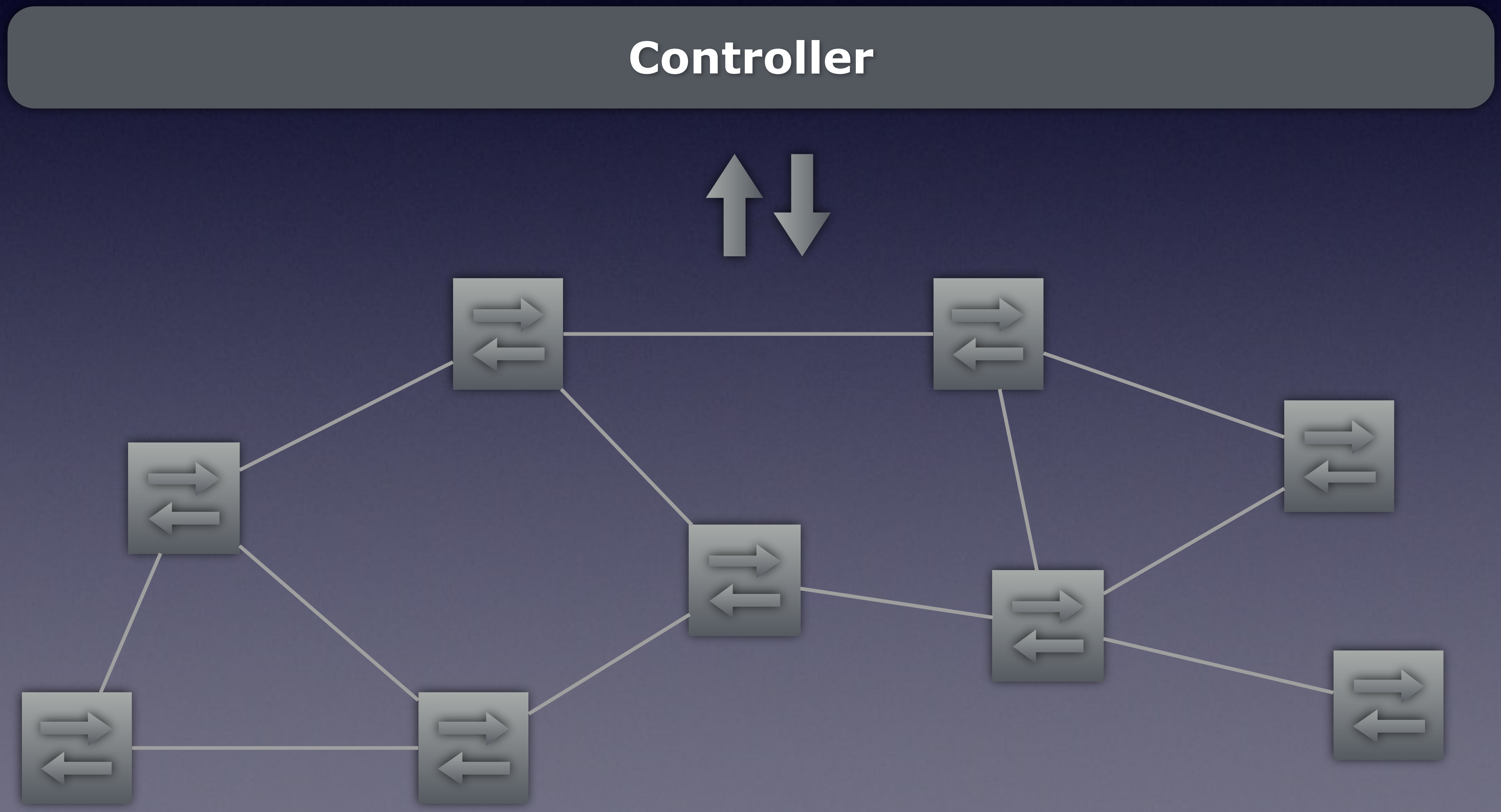
Semantic Foundations for Network Programming Language

Software-Defined Networking

Key ideas: generalize devices, separate control and forwarding



Current Controllers

One monolithic application

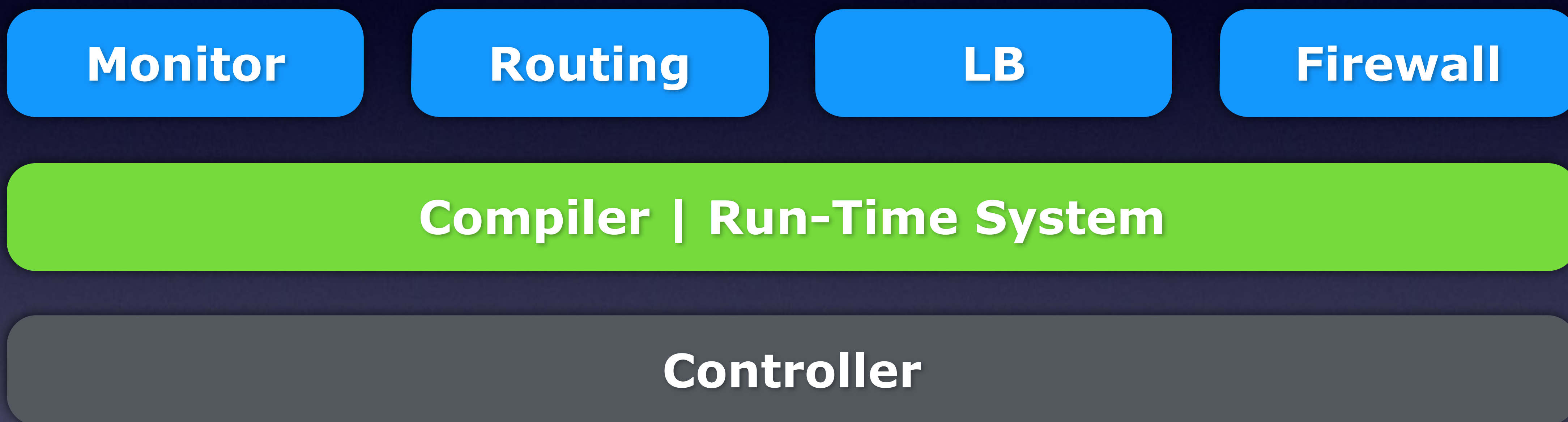
Monitor | Routing | Load Balancing | Firewall | etc.

