbracket = \{ \}$$

$$\llbracket 1 \rrbracket = \text{At}$$

$$\llbracket p \rrbracket = \{ \alpha \cdot p \cdot \beta \mid \alpha, \beta \in \text{At} \}$$

Kleene Algebra with Tests: Semantics

$$\llbracket p \rrbracket \subseteq (\text{At} \cdot \Sigma)^* \cdot \text{At}$$

$$\llbracket 0 \rrbracket = \{ \}$$

$$\llbracket 1 \rrbracket = \text{At}$$

$$\llbracket p \rrbracket = \{ \alpha \cdot p \cdot \beta \mid \alpha, \beta \in \text{At} \}$$

$$\llbracket a \rrbracket = \{ \alpha \mid \alpha \leq a \}$$

Kleene Algebra with Tests: Semantics

$$\llbracket p \rrbracket \subseteq (\text{At} \cdot \Sigma)^* \cdot \text{At}$$

$$\llbracket 0 \rrbracket = \{ \}$$

$$\llbracket 1 \rrbracket = \text{At}$$

$$\llbracket p \rrbracket = \{ \alpha \cdot p \cdot \beta \mid \alpha, \beta \in \text{At} \}$$

$$\llbracket a \rrbracket = \{ \alpha \mid \alpha \leq a \}$$

$$\llbracket \sim a \rrbracket = \text{At} \setminus \{ \alpha \mid \alpha \leq a \}$$

Kleene Algebra with Tests: Semantics

$$\llbracket p \rrbracket \subseteq (\text{At} \cdot \Sigma)^* \cdot \text{At}$$

$$\llbracket 0 \rrbracket = \{ \}$$

$$\llbracket 1 \rrbracket = \text{At}$$

$$\llbracket p \rrbracket = \{ \alpha \cdot p \cdot \beta \mid \alpha, \beta \in \text{At} \}$$

$$\llbracket a \rrbracket = \{ \alpha \mid \alpha \leq a \}$$

$$\llbracket \sim a \rrbracket = \text{At} \setminus \{ \alpha \mid \alpha \leq a \}$$

$$\llbracket p \cdot q \rrbracket = \{ x \cdot \alpha \cdot y \mid x \cdot \alpha \in \llbracket p \rrbracket, \alpha \cdot y \in \llbracket q \rrbracket \}$$

Kleene Algebra with Tests: Semantics

$$\llbracket p \rrbracket \subseteq (\text{At} \cdot \Sigma)^* \cdot \text{At}$$

$$\llbracket 0 \rrbracket = \{ \}$$

$$\llbracket 1 \rrbracket = \text{At}$$

$$\llbracket p \rrbracket = \{ \alpha \cdot p \cdot \beta \mid \alpha, \beta \in \text{At} \}$$

$$\llbracket a \rrbracket = \{ \alpha \mid \alpha \leq a \}$$

$$\llbracket \sim a \rrbracket = \text{At} \setminus \{ \alpha \mid \alpha \leq a \}$$

$$\llbracket p \cdot q \rrbracket = \{ x \cdot \alpha \cdot y \mid x \cdot \alpha \in \llbracket p \rrbracket, \alpha \cdot y \in \llbracket q \rrbracket \}$$

guarded concatenation

$$L.K = \{ x\alpha y \mid x\alpha \in L, \alpha y \in K \}$$

$$\llbracket p \cdot q \rrbracket = \llbracket p \rrbracket \cdot \llbracket q \rrbracket$$

Kleene Algebra with Tests: Semantics

$$\llbracket p \rrbracket \subseteq (\text{At} \cdot \Sigma)^* \cdot \text{At}$$

$$\llbracket 0 \rrbracket = \{ \}$$

$$\llbracket 1 \rrbracket = \text{At}$$

$$\llbracket p \rrbracket = \{ \alpha \cdot p \cdot \beta \mid \alpha, \beta \in \text{At} \}$$

$$\llbracket a \rrbracket = \{ \alpha \mid \alpha \leq a \}$$

$$\llbracket \sim a \rrbracket = \text{At} \setminus \{ \alpha \mid \alpha \leq a \}$$

$$\llbracket p \cdot q \rrbracket = \{ x \cdot \alpha \cdot y \mid x \cdot \alpha \in \llbracket p \rrbracket, \alpha \cdot y \in \llbracket q \rrbracket \}$$

$$\llbracket p + q \rrbracket = \llbracket p \rrbracket \cup \llbracket q \rrbracket$$

guarded concatenation

$$L.K = \{ x\alpha y \mid x\alpha \in L, \alpha y \in K \}$$

$$\llbracket p \cdot q \rrbracket = \llbracket p \rrbracket \cdot \llbracket q \rrbracket$$

Kleene Algebra with Tests: Semantics

$$\llbracket p \rrbracket \subseteq (\text{At} \cdot \Sigma)^* \cdot \text{At}$$

$$\llbracket 0 \rrbracket = \{ \}$$

$$\llbracket 1 \rrbracket = \text{At}$$

$$\llbracket p \rrbracket = \{ \alpha \cdot p \cdot \beta \mid \alpha, \beta \in \text{At} \}$$

$$\llbracket a \rrbracket = \{ \alpha \mid \alpha \leq a \}$$

$$\llbracket \sim a \rrbracket = \text{At} \setminus \{ \alpha \mid \alpha \leq a \}$$

$$\llbracket p \cdot q \rrbracket = \{ x \cdot \alpha \cdot y \mid x \cdot \alpha \in \llbracket p \rrbracket, \alpha \cdot y \in \llbracket q \rrbracket \}$$

$$\llbracket p + q \rrbracket = \llbracket p \rrbracket \cup \llbracket q \rrbracket$$

$$\llbracket p^* \rrbracket = \bigcup_i \llbracket p \rrbracket^i$$

guarded concatenation

$$L.K = \{ x\alpha y \mid x\alpha \in L, \alpha y \in K \}$$

$$\llbracket p \cdot q \rrbracket = \llbracket p \rrbracket \cdot \llbracket q \rrbracket$$

First example

```
while b {  
    p;  
    while c do q  
}
```

?
≡

```
if b then {  
    p;  
    while b or c do  
        if c then q else p  
}
```

First example

```
while b {  
    p;  
    while c do q  
}
```

?
≡

```
if b then {  
    p;  
    while b or c do  
        if c then q else p  
}
```

Now: can think of programs as generating guarded strings!

Proof (axiomatic)

```
while b {  
    p;  
    while c do q  
}
```

?
≡

```
if b then {  
    p;  
    while b or c do  
        if c then q else p  
}
```

Proof (axiomatic)

```
while b {  
    p;  
    while c do q  
}
```

?
≡

```
if b then {  
    p;  
    while b or c do  
        if c then q else p  
}
```

$(b \cdot p \cdot (c \cdot q)^* \cdot \sim c)^* \cdot \sim b$

Proof (axiomatic)

```
while b {  
    p;  
    while c do q  
}
```

?
≡

```
if b then {  
    p;  
    while b or c do  
        if c then q else p  
}
```

$(b \cdot p \cdot (c \cdot q)^* \cdot \sim c)^* \cdot \sim b$

$b \cdot p \cdot ((b+c) \cdot (c \cdot q + \sim c \cdot p))^* \cdot \sim (b+c) + \sim b$

Proof (axiomatic)

Proof (axiomatic)

$$b \cdot p \cdot ((b+c) \cdot (c \cdot q + \sim c \cdot p)) * \sim(b+c) + \sim b$$

Proof (axiomatic)

$$b \cdot p \cdot ((b+c) \cdot (c \cdot q + \sim c \cdot p)) * \sim(b+c) + \sim b$$

$$\equiv \{ \text{Distributivity} + \text{Boolean Algebra} \}$$

Proof (axiomatic)

$$b \cdot p \cdot ((b+c) \cdot (c \cdot q + \sim c \cdot p)) * \sim(b+c) + \sim b$$

$\equiv$ { Distributivity + Boolean Algebra }

$$b \cdot p \cdot (b \cdot c \cdot q + b \sim c \cdot p + c \cdot q + c \sim c \cdot q) * \sim b \cdot \sim c + \sim b$$

Proof (axiomatic)

$$b \cdot p \cdot ((b+c) \cdot (c \cdot q + \sim c \cdot p)) * \sim(b+c) + \sim b$$

$$\equiv \{ \text{Distributivity} + \text{Boolean Algebra} \}$$

$$b \cdot p \cdot (b \cdot c \cdot q + b \sim c \cdot p + c \cdot q + c \sim c \cdot q) * \sim b \cdot \sim c + \sim b$$

$$\equiv \{ \text{Commutativity} + \text{Distributivity} + \text{Boolean Algebra} \}$$

Proof (axiomatic)

$$b \cdot p \cdot ((b+c) \cdot (c \cdot q + \sim c \cdot p)) * \sim(b+c) + \sim b$$

$$\equiv \{ \text{Distributivity} + \text{Boolean Algebra} \}$$

$$b \cdot p \cdot (b \cdot c \cdot q + b \sim c \cdot p + c \cdot q + c \sim c \cdot q) * \sim b \cdot \sim c + \sim b$$

$$\equiv \{ \text{Commutativity} + \text{Distributivity} + \text{Boolean Algebra} \}$$

$$b \cdot p \cdot (c \cdot q + \sim c \cdot b \cdot p) * \sim b \sim c + \sim b$$

Proof (axiomatic)

$$b \cdot p \cdot ((b+c) \cdot (c \cdot q + \sim c \cdot p))^* \cdot \sim(b+c) + \sim b$$

$$\equiv \{ \text{Distributivity} + \text{Boolean Algebra} \}$$

$$b \cdot p \cdot (b \cdot c \cdot q + b \sim c \cdot p + c \cdot q + c \sim c \cdot q)^* \cdot \sim b \cdot \sim c + \sim b$$

$$\equiv \{ \text{Commutativity} + \text{Distributivity} + \text{Boolean Algebra} \}$$

$$b \cdot p \cdot (c \cdot q + \sim c \cdot b \cdot p)^* \cdot \sim b \sim c + \sim b$$

$$\equiv \{ \text{Denesting: } (e+f)^* = (e^* \cdot f)^* \cdot e^* \}$$

Proof (axiomatic)

$$b \cdot p \cdot ((b+c) \cdot (c \cdot q + \sim c \cdot p))^* \cdot \sim(b+c) + \sim b$$

$$\equiv \{ \text{Distributivity} + \text{Boolean Algebra} \}$$

$$b \cdot p \cdot (b \cdot c \cdot q + b \sim c \cdot p + c \cdot q + c \sim c \cdot q)^* \cdot \sim b \cdot \sim c + \sim b$$

$$\equiv \{ \text{Commutativity} + \text{Distributivity} + \text{Boolean Algebra} \}$$

$$b \cdot p \cdot (c \cdot q + \sim c \cdot b \cdot p)^* \cdot \sim b \sim c + \sim b$$

$$\equiv \{ \text{Denesting: } (e+f)^* = (e^* \cdot f)^* \cdot e^* \}$$

$$b \cdot p \cdot ((c \cdot q)^* \cdot \sim c \cdot b \cdot p)^* \cdot (c \cdot q)^* \cdot \sim b \sim c + \sim b$$

Proof (axiomatic)

$$b \cdot p \cdot ((b+c) \cdot (c \cdot q + \sim c \cdot p))^* \cdot \sim(b+c) + \sim b$$

$$\equiv \{ \text{Distributivity} + \text{Boolean Algebra} \}$$

$$b \cdot p \cdot (b \cdot c \cdot q + b \sim c \cdot p + c \cdot q + c \sim c \cdot q)^* \cdot \sim b \cdot \sim c + \sim b$$

$$\equiv \{ \text{Commutativity} + \text{Distributivity} + \text{Boolean Algebra} \}$$

$$b \cdot p \cdot (c \cdot q + \sim c \cdot b \cdot p)^* \cdot \sim b \sim c + \sim b$$

$$\equiv \{ \text{Denesting: } (e+f)^* = (e^* \cdot f)^* \cdot e^* \}$$

$$b \cdot p \cdot ((c \cdot q)^* \cdot \sim c \cdot b \cdot p)^* \cdot (c \cdot q)^* \cdot \sim b \sim c + \sim b$$

$$\equiv \{ \text{Sliding: } (e \cdot f)^* \cdot e = e \cdot (f \cdot e)^* \}$$

Proof (axiomatic)

$$b \cdot p \cdot ((b+c) \cdot (c \cdot q + \sim c \cdot p))^* \cdot \sim(b+c) + \sim b$$

$$\equiv \{ \text{Distributivity} + \text{Boolean Algebra} \}$$

$$b \cdot p \cdot (b \cdot c \cdot q + b \sim c \cdot p + c \cdot q + c \sim c \cdot q)^* \cdot \sim b \cdot \sim c + \sim b$$

$$\equiv \{ \text{Commutativity} + \text{Distributivity} + \text{Boolean Algebra} \}$$

$$b \cdot p \cdot (c \cdot q + \sim c \cdot b \cdot p)^* \cdot \sim b \sim c + \sim b$$

$$\equiv \{ \text{Denesting: } (e+f)^* = (e^* \cdot f)^* \cdot e^* \}$$

$$b \cdot p \cdot ((c \cdot q)^* \cdot \sim c \cdot b \cdot p)^* \cdot (c \cdot q)^* \cdot \sim b \sim c + \sim b$$

$$\equiv \{ \text{Sliding: } (e \cdot f)^* \cdot e = e \cdot (f \cdot e)^* \}$$

$$(b \cdot p \cdot (c \cdot q)^* \cdot \sim c)^* \cdot b \cdot p \cdot (c \cdot q)^* \cdot \sim b \cdot \sim c + \sim b$$

Proof (axiomatic)

$$b \cdot p \cdot ((b+c) \cdot (c \cdot q + \sim c \cdot p))^* \cdot \sim(b+c) + \sim b$$

$$\equiv \{ \text{Distributivity} + \text{Boolean Algebra} \}$$

$$b \cdot p \cdot (b \cdot c \cdot q + b \sim c \cdot p + c \cdot q + c \sim c \cdot q)^* \cdot \sim b \cdot \sim c + \sim b$$

$$\equiv \{ \text{Commutativity} + \text{Distributivity} + \text{Boolean Algebra} \}$$

$$b \cdot p \cdot (c \cdot q + \sim c \cdot b \cdot p)^* \cdot \sim b \sim c + \sim b$$

$$\equiv \{ \text{Denesting: } (e+f)^* = (e^* \cdot f)^* \cdot e^* \}$$

$$b \cdot p \cdot ((c \cdot q)^* \cdot \sim c \cdot b \cdot p)^* \cdot (c \cdot q)^* \cdot \sim b \sim c + \sim b$$

$$\equiv \{ \text{Sliding: } (e \cdot f)^* \cdot e = e \cdot (f \cdot e)^* \}$$

$$(b \cdot p \cdot (c \cdot q)^* \cdot \sim c)^* \cdot b \cdot p \cdot (c \cdot q)^* \cdot \sim b \cdot \sim c + \sim b$$

$$\equiv \{ \text{Distributivity} \}$$

Proof (axiomatic)

$$b \cdot p \cdot ((b+c) \cdot (c \cdot q + \sim c \cdot p))^* \cdot \sim(b+c) + \sim b$$

$$\equiv \{ \text{Distributivity} + \text{Boolean Algebra} \}$$

$$b \cdot p \cdot (b \cdot c \cdot q + b \sim c \cdot p + c \cdot q + c \sim c \cdot q)^* \cdot \sim b \cdot \sim c + \sim b$$

$$\equiv \{ \text{Commutativity} + \text{Distributivity} + \text{Boolean Algebra} \}$$

$$b \cdot p \cdot (c \cdot q + \sim c \cdot b \cdot p)^* \cdot \sim b \sim c + \sim b$$

$$\equiv \{ \text{Denesting: } (e+f)^* = (e^* \cdot f)^* \cdot e^* \}$$

$$b \cdot p \cdot ((c \cdot q)^* \cdot \sim c \cdot b \cdot p)^* \cdot (c \cdot q)^* \cdot \sim b \sim c + \sim b$$

$$\equiv \{ \text{Sliding: } (e \cdot f)^* \cdot e = e \cdot (f \cdot e)^* \}$$

$$(b \cdot p \cdot (c \cdot q)^* \cdot \sim c)^* \cdot b \cdot p \cdot (c \cdot q)^* \cdot \sim b \cdot \sim c + \sim b$$

$$\equiv \{ \text{Distributivity} \}$$

$$((b \cdot p \cdot (c \cdot q)^* \cdot \sim c)^* \cdot b \cdot p \cdot (c \cdot q)^* \cdot \sim c + 1) \cdot \sim b$$

Proof (axiomatic)

$$b \cdot p \cdot ((b+c) \cdot (c \cdot q + \sim c \cdot p))^* \cdot \sim(b+c) + \sim b$$

$$\equiv \{ \text{Distributivity} + \text{Boolean Algebra} \}$$

$$b \cdot p \cdot (b \cdot c \cdot q + b \sim c \cdot p + c \cdot q + c \sim c \cdot q)^* \cdot \sim b \cdot \sim c + \sim b$$

$$\equiv \{ \text{Commutativity} + \text{Distributivity} + \text{Boolean Algebra} \}$$

$$b \cdot p \cdot (c \cdot q + \sim c \cdot b \cdot p)^* \cdot \sim b \sim c + \sim b$$

$$\equiv \{ \text{Denesting: } (e+f)^* = (e^* \cdot f)^* \cdot e^* \}$$

$$b \cdot p \cdot ((c \cdot q)^* \cdot \sim c \cdot b \cdot p)^* \cdot (c \cdot q)^* \cdot \sim b \sim c + \sim b$$

$$\equiv \{ \text{Sliding: } (e \cdot f)^* \cdot e = e \cdot (f \cdot e)^* \}$$

$$(b \cdot p \cdot (c \cdot q)^* \cdot \sim c)^* \cdot b \cdot p \cdot (c \cdot q)^* \cdot \sim b \cdot \sim c + \sim b$$

$$\equiv \{ \text{Distributivity} \}$$

$$((b \cdot p \cdot (c \cdot q)^* \cdot \sim c)^* \cdot b \cdot p \cdot (c \cdot q)^* \cdot \sim c + 1) \cdot \sim b$$

$$\equiv \{ \text{Fixpoint} \}$$

Proof (axiomatic)

$$b \cdot p \cdot ((b+c) \cdot (c \cdot q + \sim c \cdot p)) * \cdot \sim(b+c) + \sim b$$

$$\equiv \{ \text{Distributivity} + \text{Boolean Algebra} \}$$

$$b \cdot p \cdot (b \cdot c \cdot q + b \sim c \cdot p + c \cdot q + c \sim c \cdot q) * \cdot \sim b \cdot \sim c + \sim b$$

$$\equiv \{ \text{Commutativity} + \text{Distributivity} + \text{Boolean Algebra} \}$$

$$b \cdot p \cdot (c \cdot q + \sim c \cdot b \cdot p) * \cdot \sim b \sim c + \sim b$$

$$\equiv \{ \text{Denesting: } (e+f)*=(e* \cdot f)* \cdot e* \}$$

$$b \cdot p \cdot ((c \cdot q) * \cdot \sim c \cdot b \cdot p) * \cdot (c \cdot q) * \cdot \sim b \sim c + \sim b$$

$$\equiv \{ \text{Sliding: } (e \cdot f)* \cdot e = e \cdot (f \cdot e)* \}$$

$$(b \cdot p \cdot (c \cdot q) * \cdot \sim c) * \cdot b \cdot p \cdot (c \cdot q) * \cdot \sim b \cdot \sim c + \sim b$$