Controller

Challenges:

- Writing, testing, and debugging programs
- Reusing code across application
- Porting application to new platforms

Language-Based Controllers



Benefits:

- Easier to write, test, and debug programs
- Easy to reuse modules across applications
- Possible to port application to new platforms

Balance of Power

- Tradeoffs:
  - Analyzability
  - Expressiveness

“A balance of power: expressive, analyzable controller programming,” HotSDN '13

Language #1

- **Frenetic: A Network Programming Language [ICFP ’11]**
- Key ideas:
  - A language abstraction between programs and hardware
  - Constructs for reading state and specifying forwarding policies
  - Support for modular composition through policy combinators
  - Run-time system pushes rules to switches **reactively**

Language #2

- **A Compiler and Run-time System for Network Programming Languages [POPL ’12]**
- Key ideas:
  - NetCore policy language
  - Compiler pushes forwarding rules to switches **proactively**
  - Reactive specialization handles features that cannot be translated

Language #3

- **Composing Software-Defined Networks [NSDI '13]**
- Key ideas:
  - NetCore
  - **Sequential composition**
  - Virtual fields

Language #4

- **Machine-Verified Network Controllers [PLDI ’13]**
- Key ideas:
  - Network-wide semantics
  - Detailed “featherweight” model of SDN
  - Machine-checked proofs of correctness in Coq
  - First real deployment

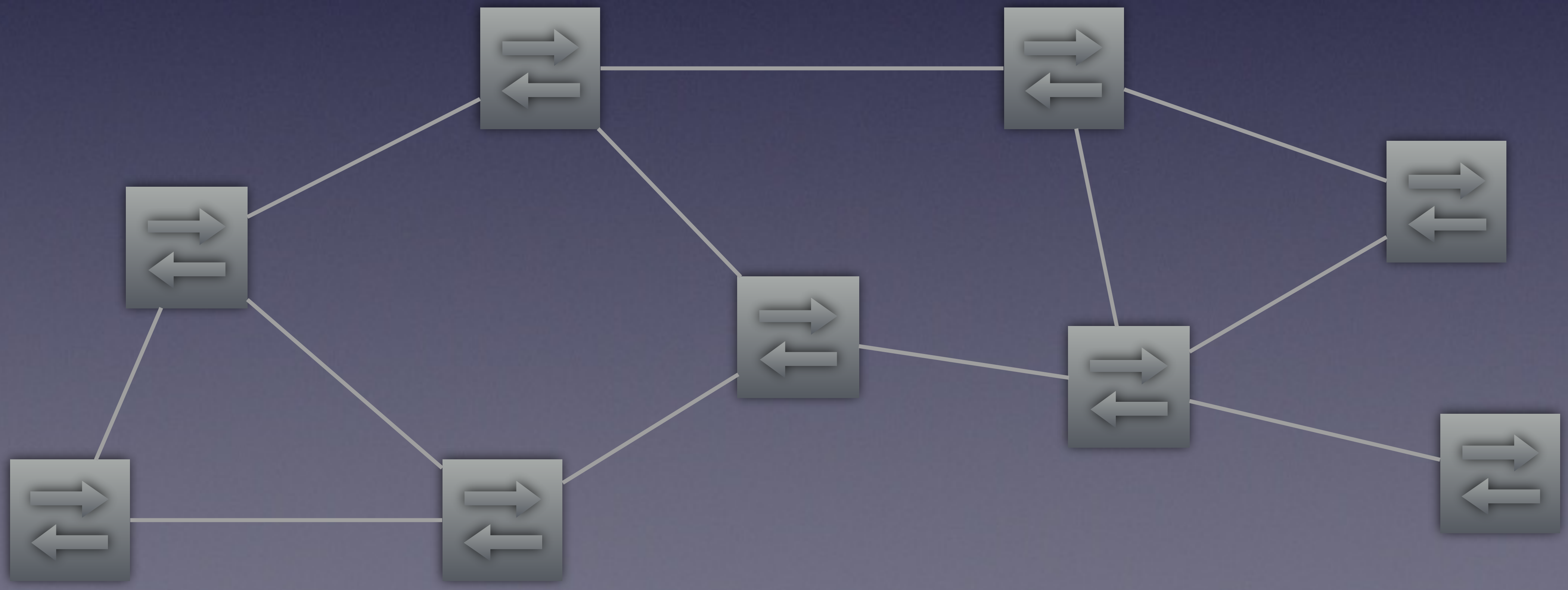
Summary

- Key design choices revisited on each iteration
- Each semantics had a precise definition but was rather **ad hoc**
- Unclear how new features should interact with old ones
- Could not **reason equationally** about network-wide behavior

- *“Can X connect to Y?”*
- *“Is traffic from A to B routed through Z?”*
- *“Is there a loop involving S?”*

Language Features

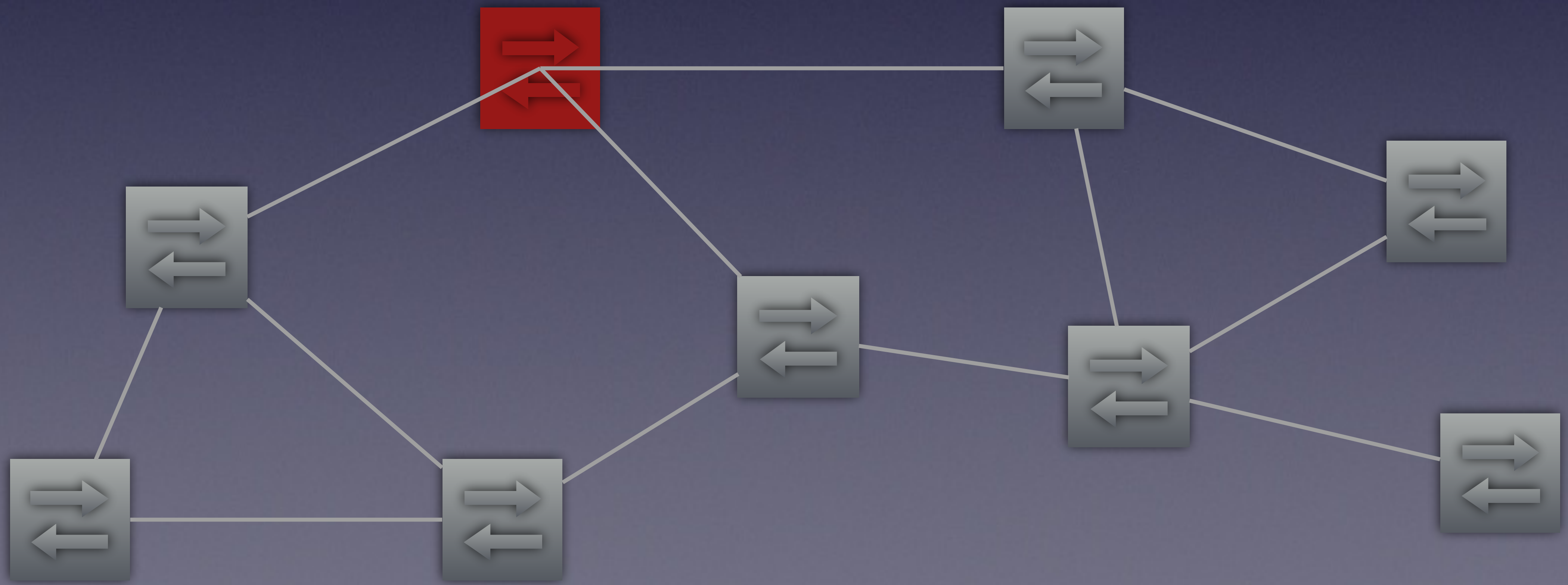
What features should a NPL provide?



Language Features

What features should a NPL provide?

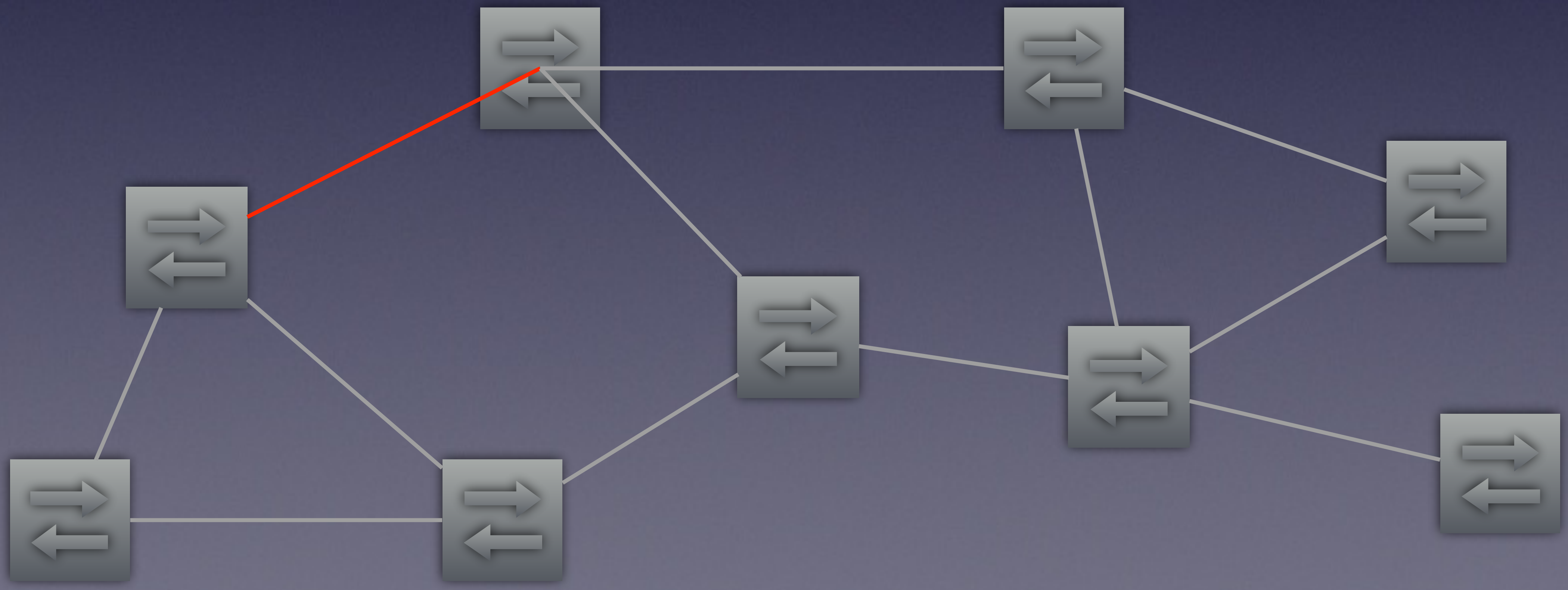
- Packet predicates
- Packet transformations



Language Features

What features should a NPL provide?