$$\equiv \{ \text{Distributivity} \}$$

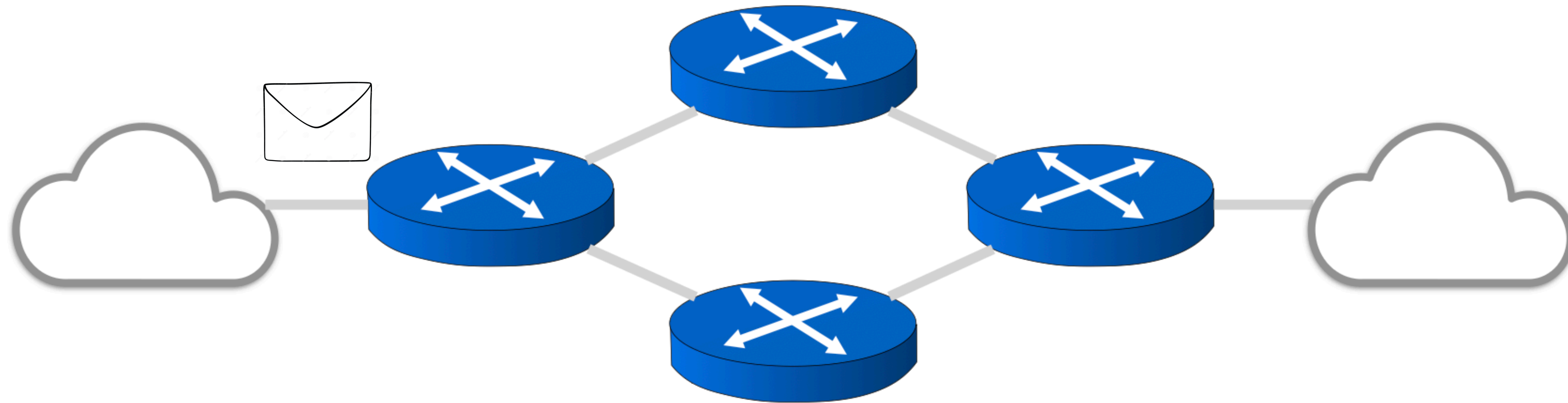
$$((b \cdot p \cdot (c \cdot q) * \cdot \sim c) * \cdot b \cdot p \cdot (c \cdot q) * \cdot \sim c + 1) \cdot \sim b$$

$$\equiv \{ \text{Fixpoint} \}$$

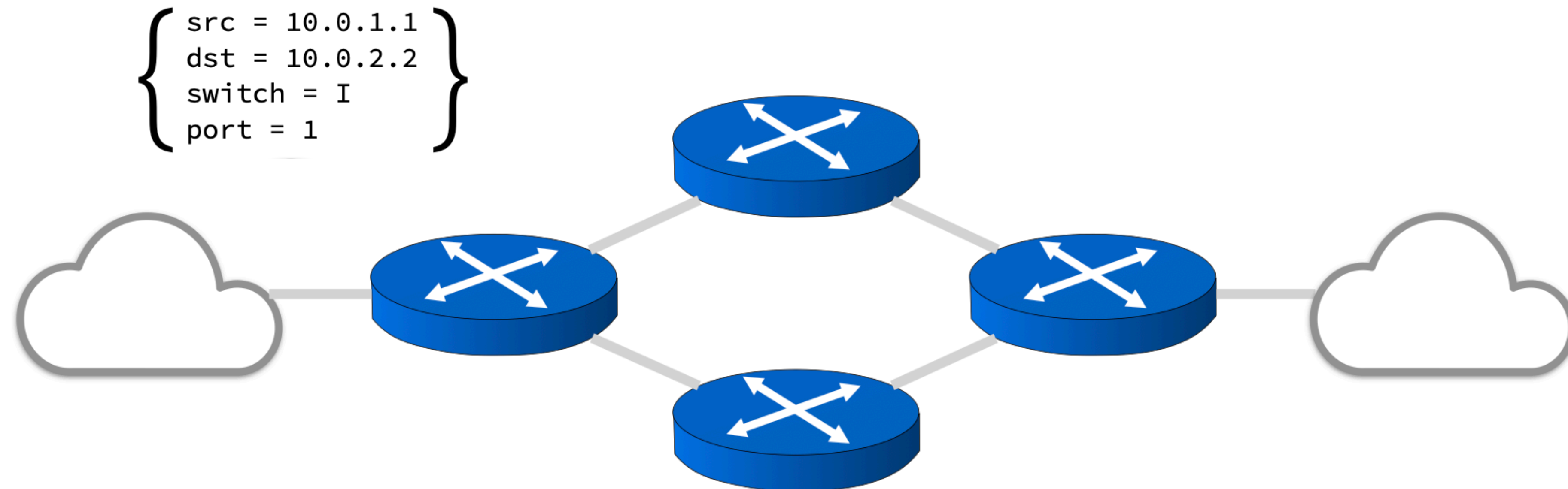
$$(b \cdot p \cdot (c \cdot q) * \cdot \sim c) * \cdot \sim b$$

From KAT to NetKAT

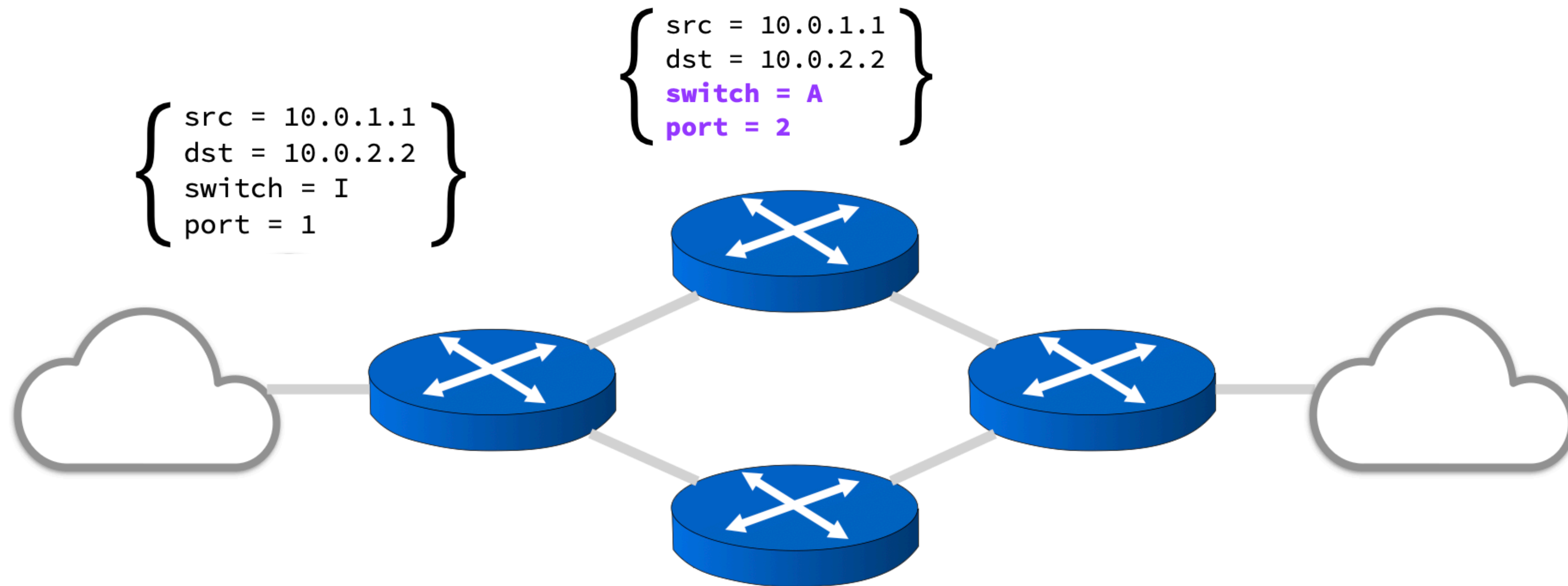
Packets: Records of fixed-width data



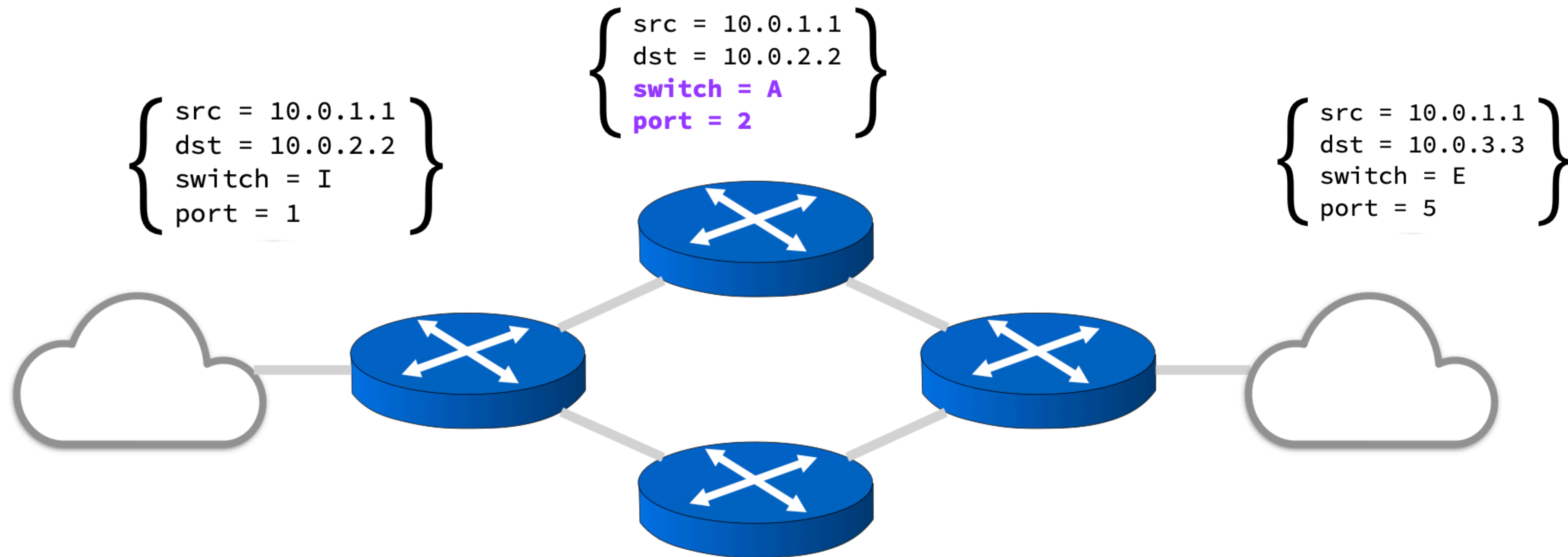
Packets: Records of fixed-width data



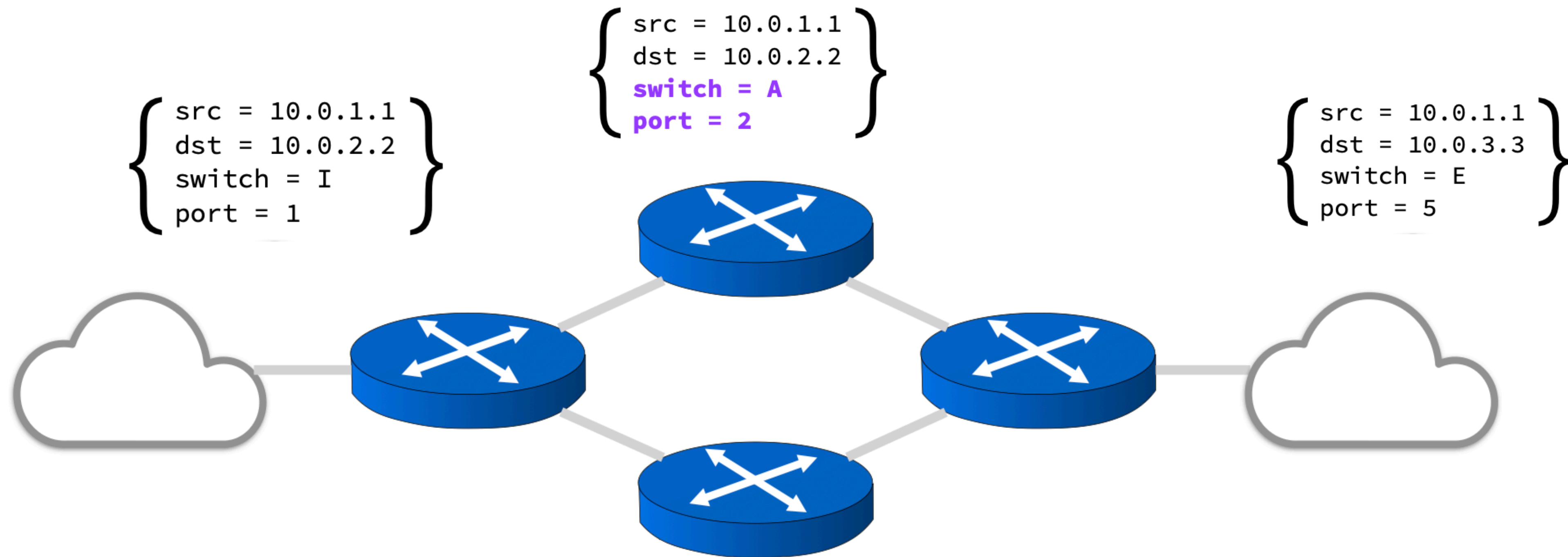
Packets: Records of fixed-width data



Packets: Records of fixed-width data



Packets: Records of fixed-width data



NetKAT = KAT where tests + actions are interpreted over packets!

NetKAT

```
p, q ::=  
| skip  
| drop  
| f = n  
| !p  
| f := n  
| p ; q  
| p + q  
| p*  
| dup
```

NetKAT

`p, q ::=`

`| skip`

`| drop`

`| f = n`

`| !p`

`| f := n`

`| p ; q`

`| p + q`

`| p*`

`| dup`

Packet Classification

To model match-action tables, need predicates on packet headers (and other variables)

Match	Action
<code>tcp_dst=22; ip_dst=10.0.1.1</code>	<code>allow</code>
<code>tcp_dst=22</code>	<code>deny</code>
<code>ip_dst=10.0.2.2</code>	<code>allow</code>
<code>ip_dst=10.0.3.3</code>	<code>allow</code>
<code>*</code>	<code>deny</code>

NetKAT

```
p, q ::=  
| skip  
| drop  
| f = n  
| !p  
| f := n  
| p ; q  
| p + q  
| p*  
| dup
```

NetKAT

```
p, q ::=  
| skip  
| drop  
| f = n  
| !p  
| f := n  
| p ; q  
| p + q  
| p*  
| dup
```

Transformations

To model behavior of switches,
need imperative updates on
packets and a way to delimit
end of processing at a switch

$$A \xRightarrow{\text{def}} B$$

dup; sw = A; sw := B; dup

NetKAT

```
p, q ::=  
| skip  
| drop  
| f = n  
| !p  
| f := n  
| p ; q  
| p + q  
| p*  
| dup
```

NetKAT

```
p, q ::=  
| skip  
| drop  
| f = n  
| !p  
| f := n  
| p ; q  
| p + q  
| p*  
| dup
```

Compositionality

To model richer pipelines, need operators that compose smaller programs both conditionally and in sequence

Iteration

To model the potentially iterated processing performed via the topology, need general loops

```
if p then q else r  $\stackrel{\text{def}}{=}$  (p ; q) + (!p ; r)  
while p do q  $\stackrel{\text{def}}{=}$  (p ; q)* ; !p
```

NetKAT - semantics

```
p, q ::=  
| skip  
| drop  
| f = n  
| !p  
| f := n  
| p ; q  
| p + q  
| p*  
| dup
```

Informal: NetKAT programs denote functions that take an input packet and produce a set of packet traces (i.e., non-empty lists).

Formal: $\llbracket p \rrbracket \in \text{Packet} \rightarrow \mathcal{P}(\text{Packet}^+)$

NetKAT - semantics

```
p, q ::=  
| skip  
| drop  
| f = n  
| !p  
| f := n  
| p ; q  
| p + q  
| p*  
| dup
```

Informal: NetKAT programs denote functions that take an input packet and produce a set of packet traces (i.e., non-empty lists).

Formal: $\llbracket p \rrbracket \in \text{Packet} \rightarrow \mathcal{P}(\text{Packet}^+)$

Q: Why a set?

A: Enables dropping packets + multicast

Q: Why a trace?

A: Captures end-to-end forwarding path

Q: Why a function?

A: Simple model of “mostly stateless” forwarding

NetKAT: Fundamental Results

NetKAT: Fundamental Results

⊢ **Axiomatization**

Can we characterize the set of valid program equalities/refinements?

NetKAT: Fundamental Results

⊢ **Axiomatization**

Can we characterize the set of valid program equalities/refinements?

✓ **Verification**

Can we automatically check whether two expressions are equivalent?

NetKAT: Fundamental Results

⊢ **Axiomatization**

Can we characterize the set of valid program equalities/refinements?

Yes: POPL'14

✓ **Verification**

Can we automatically check whether two expressions are equivalent?

NetKAT: Fundamental Results

⊢ **Axiomatization**

Can we characterize the set of valid program equalities/refinements?

Yes: POPL'14

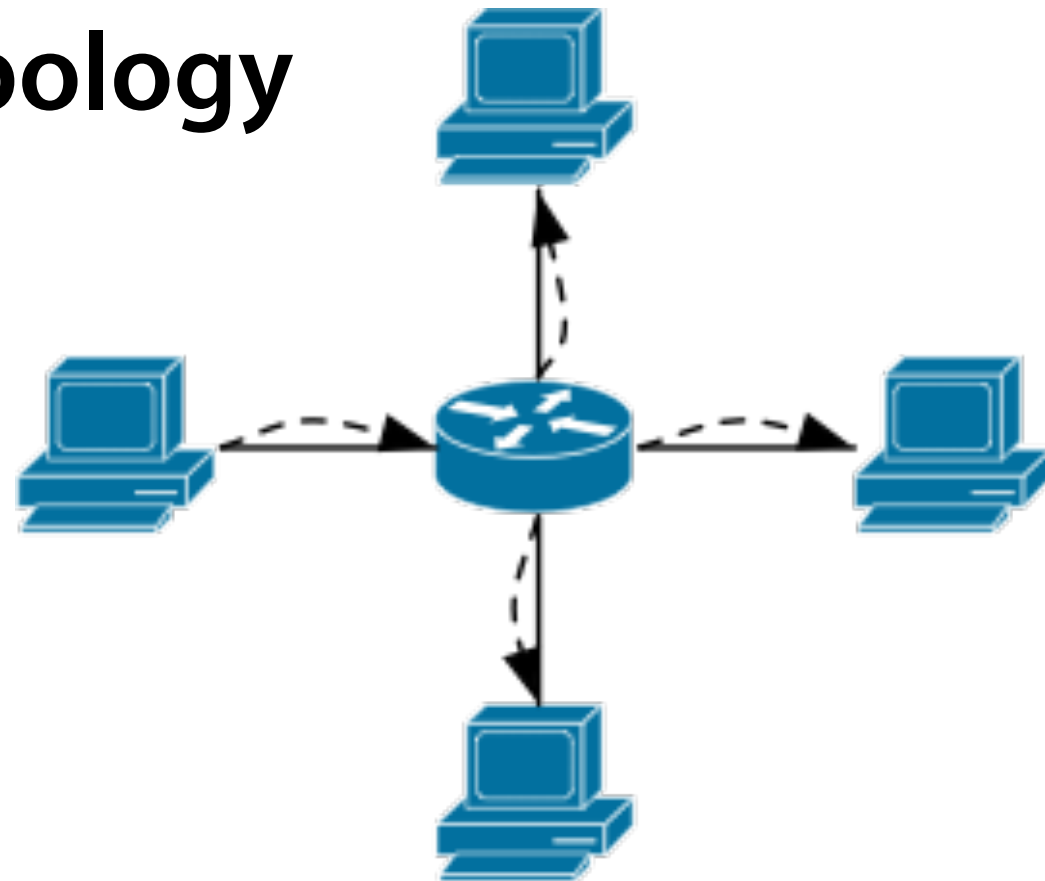
✓ **Verification**

Can we automatically check whether two expressions are equivalent?