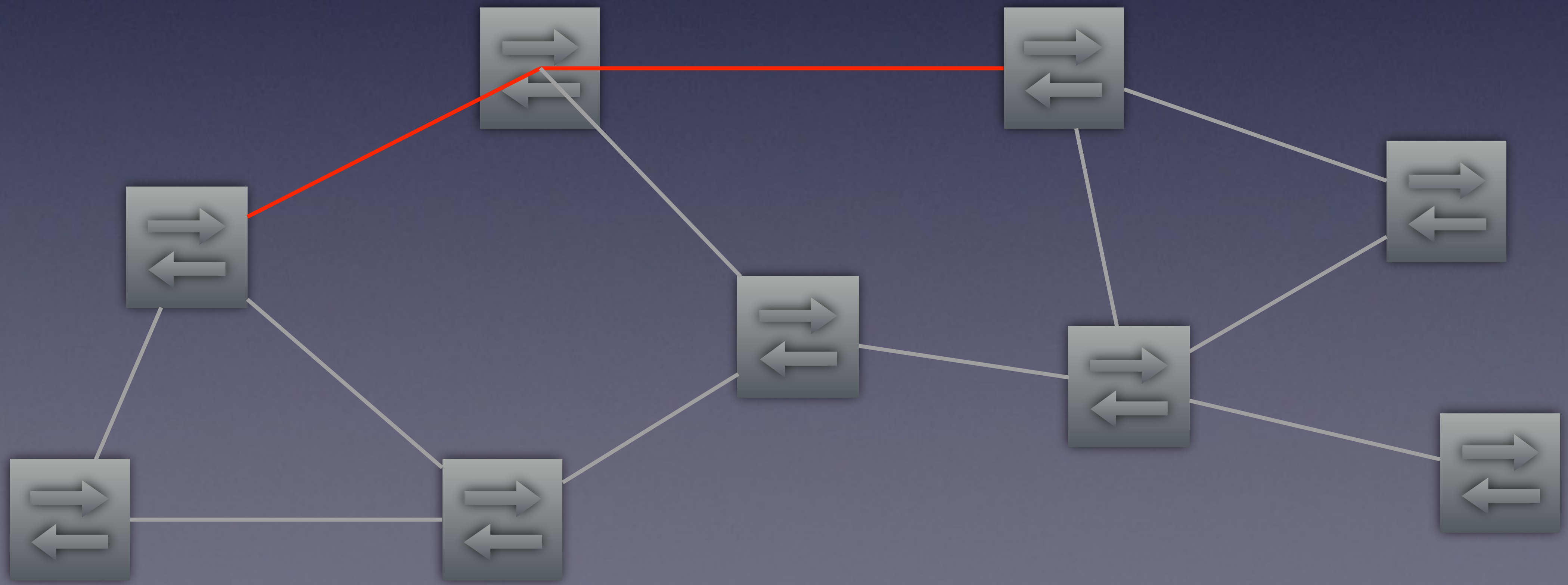
- Path construction



Language Features

What features should a NPL provide?

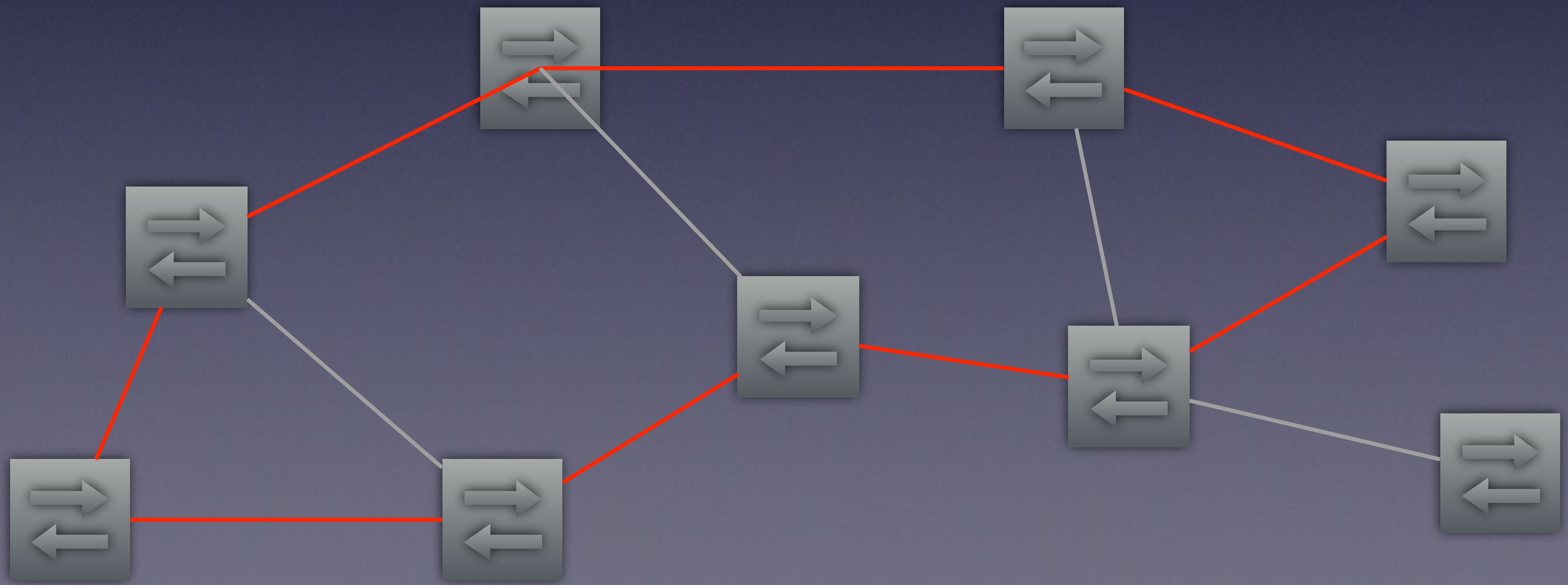
- Path construction
- Path concatenation