Yes: POPL'15, PLDI'24

Example

Topology



Specification

- Forward packets to hosts 1-4
- Monitor traffic to unknown hosts
- Flood broadcast traffic to all hosts
- Disallow SSH traffic from hosts 1-2

Flow Table

```
{pattern={ethSrc=00:00:00:00:00:01,ethTyp=0x800,ipProto=0x06, tcpDstPort=22},action=[]}  
{pattern={ethSrc=00:00:00:00:00:02,ethTyp=0x800,ipProto=0x06, tcpDstPort=22},action=[]}  
{pattern={ethDst=00:00:00:00:00:01},action=[Output(1)]}  
{pattern={ethDst=00:00:00:00:00:02},action=[Output(2)]}  
{pattern={ethDst=00:00:00:00:00:03},action=[Output(3)]}  
{pattern={ethDst=00:00:00:00:00:04},action=[Output(4)]}  
{pattern={ethDst=ff:ff:ff:ff:ff:ff,port=1},action=[Output(4), Output(3), Output(2)]}  
{pattern={ethDst=ff:ff:ff:ff:ff:ff,port=2},action=[Output(4), Output(3), Output(1)]}  
{pattern={ethDst=ff:ff:ff:ff:ff:ff,port=3},action=[Output(4), Output(2), Output(1)]}  
{pattern={ethDst=ff:ff:ff:ff:ff:ff,port=4},action=[Output(3), Output(2), Output(1)]}  
{pattern={ethDst=ff:ff:ff:ff:ff:ff},action=[]}  
{pattern={},action=[Controller]}
```

Example: Forward

```
let forward =  
  if ethDst = 00:00:00:00:00:01 then  
    port := 1  
  else if ethDst = 00:00:00:00:00:02 then  
    port := 2  
  else if ethDst = 00:00:00:00:00:03 then  
    port := 3  
  else if ethDst = 00:00:00:00:00:04 then  
    port := 4  
  else  
    false
```

Example: Broadcast

```
let flood =  
  if port = 1 then  
    port := 2 + port := 3 + port := 4  
  else if port = 2 then  
    port := 1 + port := 3 + port := 4  
  else if port = 3 then  
    port := 1 + port := 2 + port := 4  
  else if port = 4 then  
    port := 1 + port := 2 + port := 3  
  else  
    false  
let broadcast =  
  if ethDst = ff:ff:ff:ff:ff:ff then  
    flood  
  else  
    false
```

Example: Firewall

```
let firewall =  
  if (ethSrc = 00:00:00:00:00:00:01 +  
      ethSrc = 00:00:00:00:00:00:02) ;  
    ethTyp = 0x800 ;  
    ipProto = 0x06 ;  
    tcpDstPort = 22 then  
    false  
else  
    true
```

Example: main program

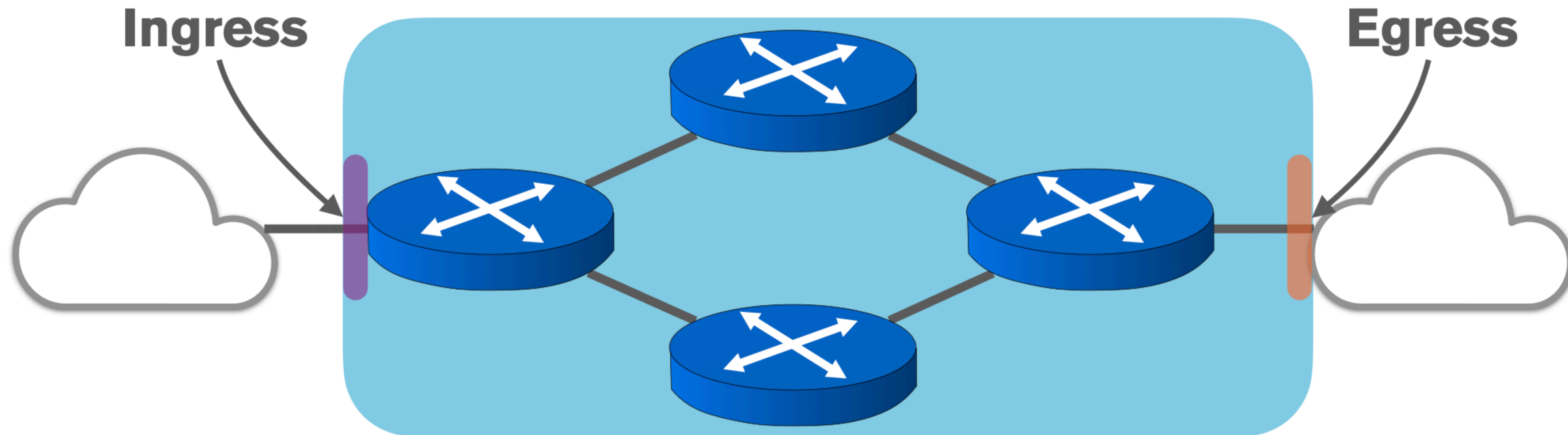
```
let main = (route + monitor); firewall
```



compiles to...

```
{pattern={ethSrc=00:00:00:00:00:01,ethTyp=0x800,ipProto=0x06, tcpDstPort=22},action=[]}  
{pattern={ethSrc=00:00:00:00:00:02,ethTyp=0x800,ipProto=0x06, tcpDstPort=22},action=[]}  
{pattern={ethDst=00:00:00:00:00:01},action=[Output(1)]}  
{pattern={ethDst=00:00:00:00:00:02},action=[Output(2)]}  
{pattern={ethDst=00:00:00:00:00:03},action=[Output(3)]}  
{pattern={ethDst=00:00:00:00:00:04},action=[Output(4)]}  
{pattern={ethDst=ff:ff:ff:ff:ff:ff,port=1},action=[Output(4), Output(3), Output(2)]}  
{pattern={ethDst=ff:ff:ff:ff:ff:ff,port=2},action=[Output(4), Output(3), Output(1)]}  
{pattern={ethDst=ff:ff:ff:ff:ff:ff,port=3},action=[Output(4), Output(2), Output(1)]}  
{pattern={ethDst=ff:ff:ff:ff:ff:ff,port=4},action=[Output(3), Output(2), Output(1)]}  
{pattern={ethDst=ff:ff:ff:ff:ff:ff},action=[]}  
{pattern={},action=[Controller]}
```

Modeling networks in NetKAT



switch

Models the processing
done at each switch

topology

Models the forwarding
done by each link

ingress & egress

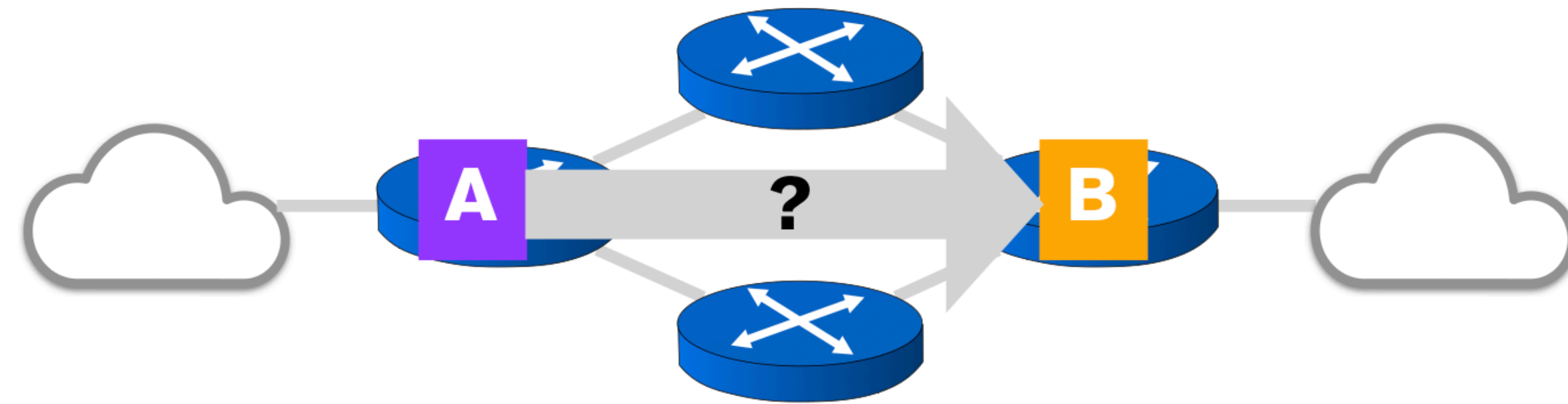
Models the perimeter
of the network

```
ingress ; (switch ; topology)* ; egress
```

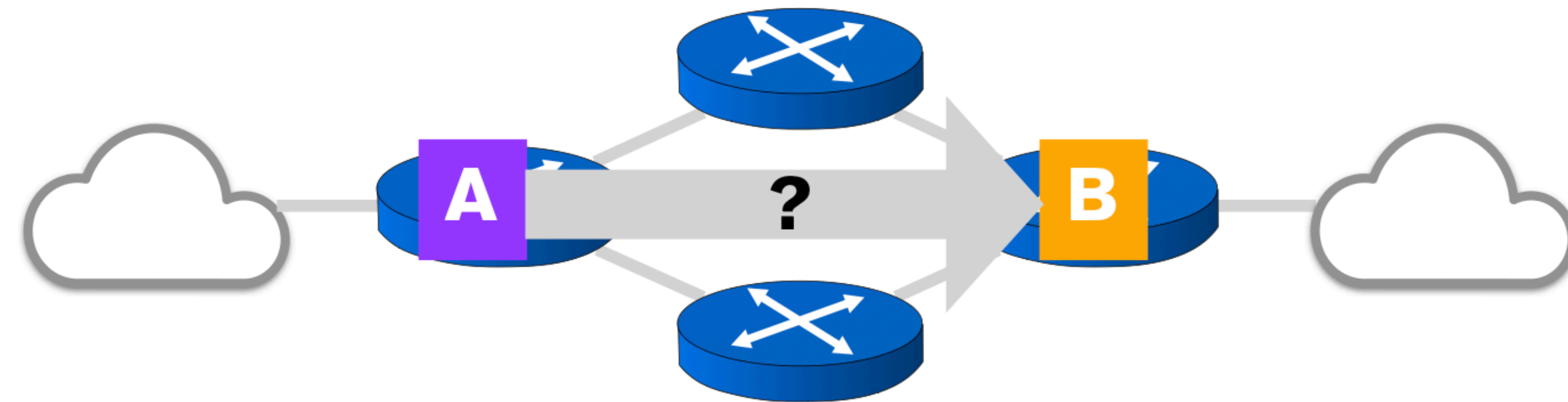
Verification in NetKAT

Idea: encode the program and its specification in a unified framework, then check **equivalence** (or inclusion, etc.)

Example: reachability

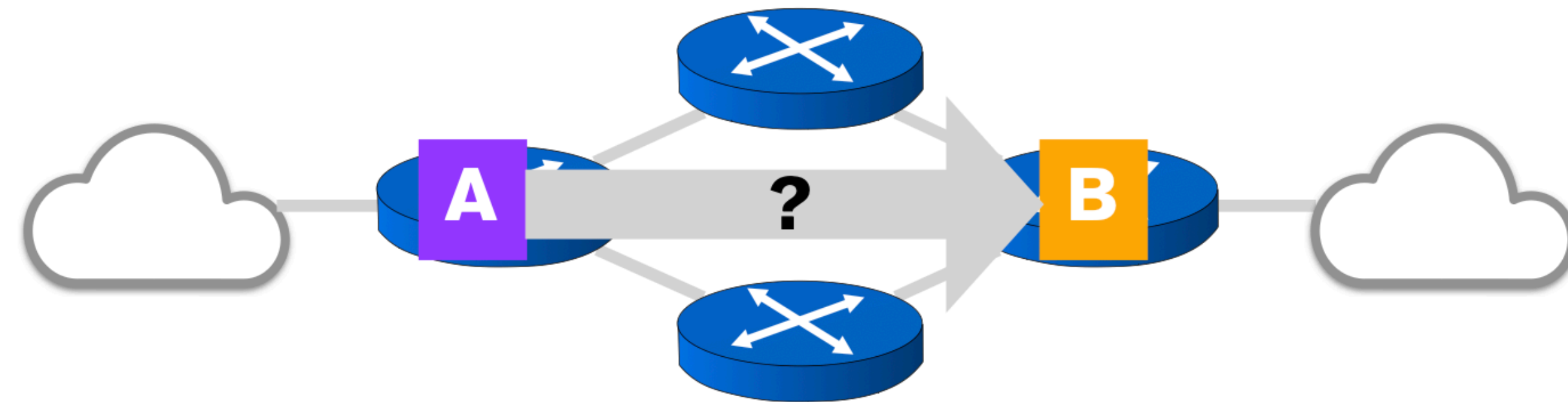


Example: reachability



Property: B reachable from A

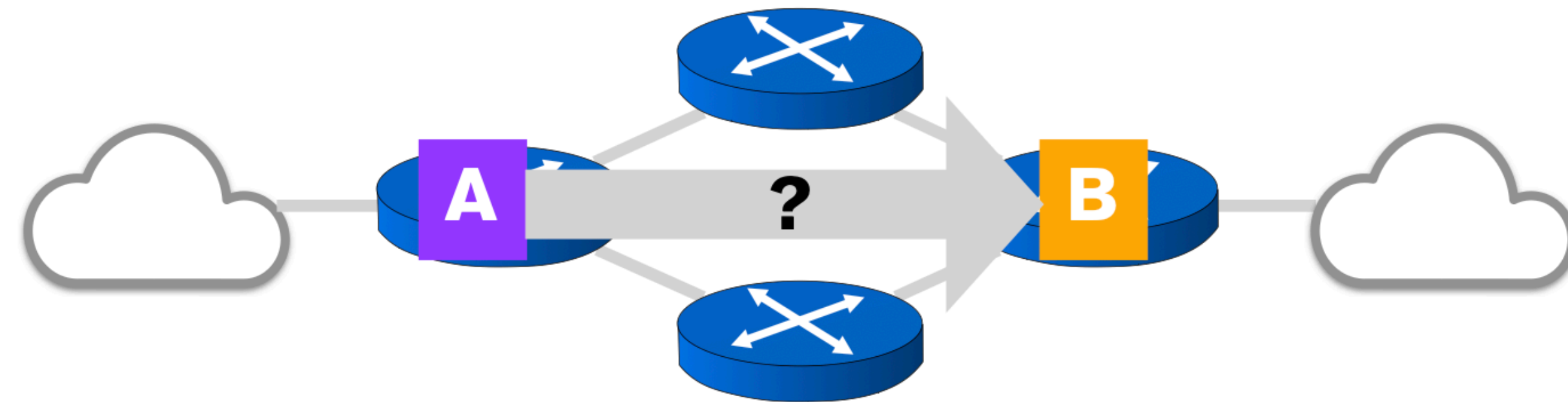
Example: reachability



Property: B reachable from A

Approach:

Example: reachability

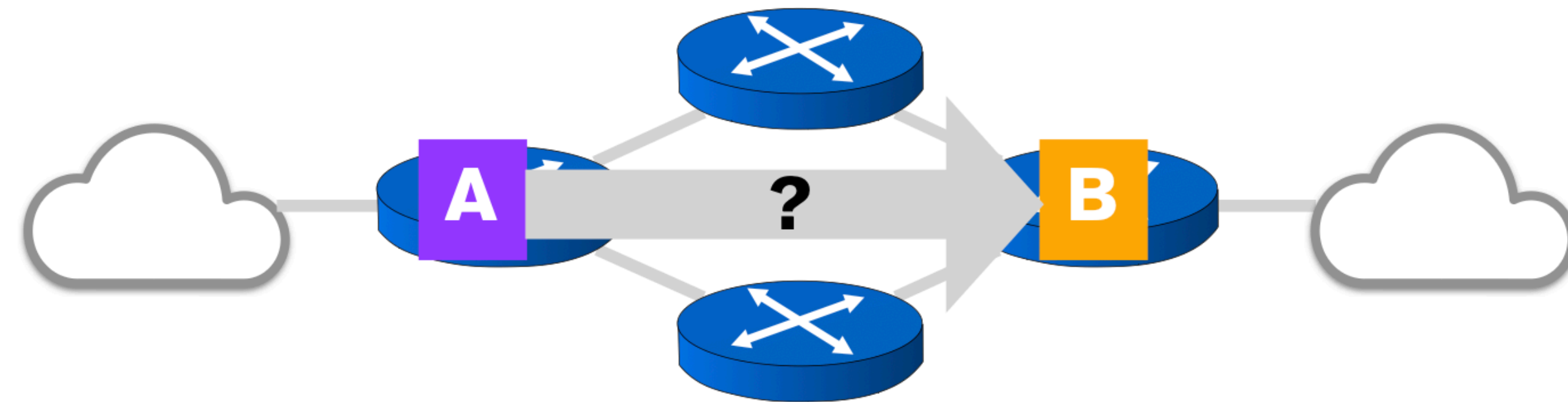


Property: B reachable from A

Approach:

- Build a model with A as ingress and B as egress

Example: reachability

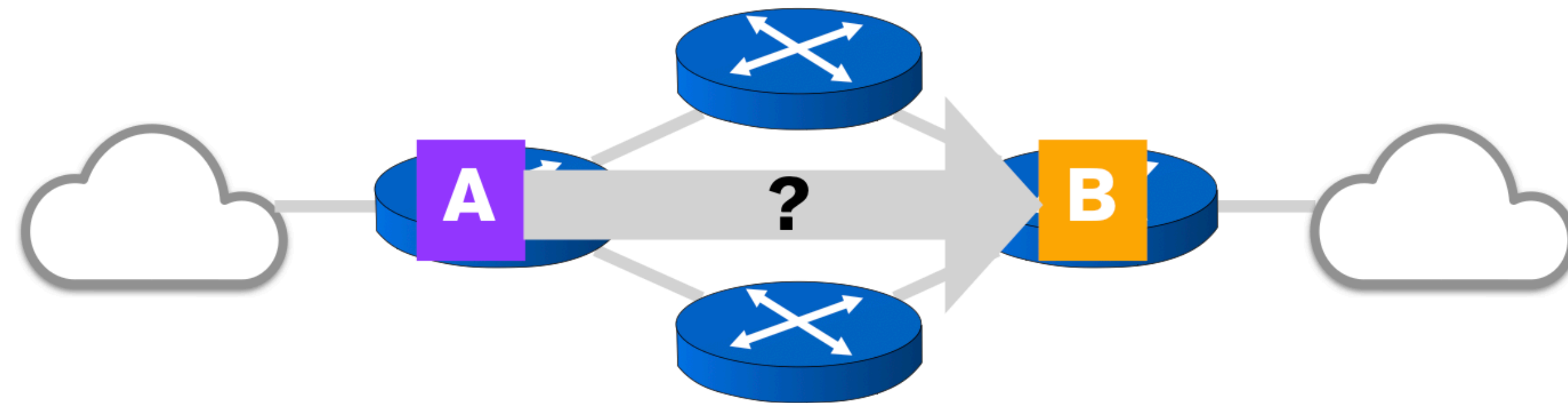


Property: B reachable from A

Approach:

- Build a model with A as ingress and B as egress
- Check for non-emptiness

Example: reachability



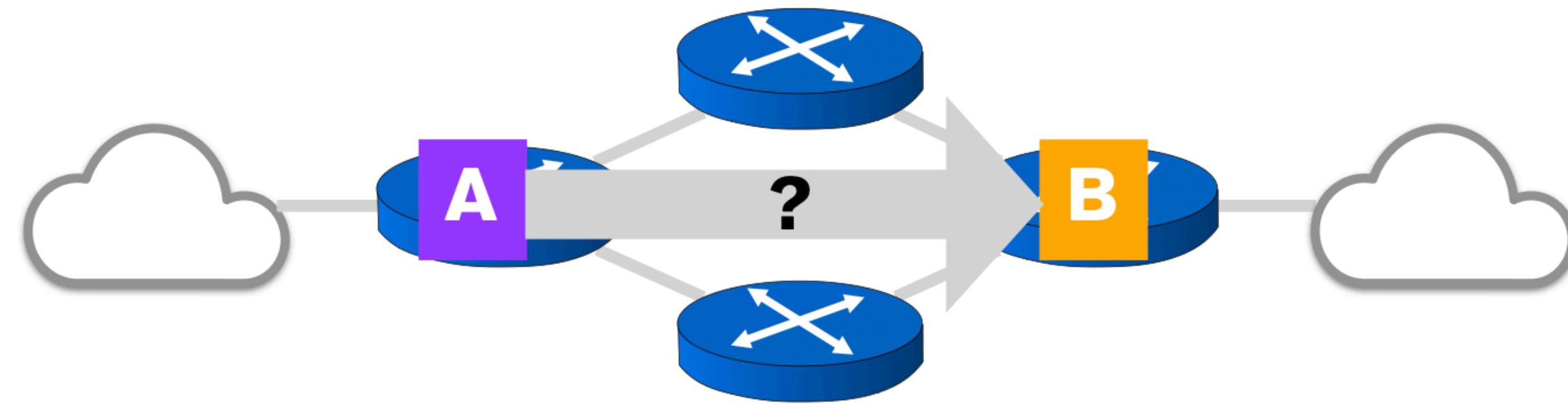
Property: B reachable from A

Approach:

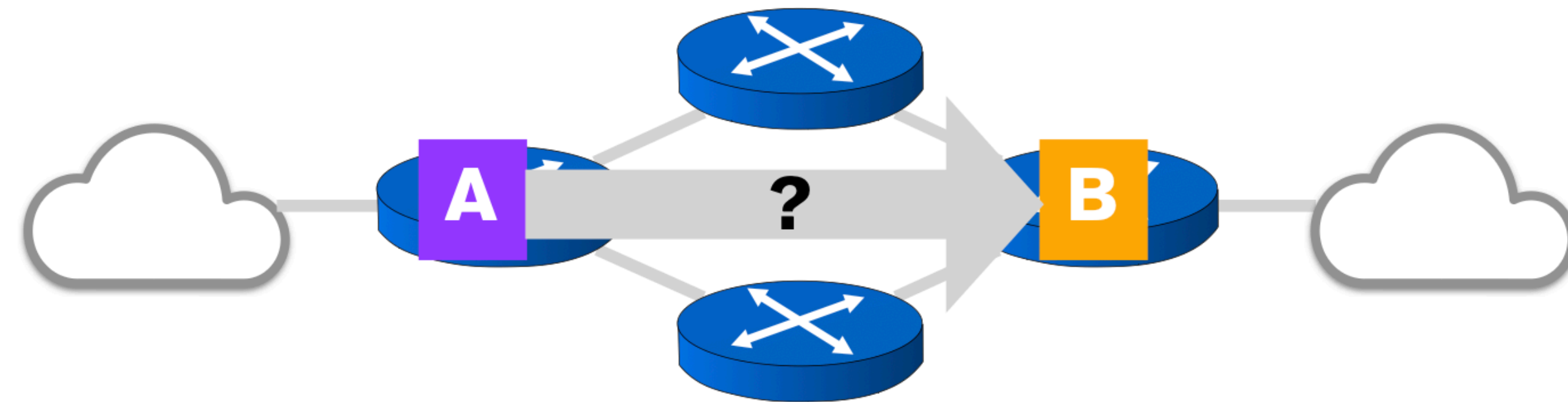
- Build a model with A as ingress and B as egress
- Check for non-emptiness

Query: `switch=A; (switch; topo)*; switch=B ≠ drop`

Example: connectivity



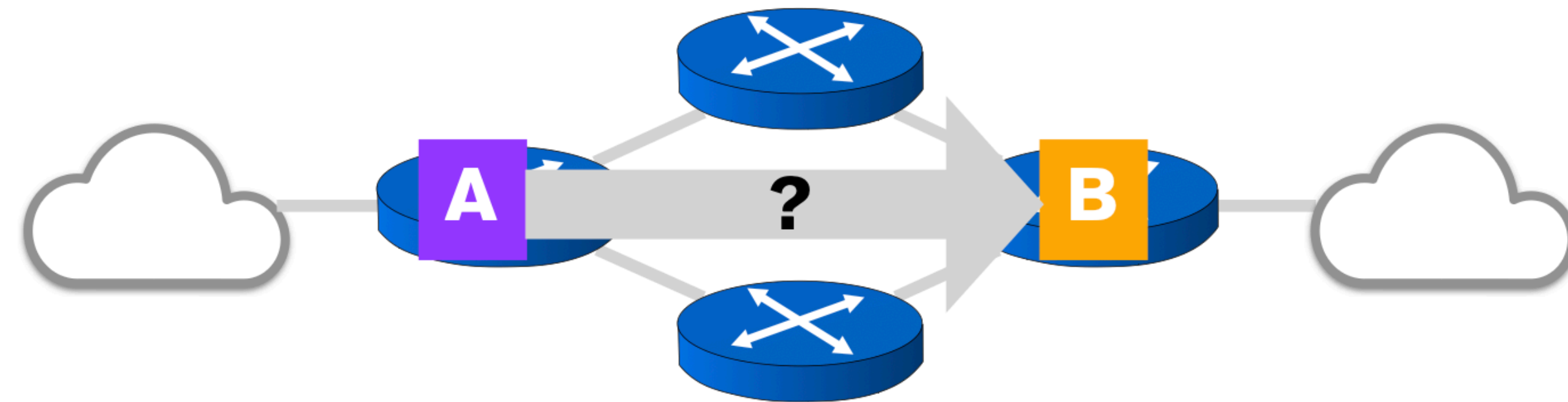
Example: connectivity



Property: *all traffic* from A should reach B

Approach:

Example: connectivity

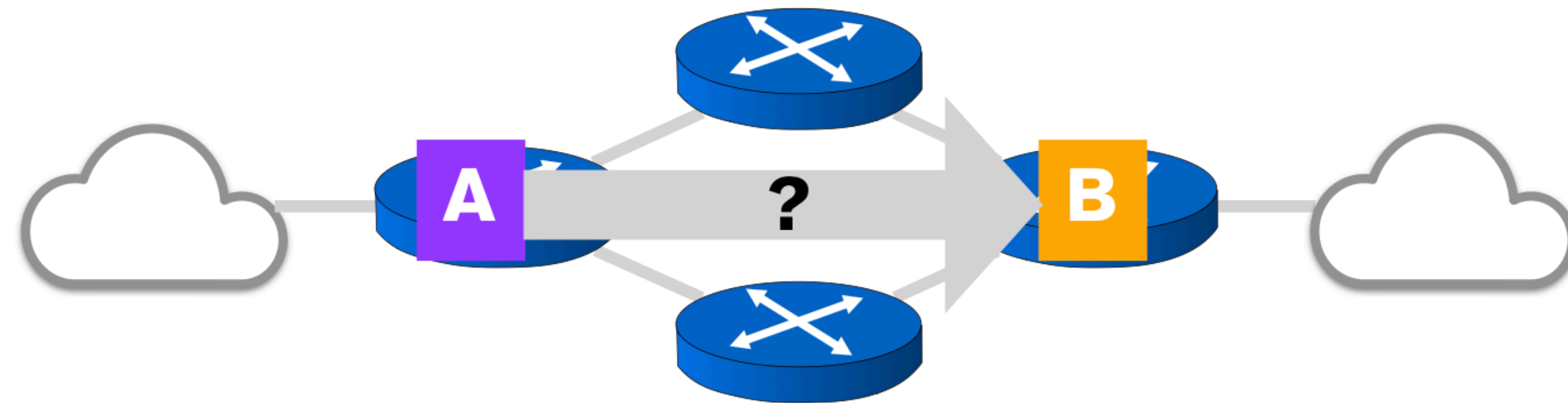


Property: *all traffic* from A should reach B

Approach:

- Build a model with A and B as ingress and egress predicates

Example: connectivity

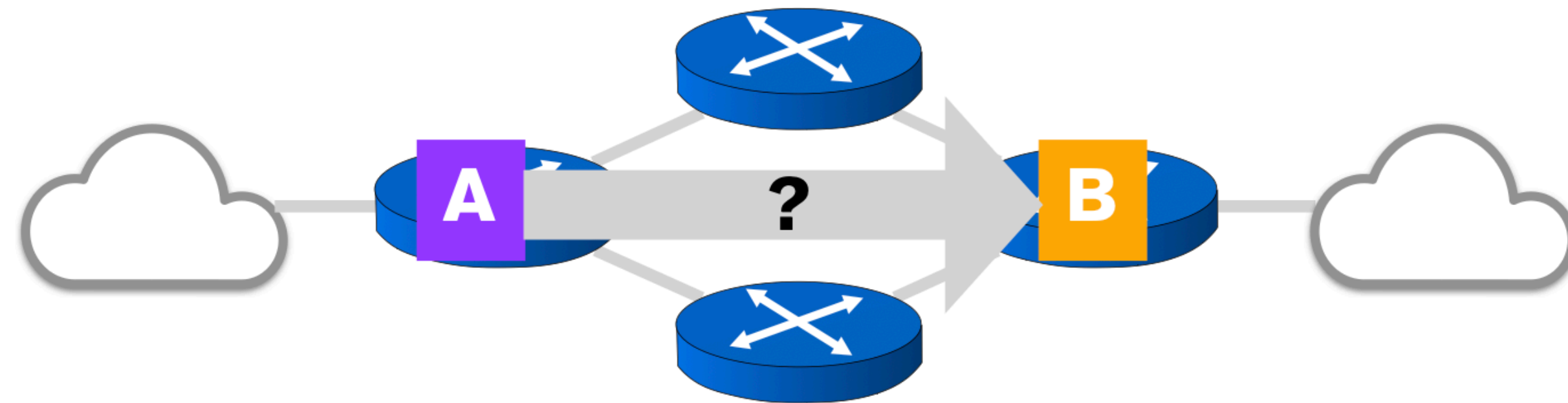


Property: *all traffic* from A should reach B

Approach:

- Build a model with A and B as ingress and egress predicates
- Check for equivalence with program that “teleports” packets from A to B

Example: connectivity



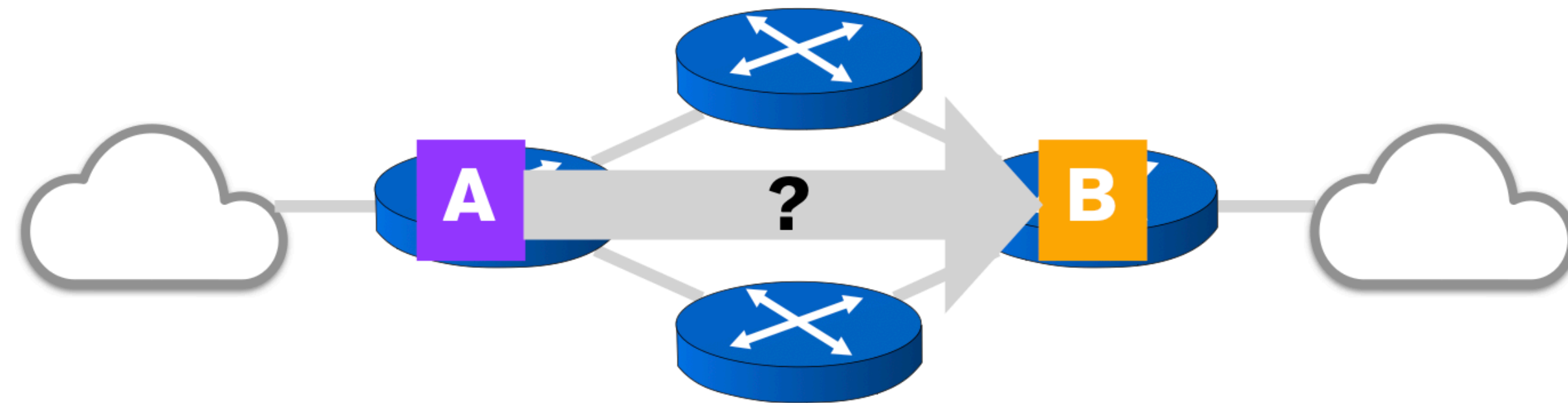
Property: *all traffic* from A should reach B

Approach:

- Build a model with A and B as ingress and egress predicates
- Check for equivalence with program that “teleports” packets from A to B

Query: `switch=A; (switch; topo)*; switch=B` $\equiv$

Example: connectivity



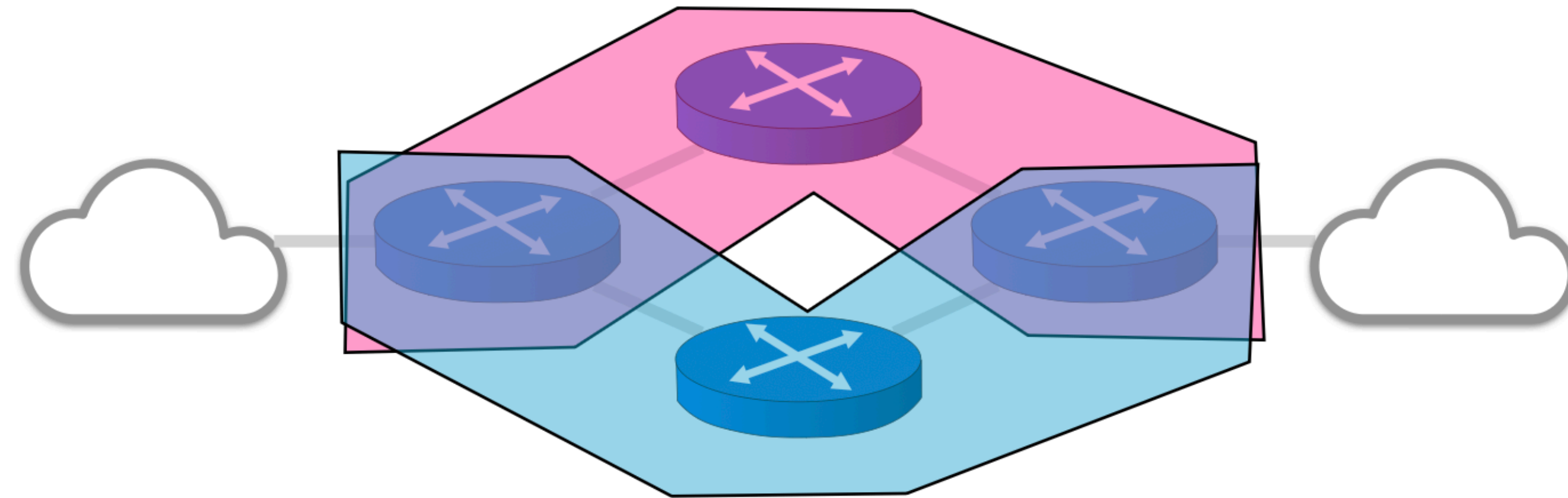
Property: *all traffic* from A should reach B

Approach:

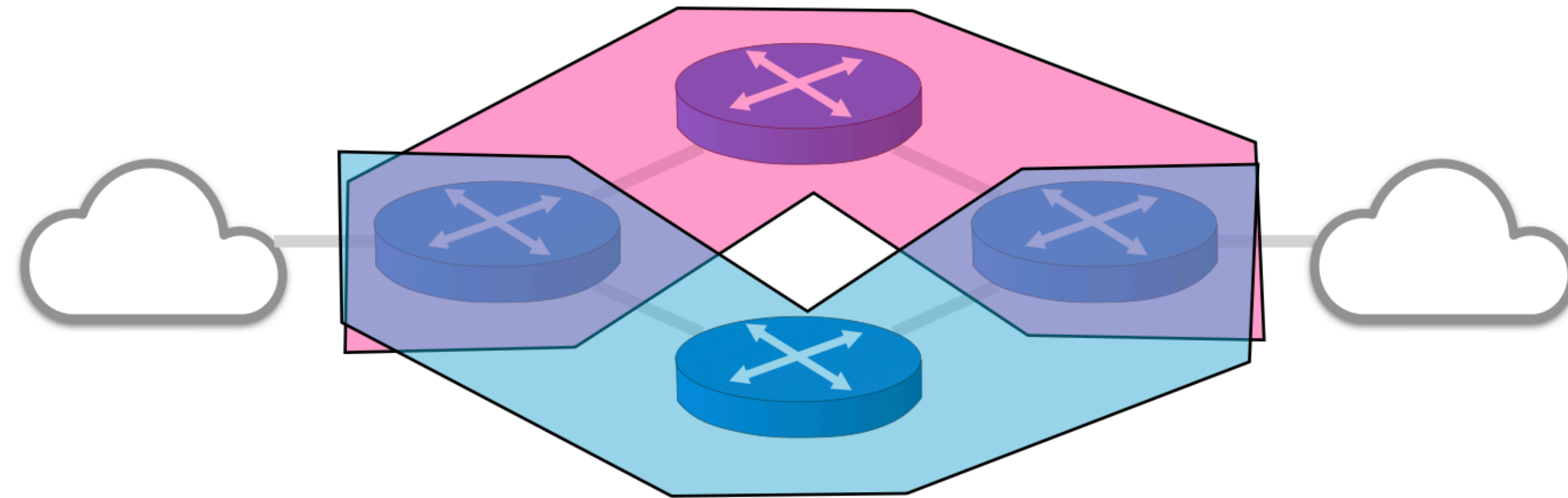
- Build a model with A and B as ingress and egress predicates
- Check for equivalence with program that “teleports” packets from A to B

Query: $\text{switch}=A; (\text{switch}; \text{topo})^*; \text{switch}=B \equiv \text{switch}=A ; \text{switch}:=B$