Language Features

What features should a NPL provide?

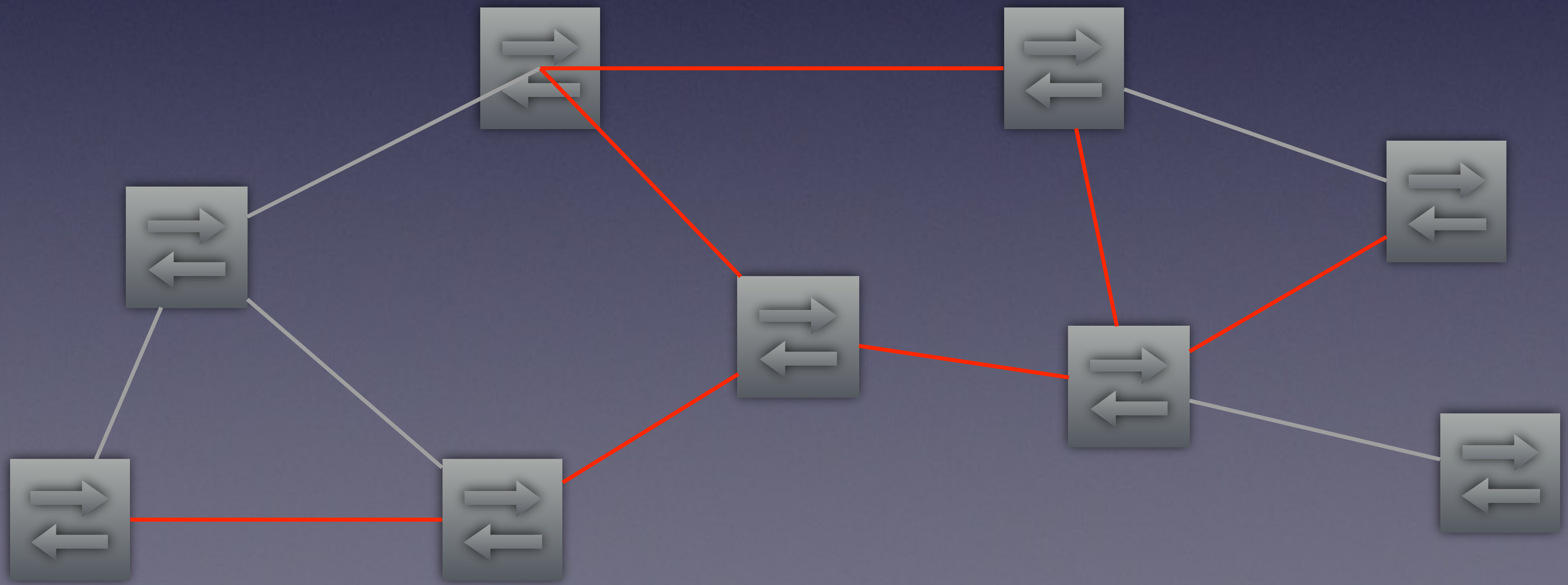
- Path construction
- Path concatenation
- Path union



Language Features

What features should a NPL provide?

- Path construction
- Path concatenation
- Path union
- Path iteration



NetKAT

```
f :: = switch | inport | srcmac | dstmac | …
v :: = 0 | 1 | 2 | 3 | …
a,b,c ::=  true                            (* true *)
         | false                           (* false *)
         | f = v                           (* test *)
         | a1 | a2                         (* disjunction *)
         | a1 & a2                         (* conjunction *)
         | !a                              (* negation *)
p,q,r ::=  filter a                        (* filter *)
         | f := v                          (* modification *)
         | p1 | p2                         (* union *)
         | p1 ; p2                         (* sequence *)
         | p*                              (* iteration *)
         | dup                             (* duplication *)
```

```
   if a then p1 else p2 ==
(filter a; p1) | (filter !a; p2)
```

Syntax

Fields	$f ::=$	$f_1 \mid \cdots \mid f_k$	
Packets	$pk ::=$	$\{f_1 = v_1, \cdots, f_k = v_k\}$	
Histories	$h ::=$	$pk::\langle\rangle \mid pk::h$	
Predicates	$a, b ::=$	1	*Identity*
	$\mid$	0	*Drop*
	$\mid$	$f = n$	*Test*
	$\mid$	$a + b$	*Disjunction*
	$\mid$	$a \cdot b$	*Conjunction*
	$\mid$	$\neg a$	*Negation*
Policies	$p, q ::=$	a	*Filter*
	$\mid$	$f \leftarrow n$	*Modification*
	$\mid$	$p + q$	*Union*
	$\mid$	$p \cdot q$	*Sequential composition*
	$\mid$	p^*	*Kleene star*
	$\mid$	dup	*Duplication*