Example: isolation

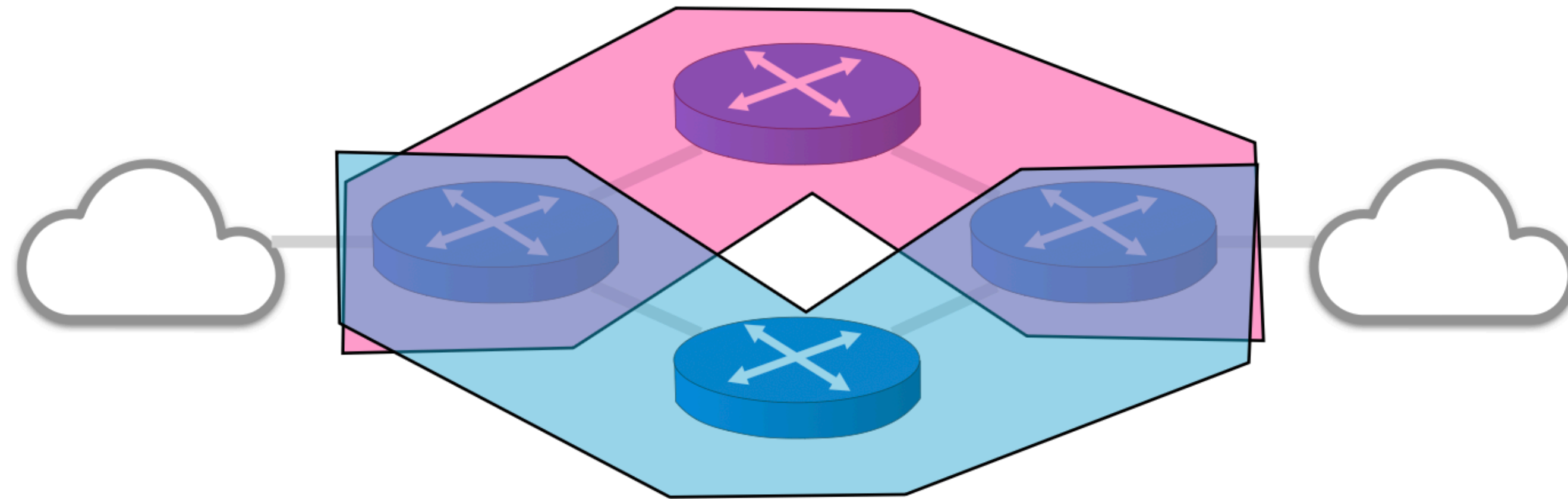


Example: isolation



Property: slice **s** logically isolated from **q**

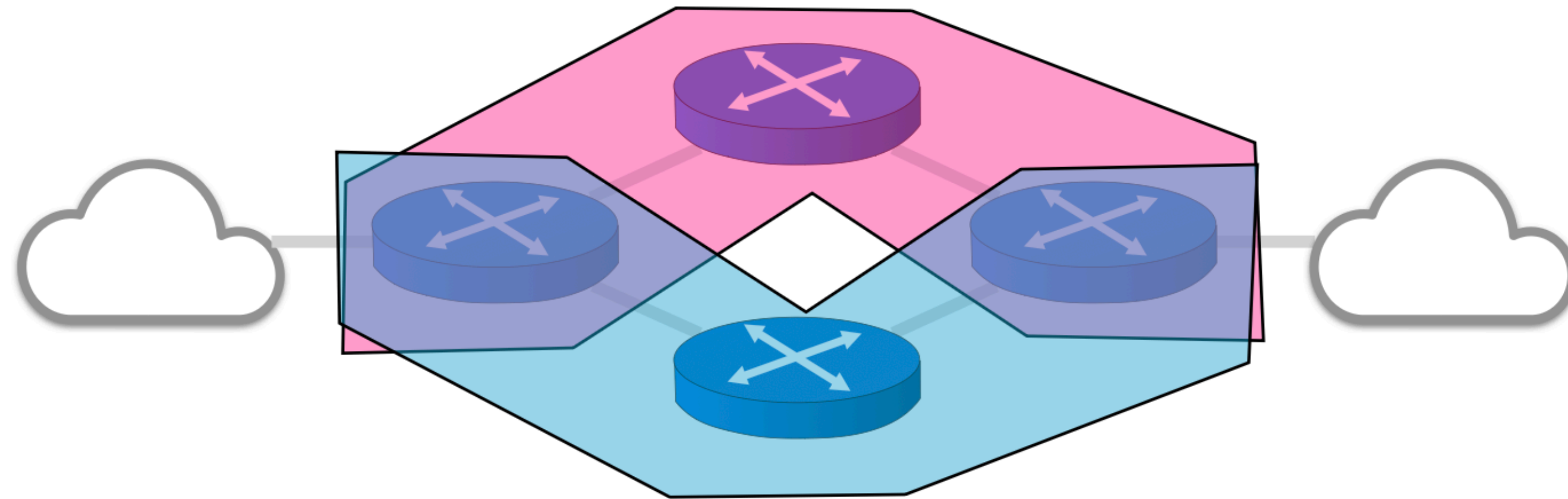
Example: isolation



Property: slice **s** logically isolated from **q**

Approach: Check that running **s** and **q** together is equivalent to running

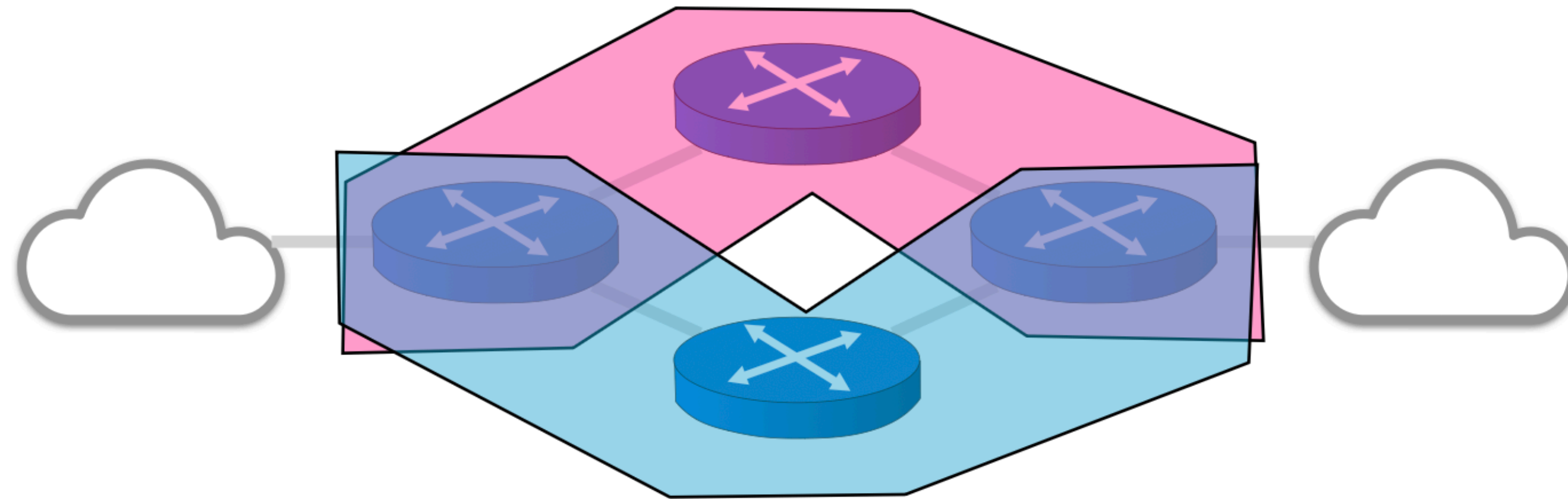
Example: isolation



Property: slice *s* logically isolated from *q*

Approach: Check that running *s* and *q* together is equivalent to running them independently on separate “copies” of the network

Example: isolation

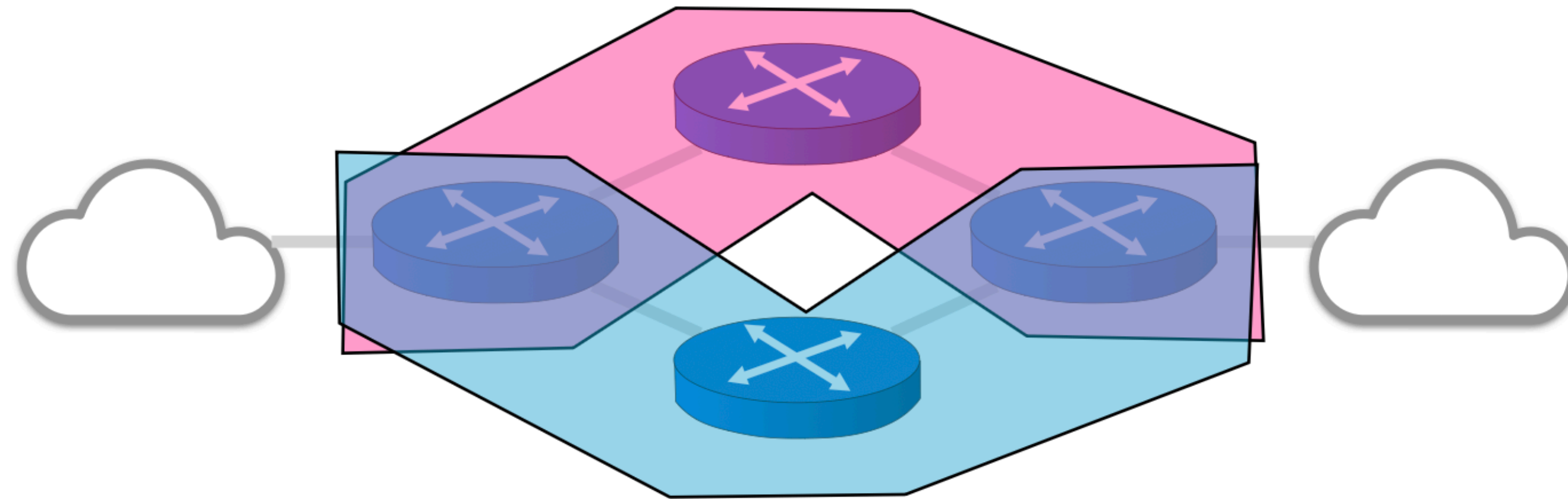


Property: slice s logically isolated from q

Approach: Check that running s and q together is equivalent to running them independently on separate “copies” of the network

Query: $\text{in}; ([s + q] ; \text{topo})^*; \text{eg} \equiv$

Example: isolation



Property: slice s logically isolated from q

Approach: Check that running s and q together is equivalent to running them independently on separate “copies” of the network

Query: $\text{in}; ([s + q] ; \text{topo})^*; \text{eg} \equiv$
 $[\text{in}; (s ; \text{topo})^*; \text{eg}] + [\text{in}; (q ; \text{topo})^*; \text{eg}]$

Axiomatization

All KA + BA + Packet Axioms:

Axiomatization

All KA + BA + Packet Axioms:

$f := n; g := m \equiv g := m; f := n$

Axiomatization

All KA + BA + Packet Axioms:

$f := n; g := m \equiv g := m; f := n$
 $f := n; g = m \equiv g = m; f := n$

Axiomatization

All KA + BA + Packet Axioms:

$$\begin{aligned} f &:= n; g := m \equiv g := m; f := n \\ f &:= n; g = m \equiv g = m; f := n \\ f &:= n; f = n \equiv f := n \end{aligned}$$

Axiomatization

All KA + BA + Packet Axioms:

$$\begin{aligned} f &:= n; g := m \equiv g := m; f := n \\ f &:= n; g = m \equiv g = m; f := n \\ f &:= n; f = n \equiv f := n \\ f &= n; f := n \equiv f = n \end{aligned}$$

Axiomatization

All KA + BA + Packet Axioms:

$$\begin{aligned} f &:= n; g := m \equiv g := m; f := n \\ f &:= n; g = m \equiv g = m; f := n \\ f &:= n; f = n \equiv f := n \\ f &= n; f := n \equiv f = n \\ f &:= n; f := m \equiv f := m \end{aligned}$$

Axiomatization

All KA + BA + Packet Axioms:

```
f := n; g := m ≡ g := m; f := n
f := n; g = m ≡ g = m; f := n
f := n; f = n ≡ f := n
f = n; f := n ≡ f = n
f := n; f := m ≡ f := m
f = n; f = m ≡ false
```

Axiomatization

All KA + BA + Packet Axioms:

```
f := n; g := m ≡ g := m; f := n
f := n; g = m ≡ g = m; f := n
f := n; f = n ≡ f := n
f = n; f := n ≡ f = n
f := n; f := m ≡ f := m
f = n; f = m ≡ false
dup; f = n ≡ f = n; dup
```

Axiomatization

All KA + BA + Packet Axioms:

```
f := n; g := m ≡ g := m; f := n
f := n; g = m ≡ g = m; f := n
f := n; f = n ≡ f := n
f = n; f := n ≡ f = n
f := n; f := m ≡ f := m
f = n; f = m ≡ false
dup; f = n ≡ f = n; dup
 $\Sigma_i f = n_i \equiv \text{true}$ 
```

Deciding equivalence

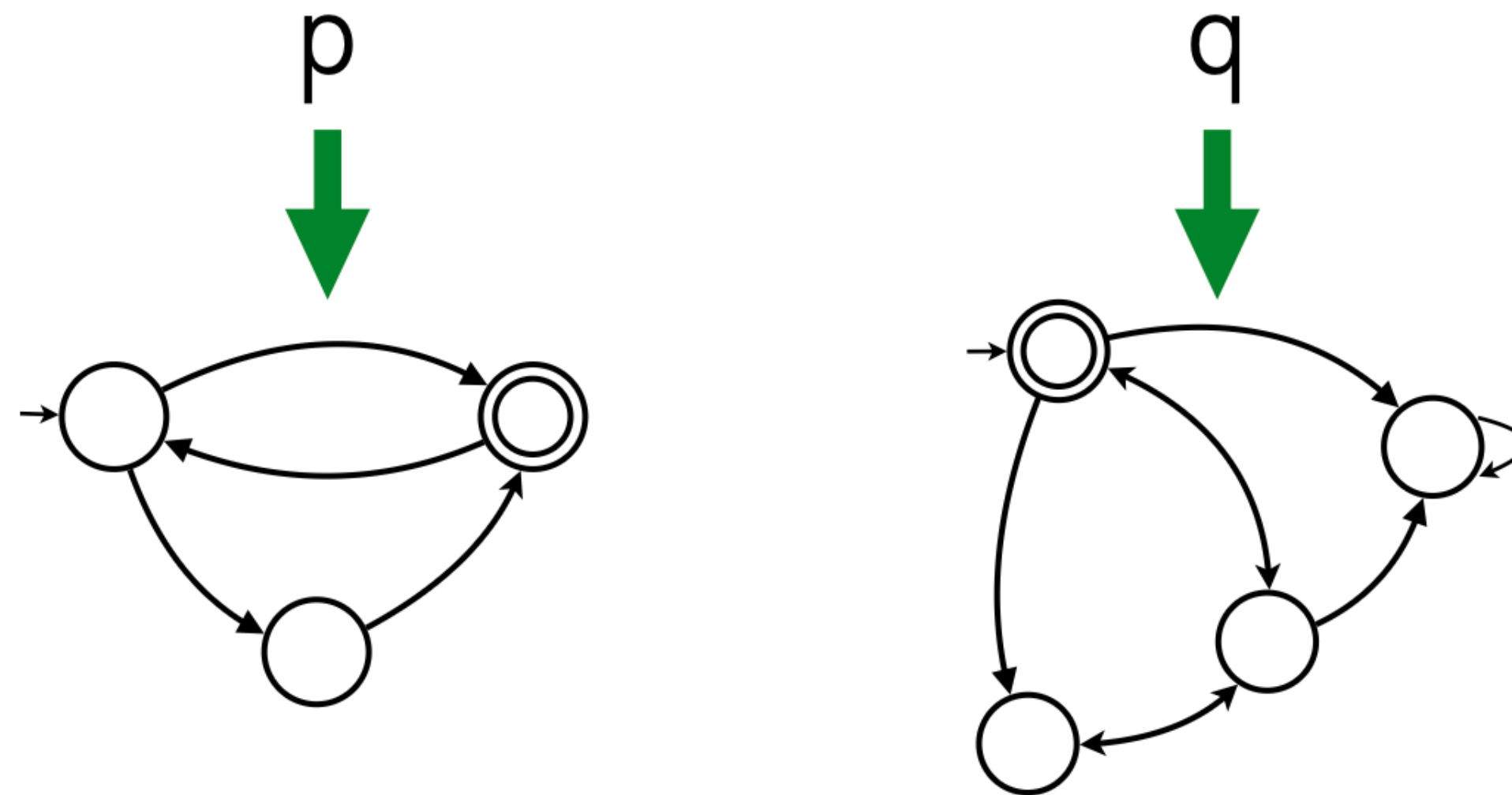
Can check whether programs are equivalent automatically!

Theoretical Insight: NetKAT programs $\leftrightarrow$ NetKAT automata

Deciding equivalence

Can check whether programs are equivalent automatically!

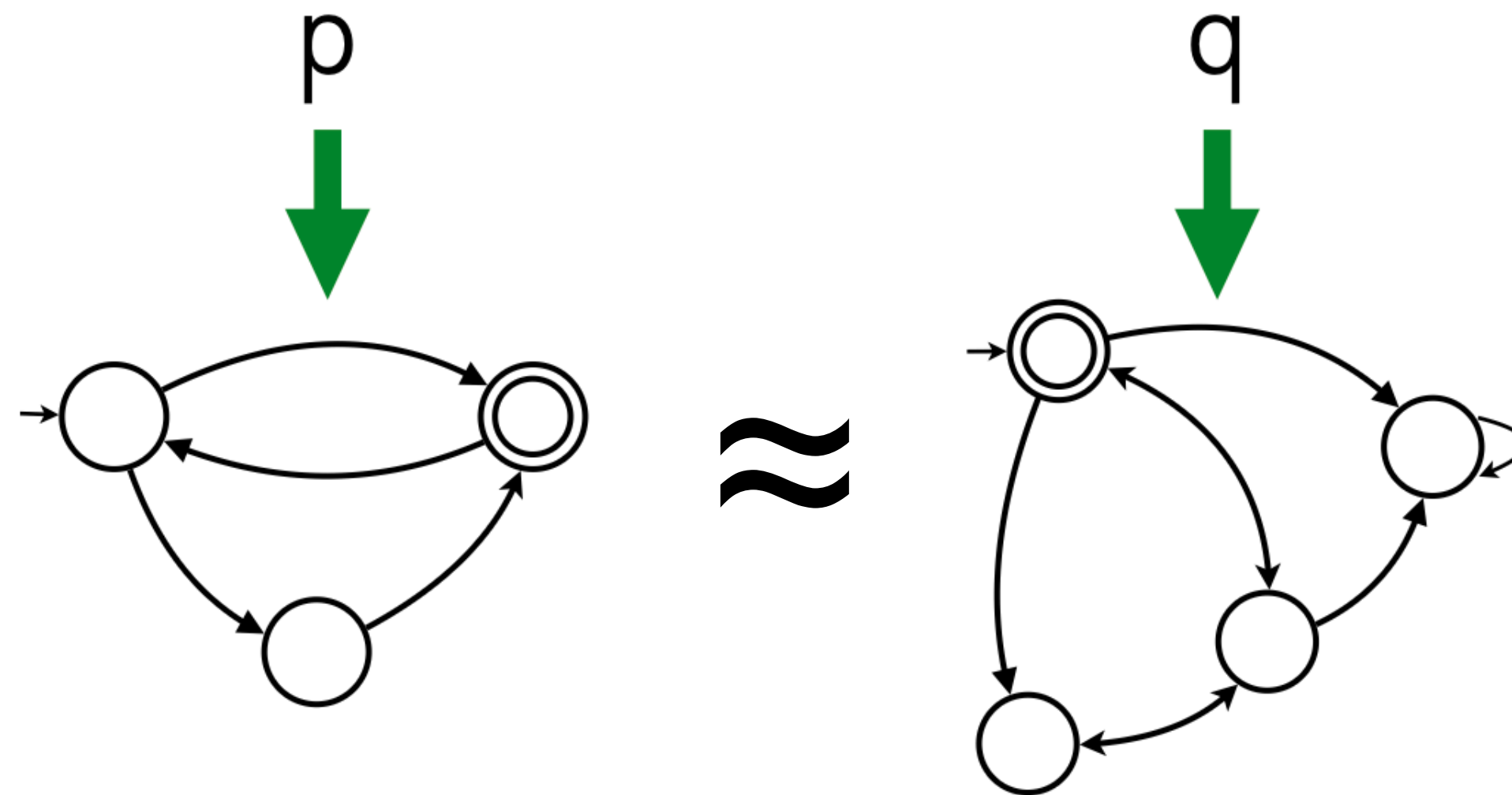
Theoretical Insight: NetKAT programs $\leftrightarrow$ NetKAT automata



Deciding equivalence

Can check whether programs are equivalent automatically!