Semantics

$$[\![p]\!] \in \mathsf{H} \to \mathcal{P}(\mathsf{H})$$

$$[\![1]\!]\ h \triangleq \{h\}$$

$$[\![0]\!]\ h \triangleq \{\}$$

$$[\![f = n]\!]\ (pk::h) \triangleq \begin{cases} \{pk::h\} & \text{if } pk.f = n \\ \{\} & \text{otherwise} \end{cases}$$

$$[\![\neg a]\!]\ h \triangleq \{h\} \setminus ([\![a]\!]\ h)$$

$$[\![f \leftarrow n]\!]\ (pk::h) \triangleq \{pk[f := n]::h\}$$

$$[\![p + q]\!]\ h \triangleq [\![p]\!]\ h \cup [\![q]\!]\ h$$

$$[\![p \cdot q]\!]\ h \triangleq ([\![p]\!] \bullet [\![q]\!])\ h$$

$$[\![p^*]\!]\ h \triangleq \bigcup_{i \in \mathbb{N}} F^i\ h$$

$$\text{where } F^0\ h \triangleq \{h\} \text{ and } F^{i+1}\ h \triangleq ([\![p]\!] \bullet F^i)\ h$$

$$[\![\mathsf{dup}]\!]\ (pk::h) \triangleq \{pk::(pk::h)\}$$

Kleene Algebra Axioms

$p + (q + r) \equiv (p + q) + r$	KA-PLUS-ASSOC
$p + q \equiv q + p$	KA-PLUS-COMM
$p + 0 \equiv p$	KA-PLUS-ZERO
$p + p \equiv p$	KA-PLUS-IDEM
$p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$	KA-SEQ-ASSOC
$1 \cdot p \equiv p$	KA-ONE-SEQ
$p \cdot 1 \equiv p$	KA-SEQ-ONE
$p \cdot (q + r) \equiv p \cdot q + p \cdot r$	KA-SEQ-DIST-L
$(p + q) \cdot r \equiv p \cdot r + q \cdot r$	KA-SEQ-DIST-R
$0 \cdot p \equiv 0$	KA-ZERO-SEQ
$p \cdot 0 \equiv 0$	KA-SEQ-ZERO
$1 + p \cdot p^* \equiv p^*$	KA-UNROLL-L
$q + p \cdot r \leq r \Rightarrow p^* \cdot q \leq r$	KA-LFP-L
$1 + p^* \cdot p \equiv p^*$	KA-UNROLL-R
$p + q \cdot r \leq q \Rightarrow p \cdot r^* \leq q$	KA-LFP-R

Additional Boolean Algebra Axioms

$a + (b \cdot c) \equiv (a + b) \cdot (a + c)$	BA-PLUS-DIST
$a + 1 \equiv 1$	BA-PLUS-ONE
$a + \neg a \equiv 1$	BA-EXCL-MID
$a \cdot b \equiv b \cdot a$	BA-SEQ-COMM
$a \cdot \neg a \equiv 0$	BA-CONTRA
$a \cdot a \equiv a$	BA-SEQ-IDEM

Packet Algebra Axioms

$f \leftarrow n \cdot f' \leftarrow n' \equiv f' \leftarrow n' \cdot f \leftarrow n$, if $f \neq f'$	PA-MOD-MOD-COMM
$f \leftarrow n \cdot f' = n' \equiv f' = n' \cdot f \leftarrow n$, if $f \neq f'$	PA-MOD-FILTER-COMM
$\mathsf{dup} \cdot f = n \equiv f = n \cdot \mathsf{dup}$	PA-DUP-FILTER-COMM
$f \leftarrow n \cdot f = n \equiv f \leftarrow n$	PA-MOD-FILTER
$f = n \cdot f \leftarrow n \equiv f = n$	PA-FILTER-MOD
$f \leftarrow n \cdot f \leftarrow n' \equiv f \leftarrow n'$	PA-MOD-MOD
$f = n \cdot f = n' \equiv 0$, if $n \neq n'$	PA-CONTRA
$\sum_i f = i \equiv 1$	PA-MATCH-ALL

Figure 2. NetKAT: syntax, semantics, and equational axioms.

Basic Primitives

```
if srcip = 10.0.0.1 & !(dstport = 22) then
   port := 1
else
   port := 2
```

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

Union

```
if srcip=1.2.3.4
then port := 3
```

```
if dstip=10.0.0.1 then
port := 1
else if dstip=10.0.0.2
then port := 2
```

Monitor | **Route**

Controller

Pattern	Actions
srcip=1.2.3.4, dstip=10.0.0.1	Forward 1, Forward 3
srcip=1.2.3.4, dstip=10.0.0.2	Forward 2, Forward 3
srcip=1.2.3.4	Forward 3
dstip=10.0.0.1	Forward 1
dstip=10.0.0.2	Forward 2

Sequence

```
if srcip=*0 then dstip := 10.0.0.1
else if srcip=*1 then dstip :=
10.0.0.2
```

```
if dstip=10.0.0.1 then
port := 1
else if dstip=10.0.0.2
then port := 2
```

Load Balancing ; **Route**

Controller

Pattern	Actions
srcip=*0	dstip:=10.0.0.1,Forward 1
srcip=*1	dstip:=10.0.0.2,Forward 2

Iteration

```
if dstip=192.168.0.0/16 then
port := B
else if port=A & dstport=80 then
port := 1
```

```
if dstip=10.0.0.0/8 then
port := A
else if port=B & dstport=22
then port := 2
```

(**Tenant A** | **Tenant B**)*

Controller

Pattern	Actions
dstip=10.0.0.0/8, dstport=80	Forward 1
dstip=192.168.0.0/16, dstport=22	Forward 2
*	Drop

Semantic Foundation

- The foundation rests upon canonical mathematical structure:
  - Regular operators (|, ;, and *) encode paths through **topology**
  - Boolean operators (&, |, and !) encode **switch tables**
- This is called a **Kleene Algebra with Tests (KAT)** [Kozen ’96]

Semantic Foundation

- Kleene Algebra for reasoning about network structure
- Boolean Algebra for reasoning about predicates that define switch behavior.
- KAT (Kleene Algebra with Tests): expressiveness & analyzability.

Semantic Foundation

- Theorems
  - **Soundness**: programs related by the axioms are equivalent
  - **Completeness**: equivalent programs are related by the axioms
  - **Decidability**: there is an algorithm for deciding equivalence (PSPACE-complete)

Syntax

$$
\begin{array}{rrll}
\text{Fields} & f ::= & f_1 \mid \cdots \mid f_k & \\
\text{Packets} & pk ::= & \{f_1 = v_1, \cdots, f_k = v_k\} & \\
\text{Histories} & h ::= & pk{::}\langle\rangle \mid pk{::}h & \\
\text{Predicates} & a, b ::= & 1 & \textit{Identity} \\
& \mid & 0 & \textit{Drop} \\
& \mid & f = n & \textit{Test} \\
& \mid & a + b & \textit{Disjunction} \\
& \mid & a \cdot b & \textit{Conjunction} \\
& \mid & \neg a & \textit{Negation} \\
\text{Policies} & p, q ::= & a & \textit{Filter} \\
& \mid & f \leftarrow n & \textit{Modification} \\
& \mid & p + q & \textit{Union} \\
& \mid & p \cdot q & \textit{Sequential composition} \\
& \mid & p^* & \textit{Kleene star} \\
& \mid & \mathsf{dup} & \textit{Duplication}
\end{array}
$$

Semantics

$$
\begin{aligned}
[\![p]\!] &\in \mathsf{H} \to \mathcal{P}(\mathsf{H}) \\
[\![1]\!]\ h &\triangleq \{h\} \\
[\![0]\!]\ h &\triangleq \{\} \\
[\![f = n]\!]\ (pk{::}h) &\triangleq \begin{cases} \{pk{::}h\} & \text{if } pk.f = n \\ \{\} & \text{otherwise} \end{cases} \\
[\![\neg a]\!]\ h &\triangleq \{h\} \setminus ([\![a]\!]\ h) \\
[\![f \leftarrow n]\!]\ (pk{::}h) &\triangleq \{pk[f := n]{::}h\} \\
[\![p + q]\!]\ h &\triangleq [\![p]\!]\ h \cup [\![q]\!]\ h \\
[\![p \cdot q]\!]\ h &\triangleq ([\![p]\!] \bullet [\![q]\!])\ h \\
[\![p^*]\!]\ h &\triangleq \textstyle\bigcup_{i \in \mathbb{N}} F^i\ h \\
\text{where } F^0\ h \triangleq \{h\} &\text{ and } F^{i+1}\ h \triangleq ([\![p]\!] \bullet F^i)\ h \\
[\![\mathsf{dup}]\!]\ (pk{::}h) &\triangleq \{pk{::}(pk{::}h)\}
\end{aligned}
$$

Kleene Algebra Axioms

$p + (q + r) \equiv (p + q) + r$	KA-PLUS-ASSOC
$p + q \equiv q + p$	KA-PLUS-COMM
$p + 0 \equiv p$	KA-PLUS-ZERO
$p + p \equiv p$	KA-PLUS-IDEM
$p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$	KA-SEQ-ASSOC
$1 \cdot p \equiv p$	KA-ONE-SEQ
$p \cdot 1 \equiv p$	KA-SEQ-ONE
$p \cdot (q + r) \equiv p \cdot q + p \cdot r$	KA-SEQ-DIST-L
$(p + q) \cdot r \equiv p \cdot r + q \cdot r$	KA-SEQ-DIST-R
$0 \cdot p \equiv 0$	KA-ZERO-SEQ
$p \cdot 0 \equiv 0$	KA-SEQ-ZERO
$1 + p \cdot p^* \equiv p^*$	KA-UNROLL-L
$q + p \cdot r \leq r \Rightarrow p^* \cdot q \leq r$	KA-LFP-L
$1 + p^* \cdot p \equiv p^*$	KA-UNROLL-R
$p + q \cdot r \leq q \Rightarrow p \cdot r^* \leq q$	KA-LFP-R

Additional Boolean Algebra Axioms

$a + (b \cdot c) \equiv (a + b) \cdot (a + c)$	BA-PLUS-DIST
$a + 1 \equiv 1$	BA-PLUS-ONE
$a + \neg a \equiv 1$	BA-EXCL-MID
$a \cdot b \equiv b \cdot a$	BA-SEQ-COMM
$a \cdot \neg a \equiv 0$	BA-CONTRA
$a \cdot a \equiv a$	BA-SEQ-IDEM

Packet Algebra Axioms

$f \leftarrow n \cdot f' \leftarrow n' \equiv f' \leftarrow n' \cdot f \leftarrow n$, if $f \neq f'$	PA-MOD-MOD-COMM
$f \leftarrow n \cdot f' = n' \equiv f' = n' \cdot f \leftarrow n$, if $f \neq f'$	PA-MOD-FILTER-COMM
$\mathsf{dup} \cdot f = n \equiv f = n \cdot \mathsf{dup}$	PA-DUP-FILTER-COMM
$f \leftarrow n \cdot f = n \equiv f \leftarrow n$	PA-MOD-FILTER
$f = n \cdot f \leftarrow n \equiv f = n$	PA-FILTER-MOD
$f \leftarrow n \cdot f \leftarrow n' \equiv f \leftarrow n'$	PA-MOD-MOD
$f = n \cdot f = n' \equiv 0$, if $n \neq n'$	PA-CONTRA
$\sum_i f = i \equiv 1$	PA-MATCH-ALL

Figure 2. NetKAT: syntax, semantics, and equational axioms.