Theoretical Insight: NetKAT programs $\leftrightarrow$ NetKAT automata



Algorithm checks whether automata are bisimilar

NetKAT automata

A NetKAT Automaton is a tuple $M = \langle S, s_0, \varepsilon, \delta \rangle$ where:

- S is a finite set of states,
- $s_0 \in S$ is the start state,
- $\varepsilon \in S \rightarrow \text{Packet} \rightarrow \text{Packet Set}$
- $\delta \in S \rightarrow \text{Packet} \rightarrow (S * \text{Packet}) \text{ Set}$

NetKAT automata

A NetKAT Automaton is a tuple $M = \langle S, s_0, \varepsilon, \delta \rangle$ where:

- S is a finite set of states,
- $s_0 \in S$ is the start state,
- $\varepsilon \in S \rightarrow \text{Packet} \rightarrow \text{Packet Set}$
- $\delta \in S \rightarrow \text{Packet} \rightarrow (S * \text{Packet}) \text{ Set}$

M accepts a trace in state s if:

- $\text{accept } s [p, p'] \iff p' \in \varepsilon s p$
- $\text{accept } s [p, p'] @ \text{rest} \iff \exists s'. (p', s') \in \delta s p \wedge \text{accept } s' (p' @ \text{rest})$

Evaluation

Dataset

Internet Topology Zoo, a dataset of 140 real-world topologies, mostly large ISPs

Configurations

Synthetic programs that forward traffic along shortest paths

Property

All-pairs reachability

Key question

Performance relative to APKeep, a state-of-the-art network verification tool



Topology

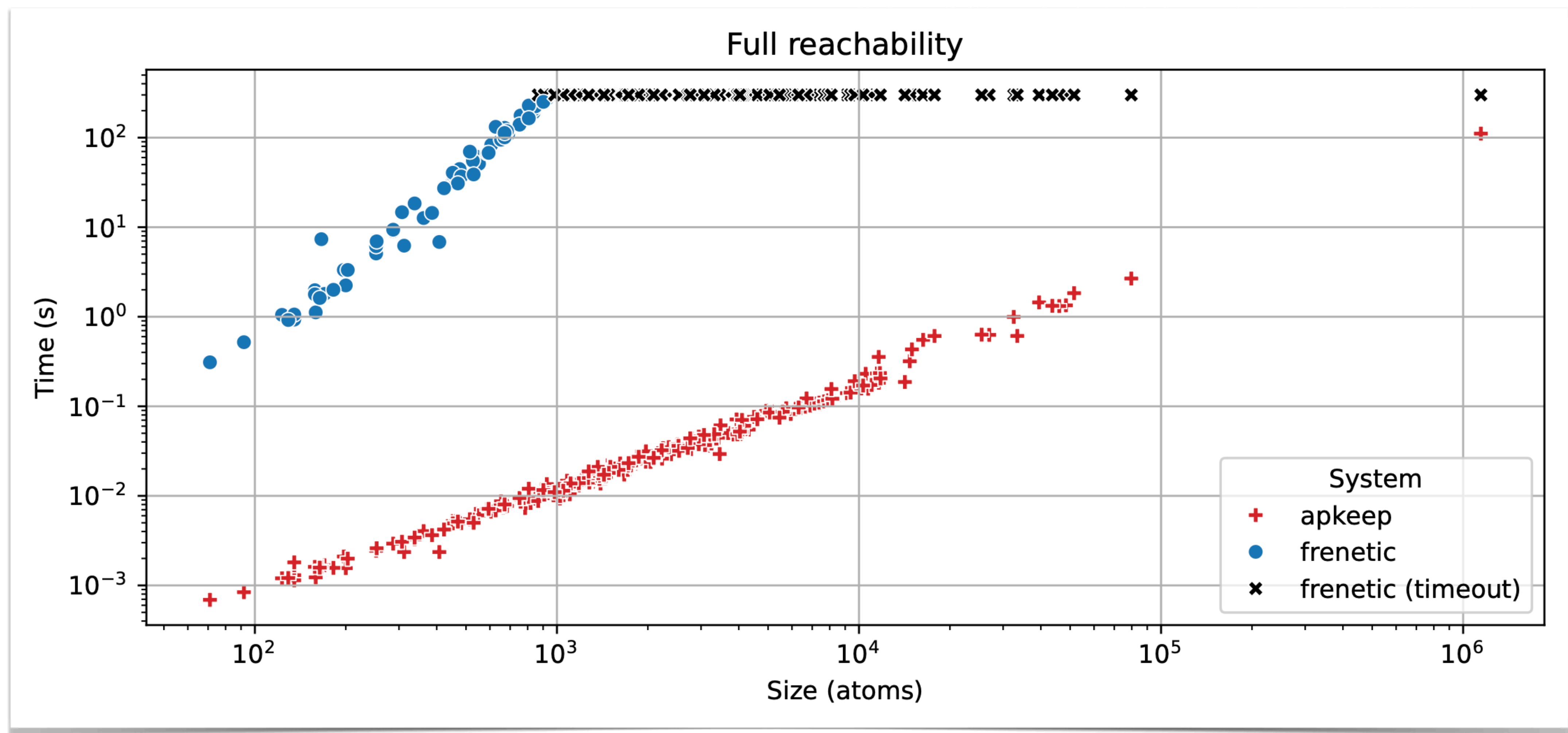
$N_0 = 0 \ N_1 = 1 \ N_2 = 2 \ N_3 = 3 \ N_4 = 4 \ N_5 = 5$

```
top = @pt=-1?·εU(@sw=N5?·(@pt=2?·(@sw←N4·@pt←1)U@pt=1?
·(@sw←N3·@pt←2)U@pt=0?·(@sw←N1·@pt←3))U@sw=N4?·(@pt=1?
·(@sw←N5·@pt←2)U@pt=0?·(@sw←N3·@pt←1))U@sw=N3?·(@pt=2?
·(@sw←N5·@pt←1)U@pt=1?·(@sw←N4·@pt←0)U@pt=0?
·(@sw←N1·@pt←2))U@sw=N2?·(@pt=0?·(@sw←N1·@pt←1))U@sw=N1?·(@pt=3?
·(@sw←N5·@pt←0)U@pt=2?·(@sw←N3·@pt←0)U@pt=1?·(@sw←N2·@pt←0)U@pt=0?
·(@sw←N0·@pt←0))U@sw=N0?·(@pt=0?·(@sw←N1·@pt←0)))
```

Switches

```
@sw=N5?·(@dst=N4?·@pt←2U@dst=N3?·@pt←1U@dst=N2?·@pt←0U@dst=N1?
·@pt←0U@dst=N0?·@pt←0U@dst=N5?·@pt←-1)U@sw=N4?·(@dst=N5?
·@pt←1U@dst=N3?·@pt←0U@dst=N2?·@pt←0U@dst=N1?·@pt←0U@dst=N0?
·@pt←0U@dst=N4?·@pt←-1)U@sw=N3?·(@dst=N5?·@pt←2U@dst=N4?
·@pt←1U@dst=N2?·@pt←0U@dst=N1?·@pt←0U@dst=N0?·@pt←0U@dst=N3?
·@pt←-1)U@sw=N2?·(@dst=N5?·@pt←0U@dst=N4?·@pt←0U@dst=N3?
·@pt←0U@dst=N1?·@pt←0U@dst=N0?·@pt←0U@dst=N2?·@pt←-1)U@sw=N1?
·(@dst=N5?·@pt←3U@dst=N4?·@pt←3U@dst=N3?·@pt←2U@dst=N2?
·@pt←1U@dst=N0?·@pt←0U@dst=N1?·@pt←-1)U@sw=N0?·(@dst=N5?
·@pt←0U@dst=N4?·@pt←0U@dst=N3?·@pt←0U@dst=N2?·@pt←0U@dst=N1?
·@pt←0U@dst=N0?·@pt←-1)
```

Evaluation



Symbolic representation

A NetKAT Automaton ...

- $\varepsilon \in S \longrightarrow \text{Packet} \longrightarrow \text{Packet Set}$
- $\delta \in S \longrightarrow \text{Packet} \longrightarrow (S * \text{Packet}) \text{ Set}$

Symbolic representation

There are a *lot* of packets (and even more relations on packets)

A NetKAT Automaton ...

- $\varepsilon \in S \rightarrow \text{Packet} \rightarrow \text{Packet Set}$
- $\delta \in S \rightarrow \text{Packet} \rightarrow (S * \text{Packet}) \text{ Set}$

Symbolic representation

There are a *lot* of packets (and even more relations on packets)

A NetKAT Automaton ...

- $\varepsilon \in S \rightarrow \text{Packet} \rightarrow \text{Packet Set}$
- $\delta \in S \rightarrow \text{Packet} \rightarrow (S * \text{Packet}) \text{ Set}$

Idea: encode relations using a symbolic representation that is optimized for the common case—i.e., most fields are not changed

Symbolic representation

There are a *lot* of packets (and even more relations on packets)

A NetKAT Automaton ...

- $\varepsilon \in S \rightarrow \text{Packet} \rightarrow \text{Packet Set}$
- $\delta \in S \rightarrow \text{Packet} \rightarrow (S * \text{Packet}) \text{ Set}$

Idea: encode relations using a symbolic representation that is optimized for the common case—i.e., most fields are not changed

Symbolic representation

There are a *lot* of packets (and even more relations on packets)

A NetKAT Automaton ...

- $\varepsilon \in S \rightarrow \text{Packet} \rightarrow \text{Packet Set}$
- $\delta \in S \rightarrow \text{Packet} \rightarrow (S * \text{Packet}) \text{ Set}$

Idea: encode relations using a symbolic representation that is optimized for the common case—i.e., most fields are not changed

SPPs: two-layer binary decision diagrams where first layer encodes predicates and second layer encodes modifications.

Symbolic representation

There are a *lot* of packets (and even more relations on packets)

A NetKAT Automaton ...

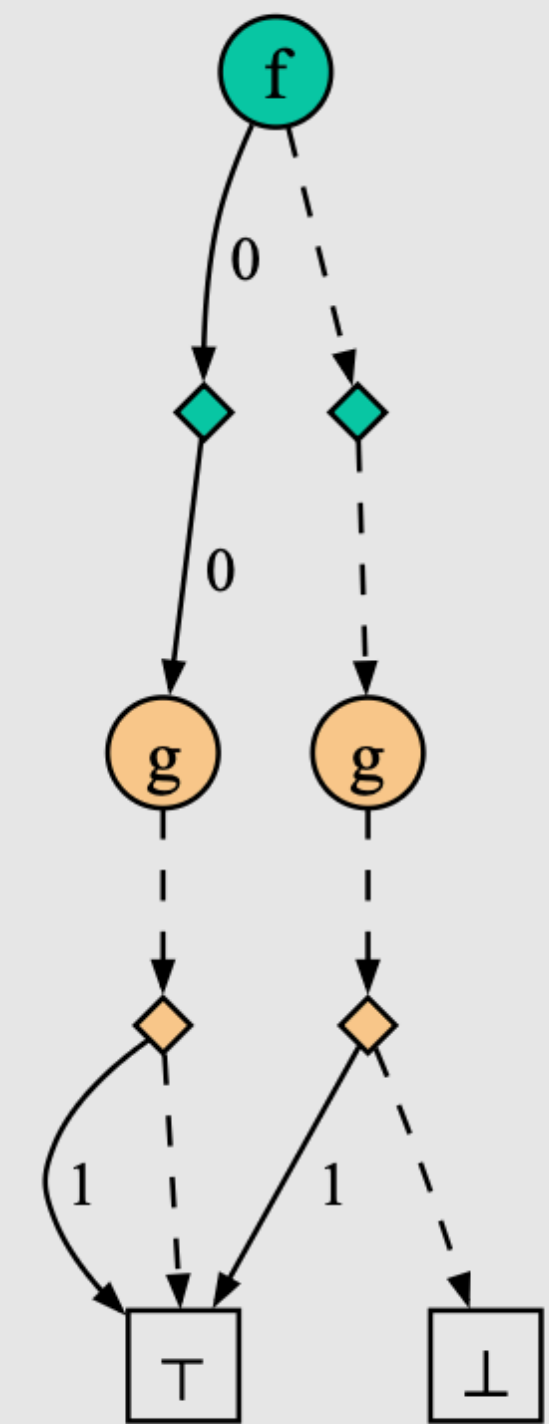
- $\varepsilon \in S \rightarrow \text{Packet} \rightarrow \text{Packet Set}$
- $\delta \in S \rightarrow \text{Packet} \rightarrow (S * \text{Packet}) \text{ Set}$

Idea: encode relations using a symbolic representation that is optimized for the common case—i.e., most fields are not changed

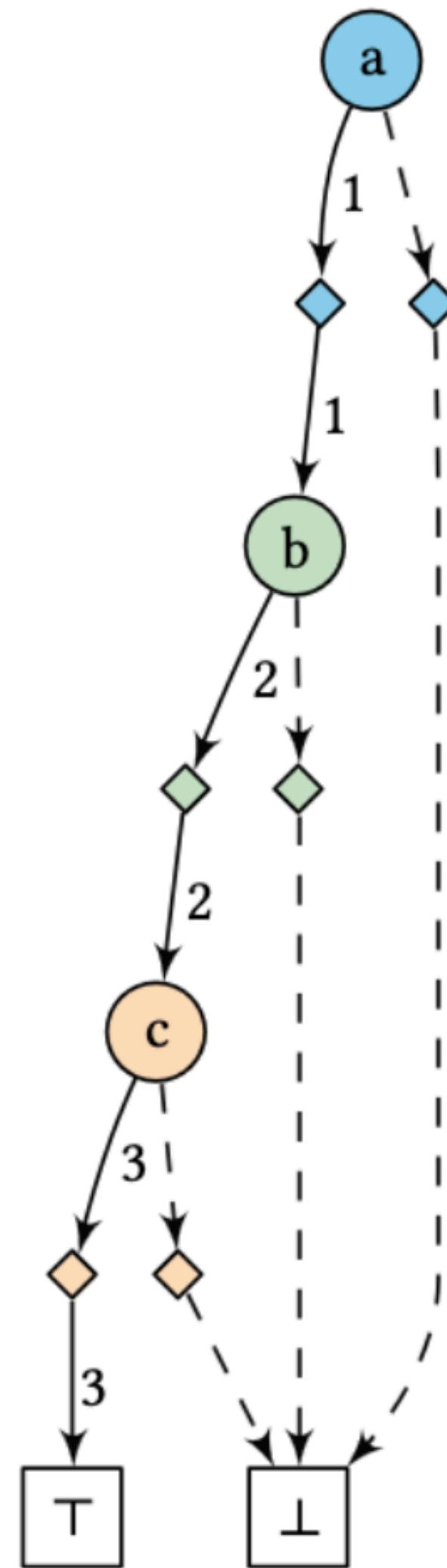
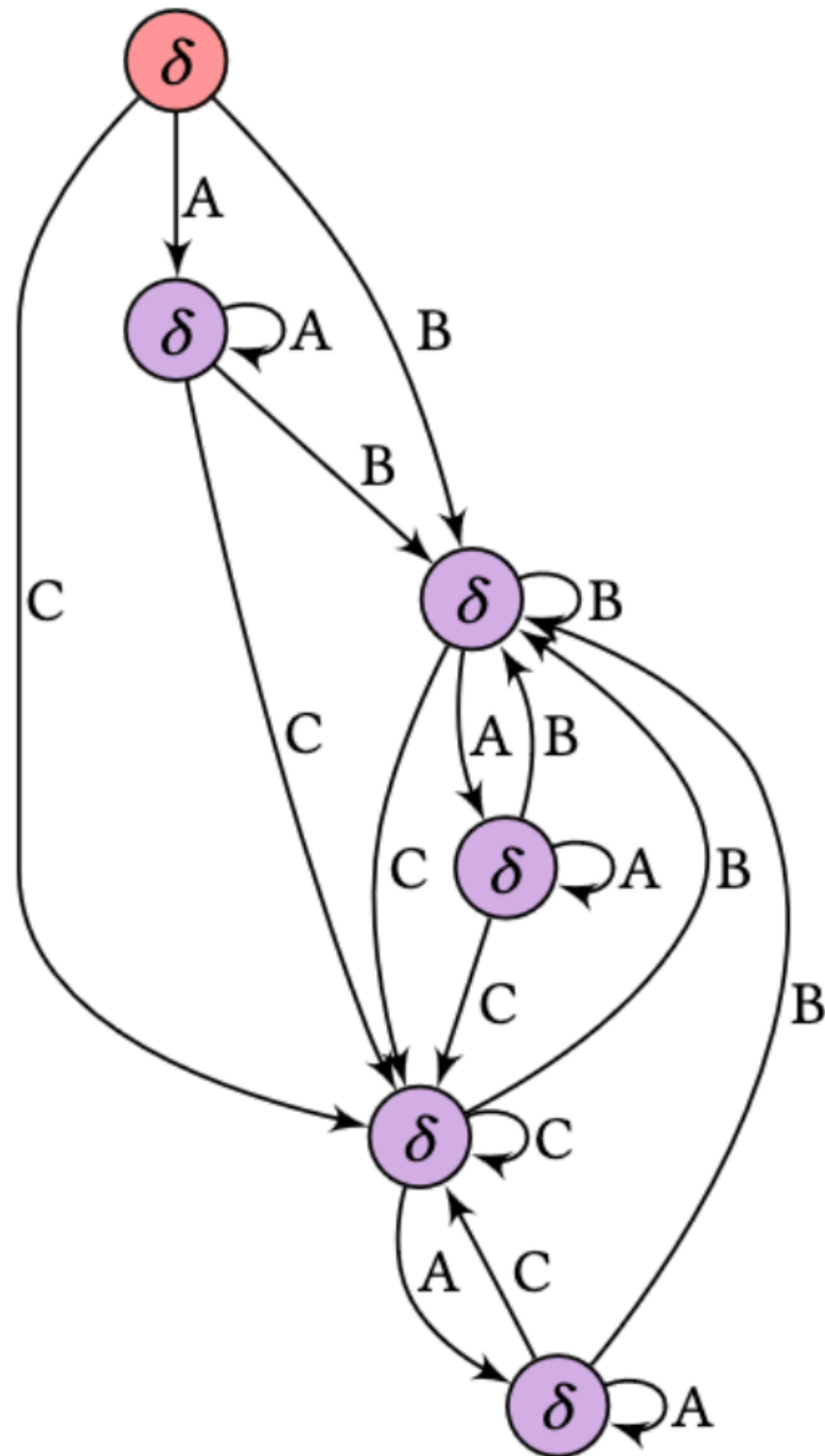
SPPs: two-layer binary decision diagrams where first layer encodes predicates and second layer encodes modifications.