Policy

- **Forwarding**: transfer packets between hosts,
- **Access control**: filter or block specific packets
- …

Topology

- Directed graph with hosts and switches as nodes and links as edges
- To model an internal link, use sequential composition of a filter and a modification.
- To model a link at the perimeter of the network, use filter that retains packets located at the ingress port.

Application: Reachability

- Input:
  - Ingress predicate i
  - Topology t
  - Switch program p
  - Egress predicate e
- Test:
  - **filter** i; **dup**; (p; **dup**; t)*; **filter** e ~= **filter** false

Application: Optimization

- "Will the network behave the same if I put the firewall rules on switch A, or on switch B?"
- Formally, does the following equivalence hold?
  - (**filter** sw=A; fw; routing) | (**filter** sw=B; routing)
  - (**filter** sw=A; routing) | (**filter** sw=B; fw; routing)

Application: Optimization

$in; (p_A; t)^*; p_A; out$
$\equiv \{$ definition in, out, and p_A $\}$
$s_A; \text{SSH}; ((s_A; \neg\text{SSH}; p + s_B; p); t)^*; p_A; s_B$
$\equiv \{$ KAT-Invariant $\}$
$s_A; \text{SSH}; ((s_A; \neg\text{SSH}; p + s_B; p); t; \text{SSH})^*; p_A; s_B$
$\equiv \{$ KA-Seq-Dist-R $\}$
$s_A; \text{SSH}; (s_A; \neg\text{SSH}; p; t; \text{SSH} + s_B; p; t; \text{SSH})^*; p_A; s_B$
$\equiv \{$ KAT-Commute $\}$
$s_A; \text{SSH}; (s_A; \neg\text{SSH}; \text{SSH}; p; t + s_B; p; t; \text{SSH})^*; p_A; s_B$
$\equiv \{$ BA-Contra $\}$
$s_A; \text{SSH}; (s_A; \text{drop}; p; t + s_B; p; t; \text{SSH})^*; p_A; s_B$
$\equiv \{$ KA-Seq-Zero, KA-Zero-Seq,KA-Plus-Comm, KA-Plus-Zero $\}$
$s_A; \text{SSH}; (s_B; p; t; \text{SSH})^*; p_A; s_B$
$\equiv \{$ KA-Unroll-L $\}$
$s_A; \text{SSH}; (\text{id} + (s_B; p; t; \text{SSH}); (s_B; p; t; \text{SSH})^*); p_A; s_B$
$\equiv \{$ KA-Seq-Dist-L and KA-Seq-Dist-R $\}$
$(s_A; \text{SSH}; p_A; s_B)+$
$(s_A; \text{SSH}; s_B; p; t; \text{SSH}; (s_B; p; t; \text{SSH})^*; p_A; s_B)$
$\equiv \{$ KAT-Commute $\}$
$(s_A; s_B; \text{SSH}; p_A)+$
$(s_A; s_B; \text{SSH}; p; t; \text{SSH}; (s_B; p; t; \text{SSH})^*; p_A; s_B)$
$\equiv \{$ PA-Contra $\}$
$(\text{drop}; \text{SSH}; p_A)+$
$(\text{drop}; \text{SSH}; p; t; \text{SSH}; (s_B; p; t; \text{SSH})^*; p_A; s_B)$
$\equiv \{$ KA-Zero-Seq, KA-Plus-Idem $\}$
drop
$\equiv \{$ KA-Seq-Zero, KA-Zero-Seq, KA-Plus-Idem $\}$
$s_A; (p_B; t)^*; (\text{SSH}; \text{drop}; p + s_B; \text{drop}; p; s_B)$
$\equiv \{$ PA-Contra and BA-Contra $\}$
$s_A; (p_B; t)^*; (\text{SSH}; s_A; s_B; p + s_B; \text{SSH}; \neg\text{SSH}; p; s_B)$
$\equiv \{$ KAT-Commute $\}$
$s_A; (p_B; t)^*; (\text{SSH}; s_A; p; s_B + \text{SSH}; s_B; \neg\text{SSH}; p; s_B)$
$\equiv \{$ KA-Seq-Dist-L and KA-Seq-Dist-R $\}$
$s_A; (p_B; t)^*; \text{SSH}; (s_A; p + s_B; \neg\text{SSH}; p); s_B$
$\equiv \{$ KAT-Commute $\}$
$s_A; \text{SSH}; (p_B; t)^*; (s_A; p + s_B; \neg\text{SSH}; p); s_B$
$\equiv \{$ definition in, out, and p_B $\}$
$in; (p_B; t)^*; p_B; out$

Conclusion

- A new semantic foundation for NPL based on Kleene Algebra with Tests (KAT)
- Formalize NetKAT with sound and complete equational axioms
- Applications in network reasoning about reachability, traffic isolation, etc.
- Further improvement opportunities:
    - Explore other semantic foundations
    - Non-deterministic NetKAT

Thank you & QA