Example

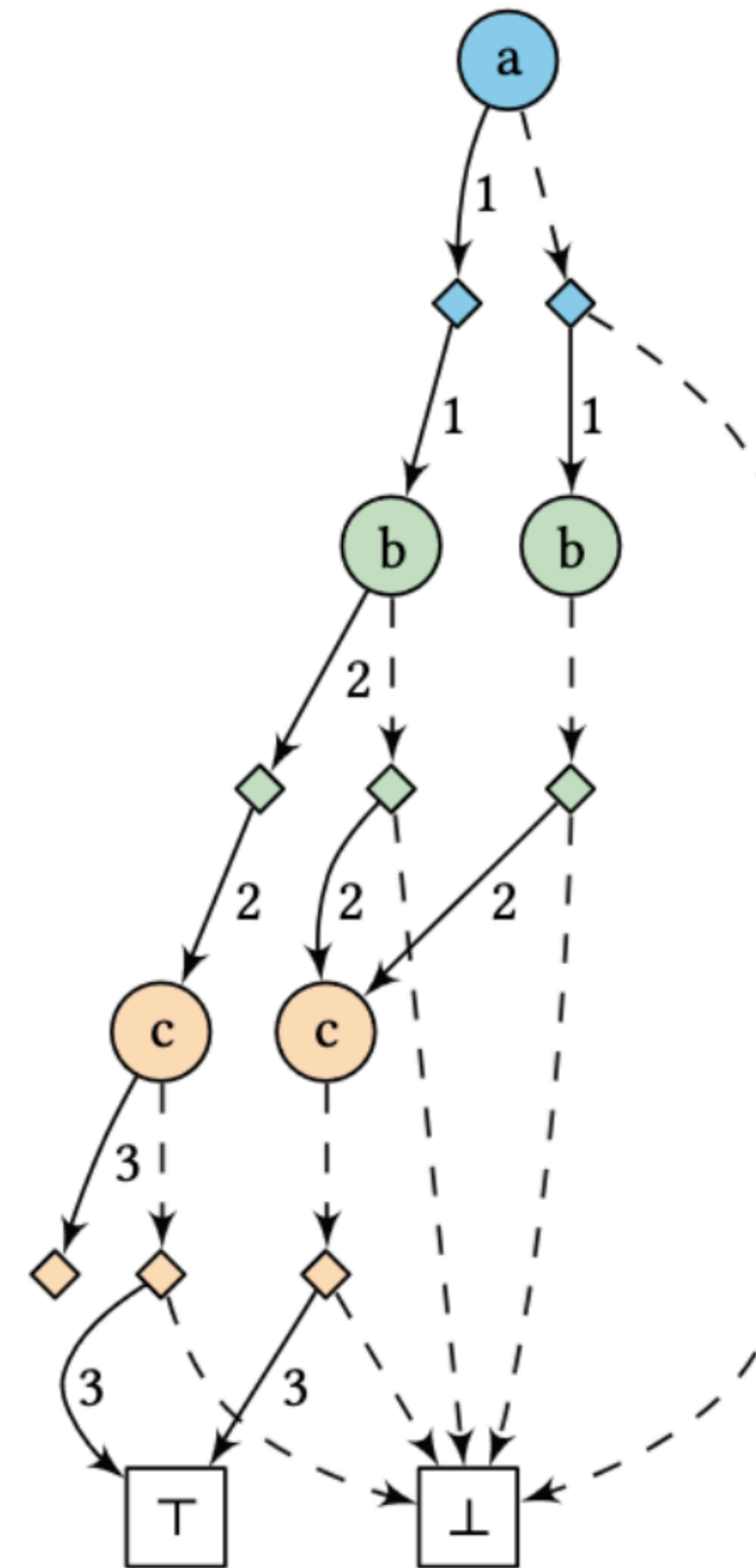
$f = 0 + g := 1$



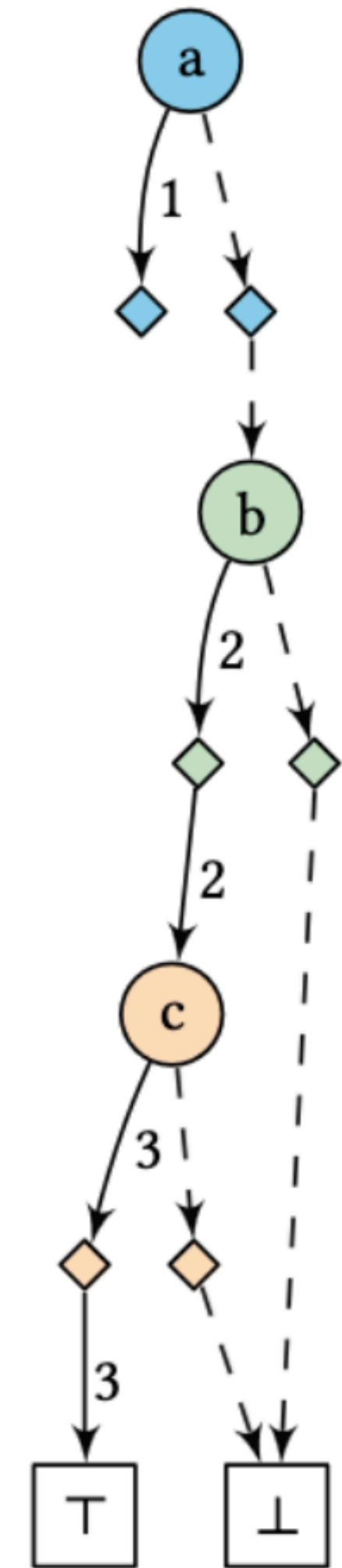
Symbolic representation - automata



A

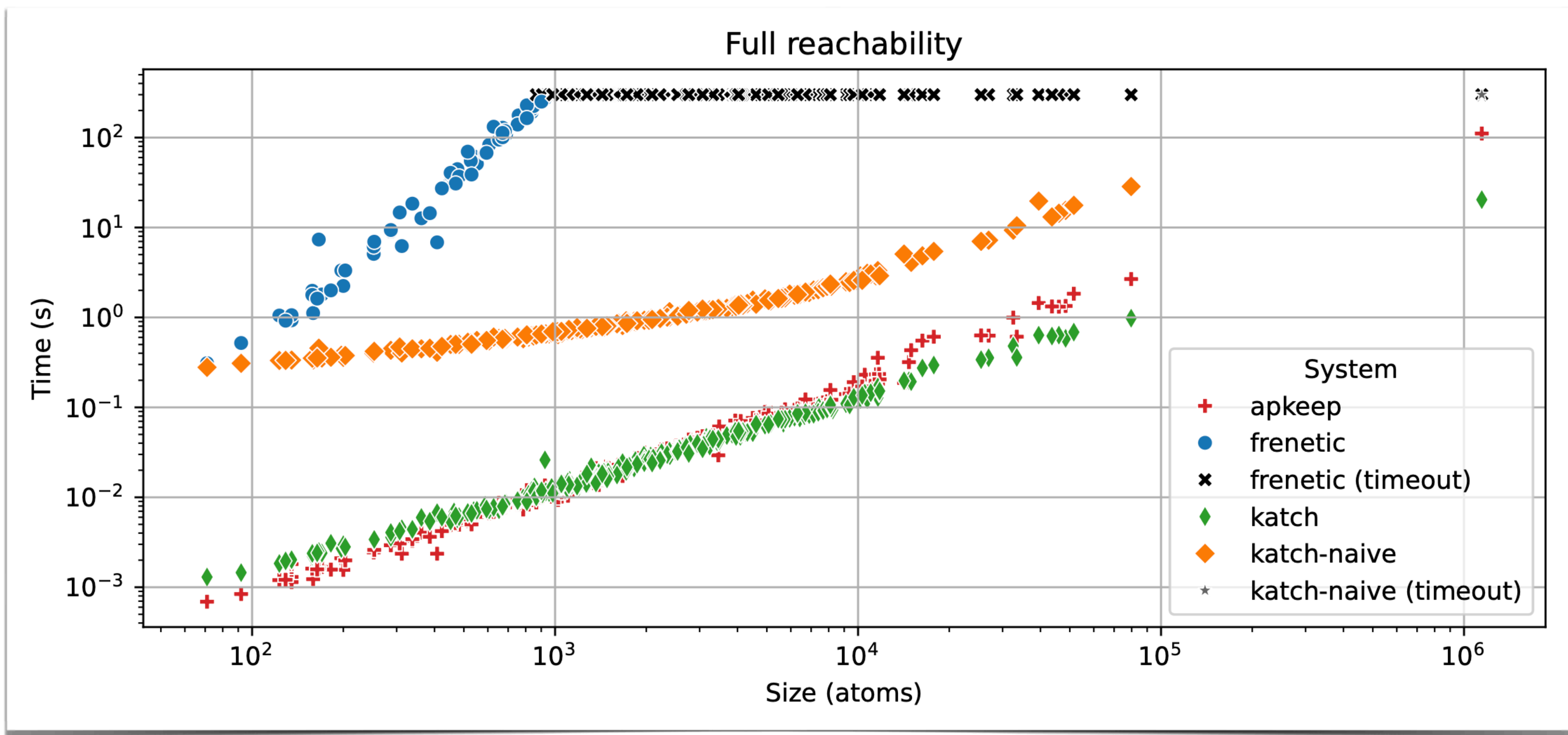


B

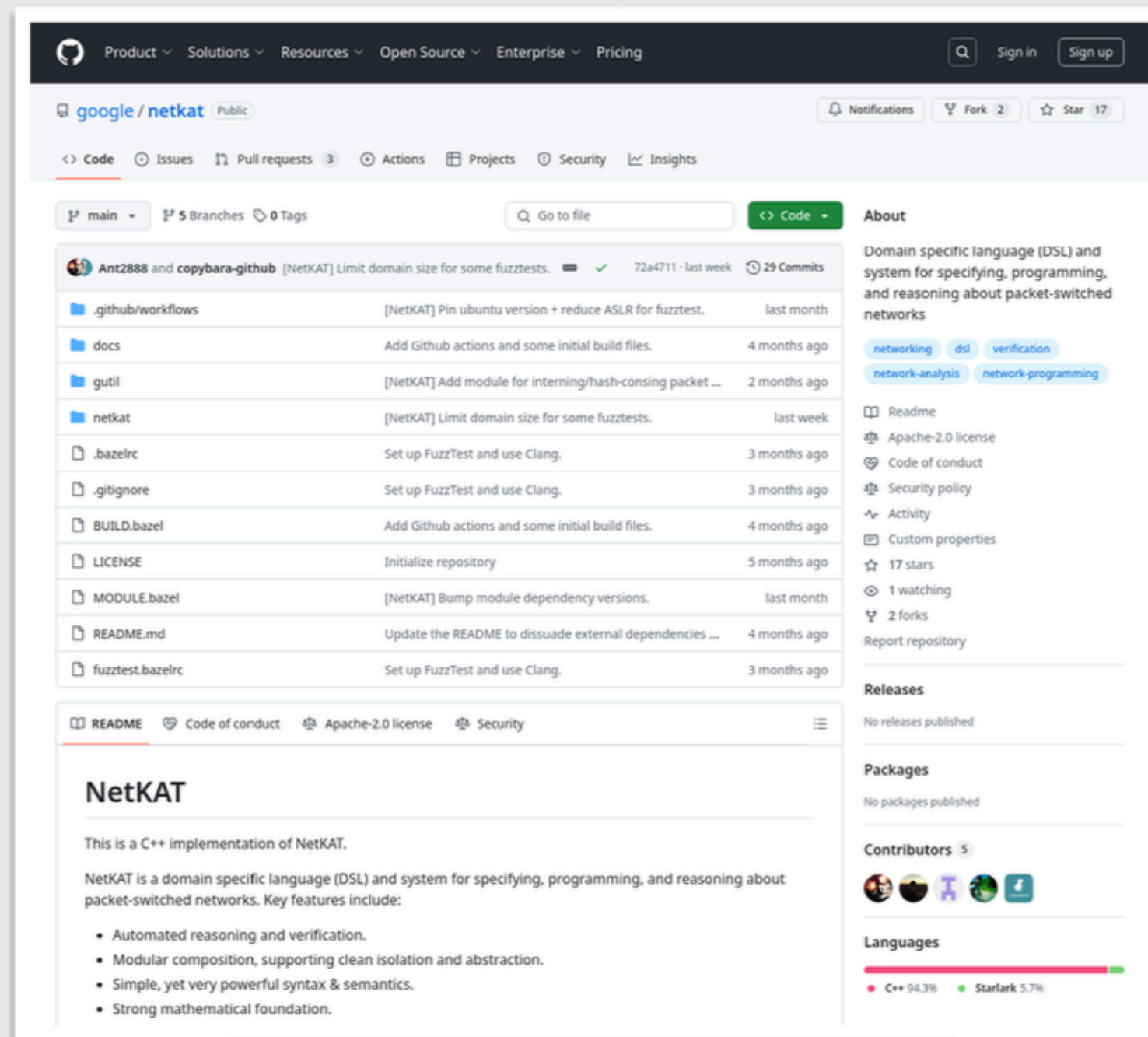


C

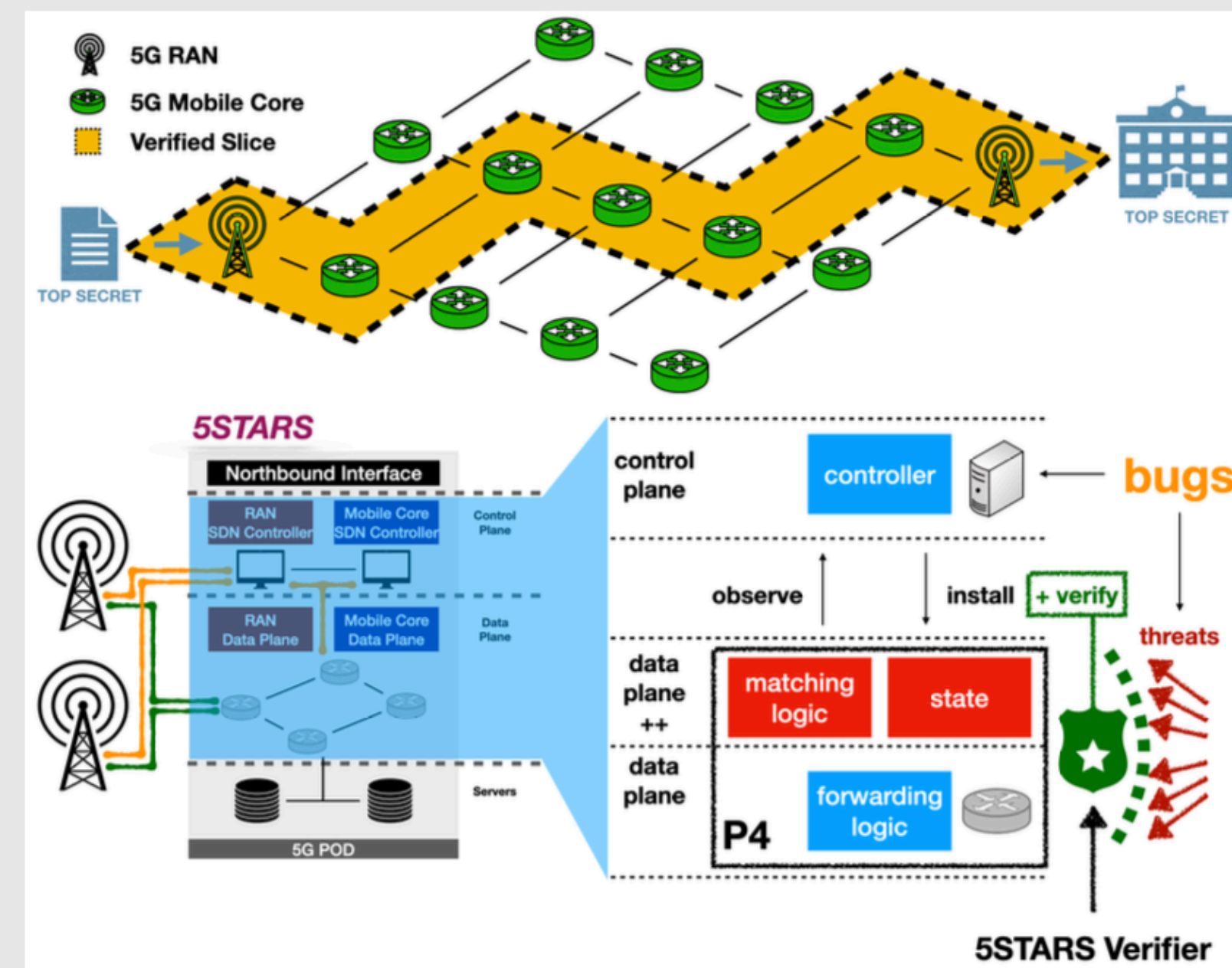
Evaluation: KATch



NetKAT in industry

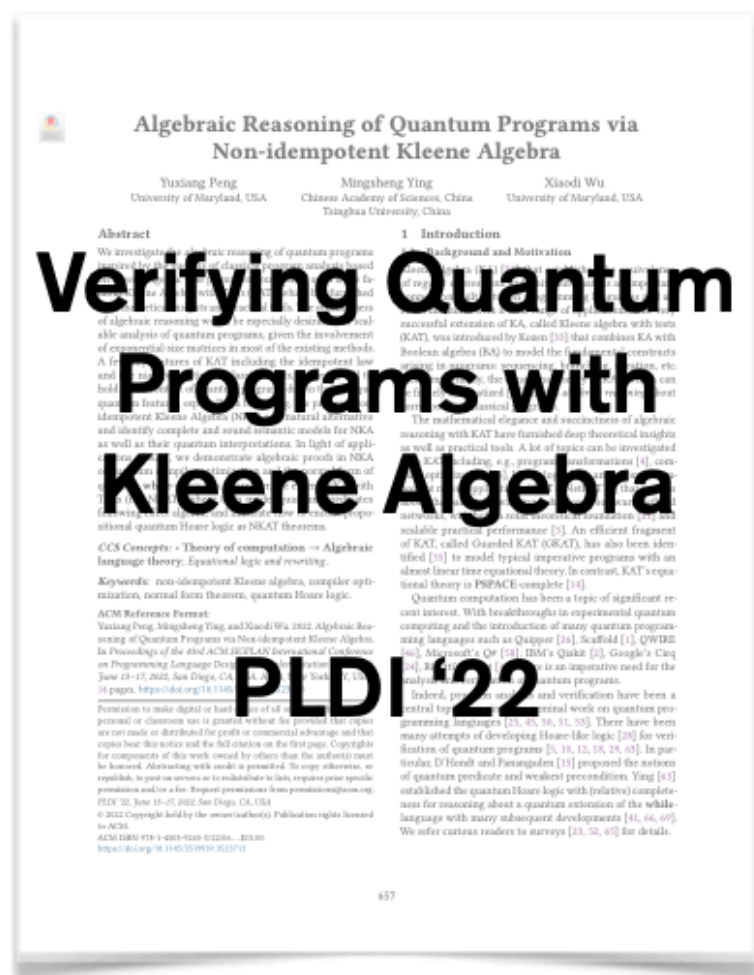
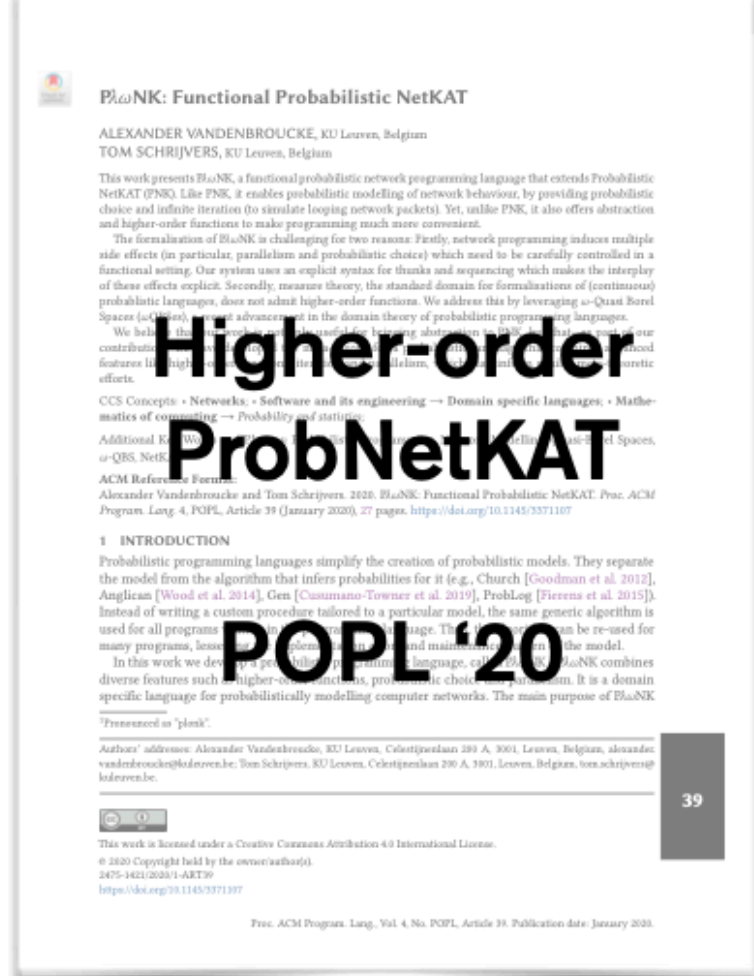
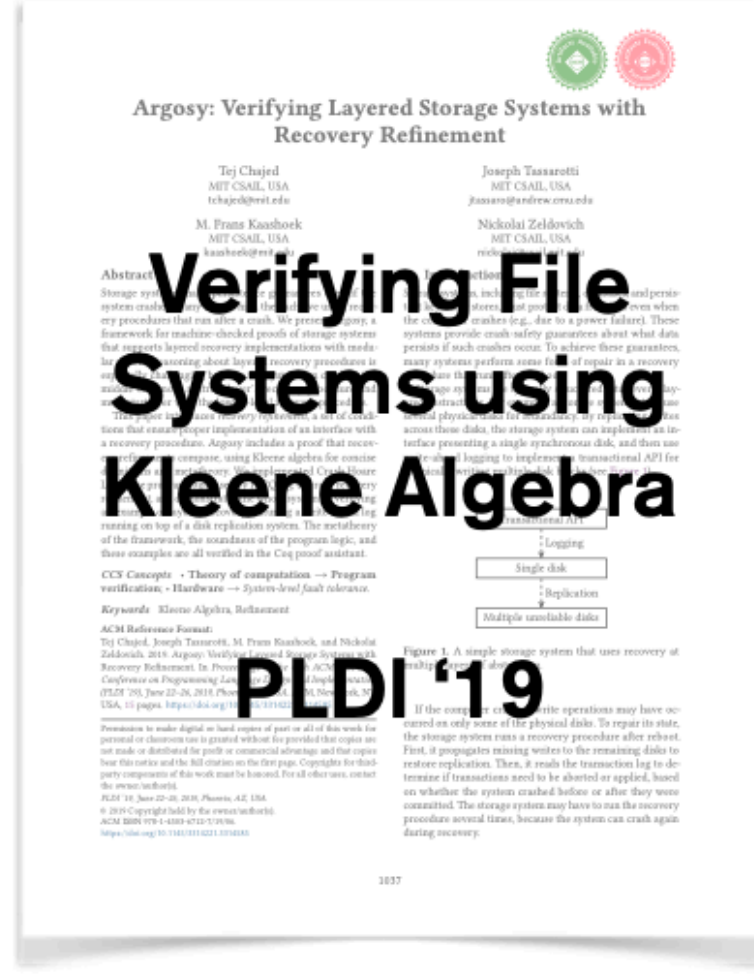
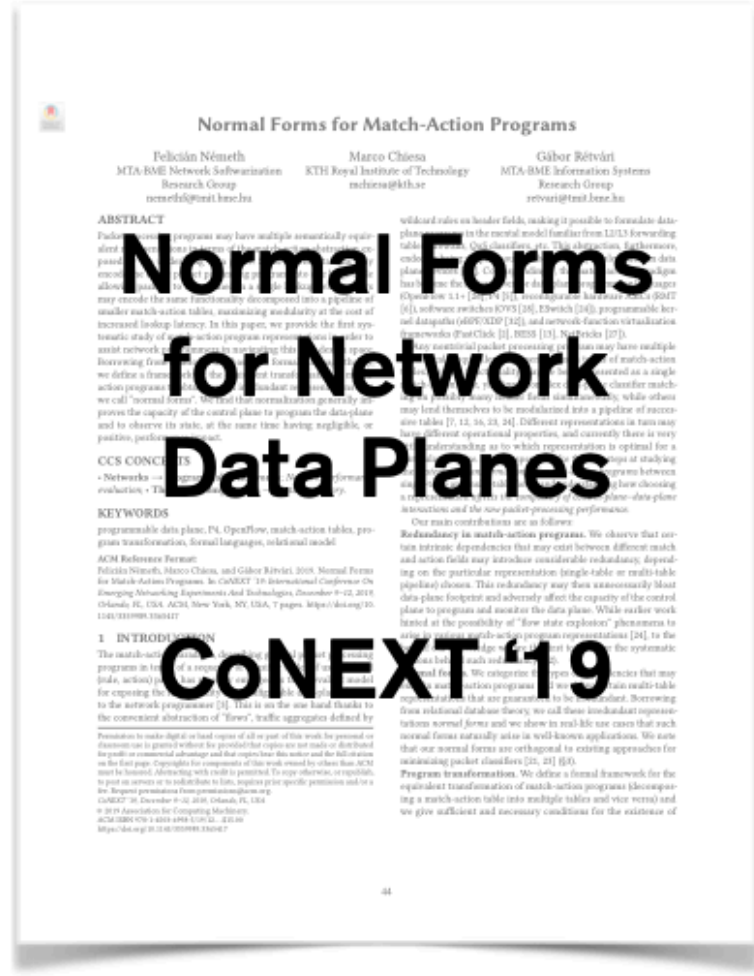
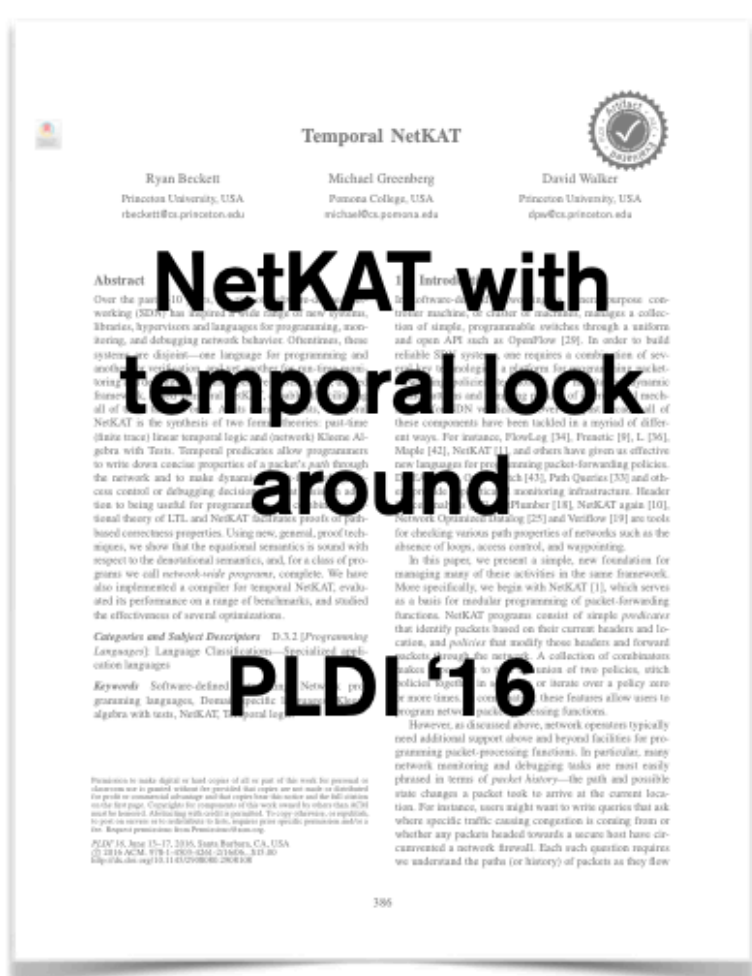


C++ implementation of NetKAT
for verifying cloud isolation

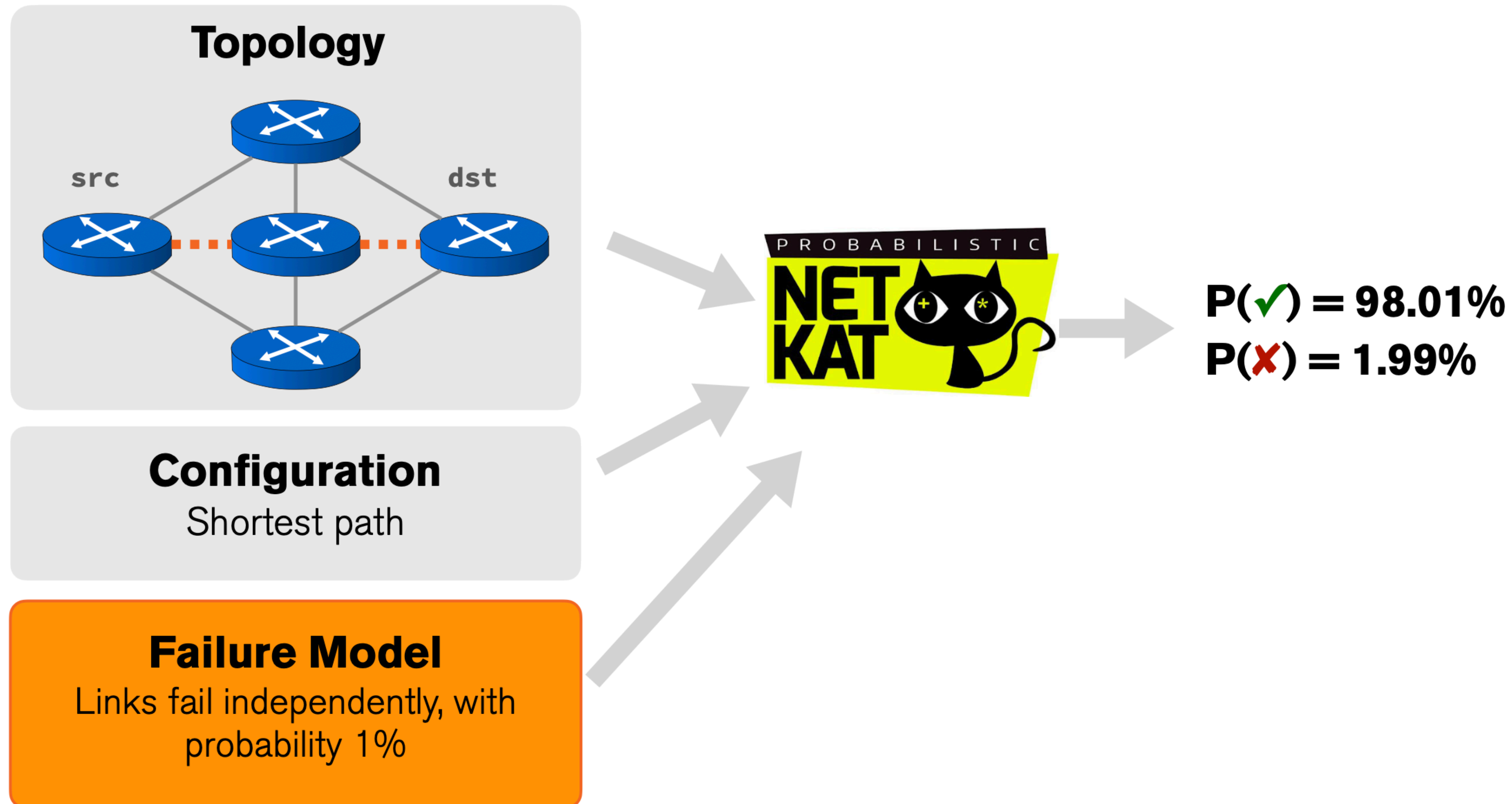


Haskell implementation of NetKAT
for verifying secure 5G slicing

Lots of KATs



Beyond deterministic reasoning

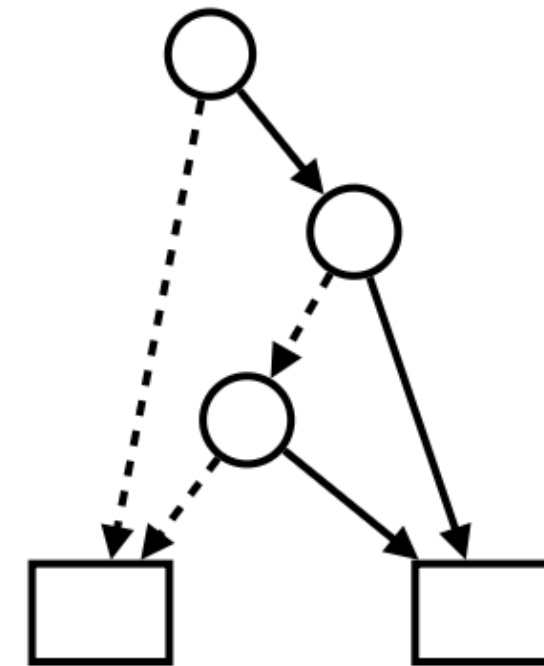


Beyond deterministic reasoning

```
if port=1 then
  port ← 2  $\oplus_{0.5}$  port ← 3
else if port=2 then
  port ← 1
else
  drop
```

Network Model
= ProbNetKAT program
[ESOP '16, POPL '17]

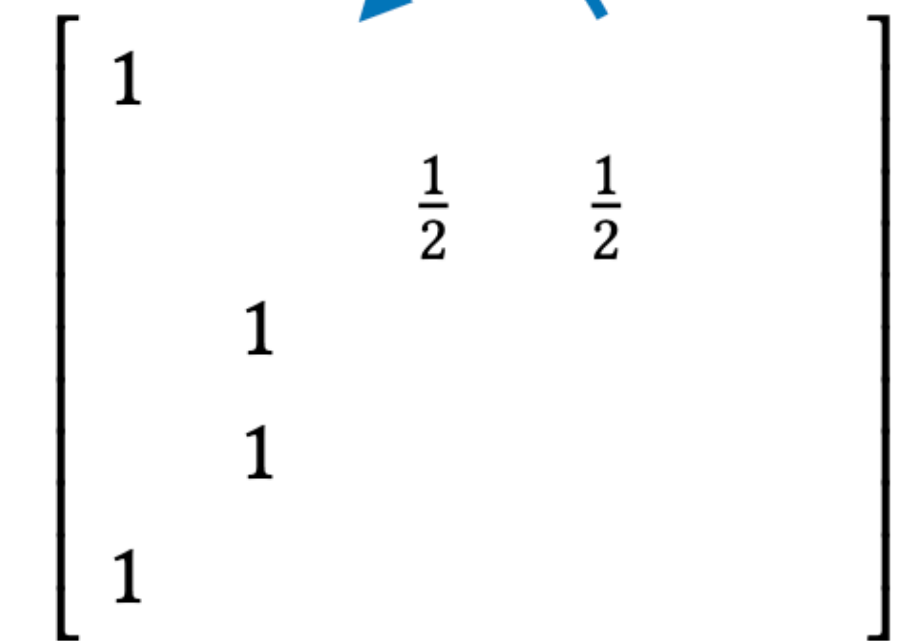
compile



Symbolic IR

[ICFP '15, PLDI '24]

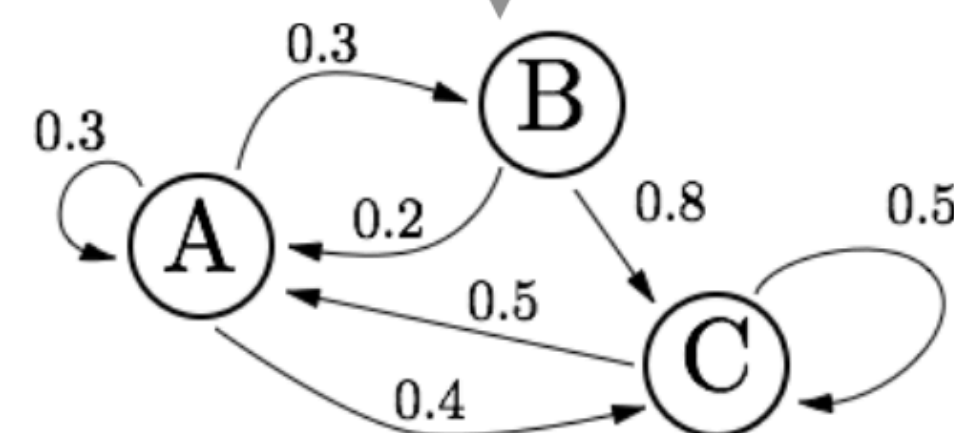
convert



sparse matrix

compute limiting distribution

[-]



Markov chain

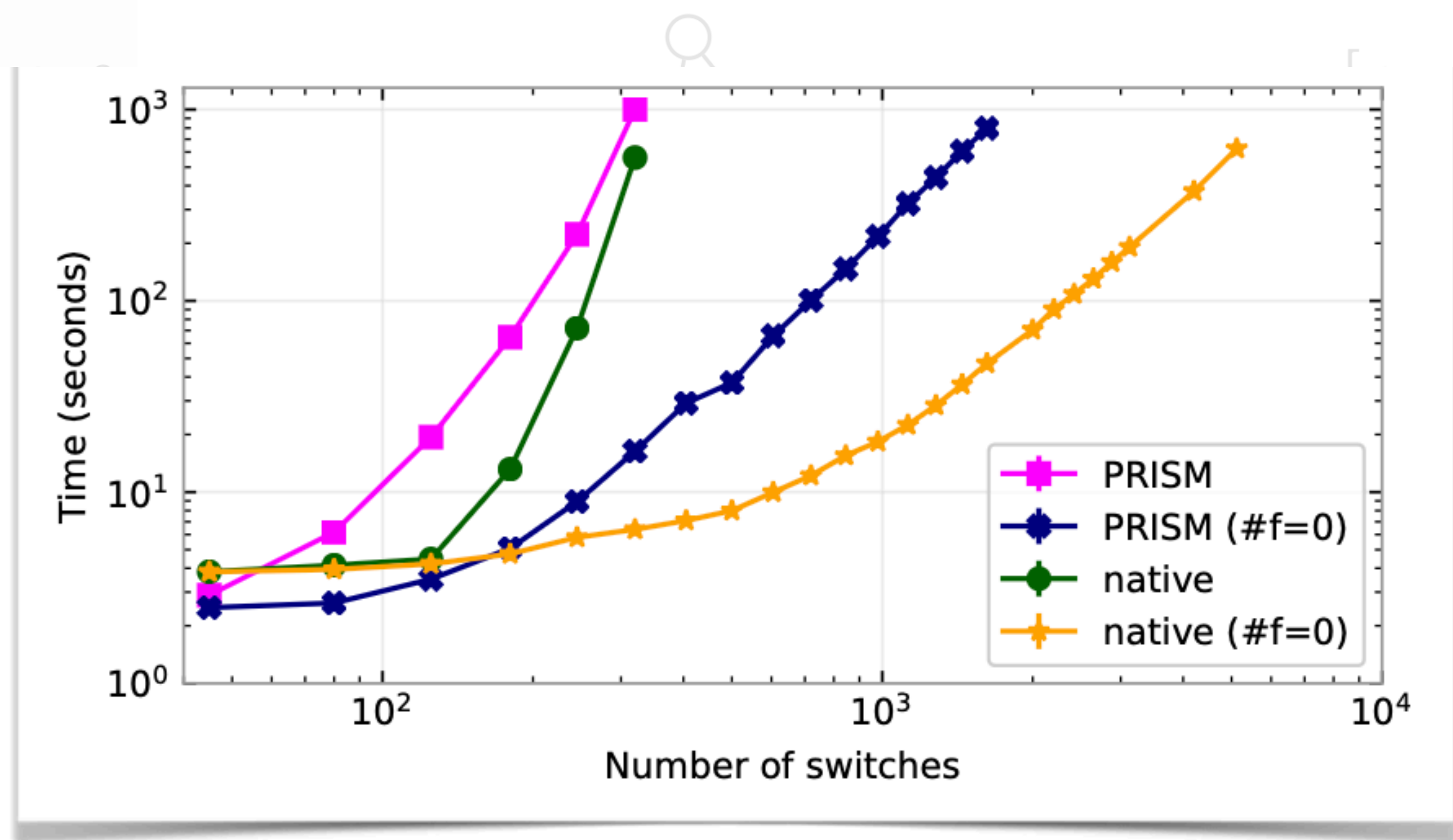
[-]

Beyond deterministic reasoning

```
if port=1 then
  port←2 &
else if por
  port←1
else
  drop
```

Netwo
= ProbNet
[ESOP '10]

compute limiting distribution



$\frac{1}{2}$ $\frac{1}{2}$

e matrix



Markov chain

Conclusion

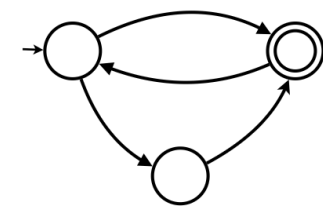
“Programming languages (in particular, domain-specific languages based on solid foundations) have a key role to play in formal verification of current and future systems”

Conclusion

“Programming languages (in particular, domain-specific languages based on solid foundations) have a key role to play in formal verification of current and future systems”

Theory

$\llbracket p \rrbracket$
denotational semantics



automata theory

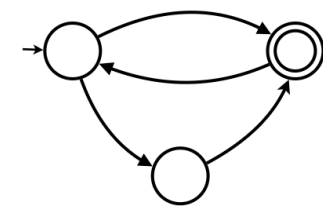
$\vdash p \equiv q$
sound & complete axiomatization

Conclusion

“Programming languages (in particular, domain-specific languages based on solid foundations) have a key role to play in formal verification of current and future systems”

Theory

$\llbracket p \rrbracket$
denotational semantics



automata theory

$\vdash p \equiv q$
sound & complete axiomatization

Applications

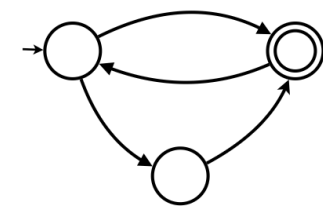
Network Verification
Probabilistic Reasoning
Learning
Compilation and Synthesis
...

Conclusion

“Programming languages (in particular, domain-specific languages based on solid foundations) have a key role to play in formal verification of current and future systems”

Theory

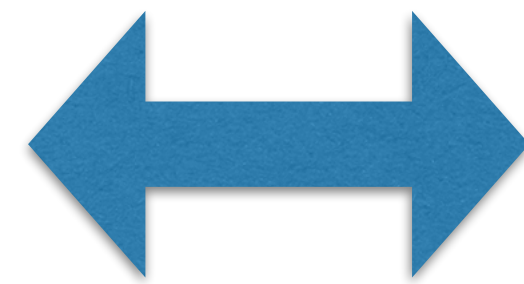
$\llbracket p \rrbracket$
denotational semantics



automata theory

$\vdash p \equiv q$
sound & complete axiomatization

Enables



Inspires

Applications

Network Verification
Probabilistic Reasoning
Learning
Compilation and Synthesis
...