Are All Firewall Systems Equally Powerful?

Lorenzo Ceragioli, **Pierpaolo Degano**
Dipartimento di Informatica — Università di Pisa

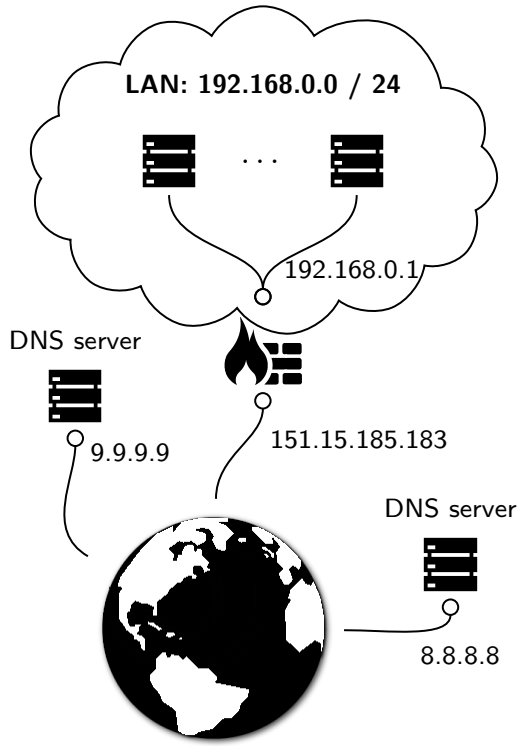
Letterio Galletta
IMT School for Advanced Studies, Lucca

Firewall – Network Access Control

Prevent illegit network traffic

For each packet

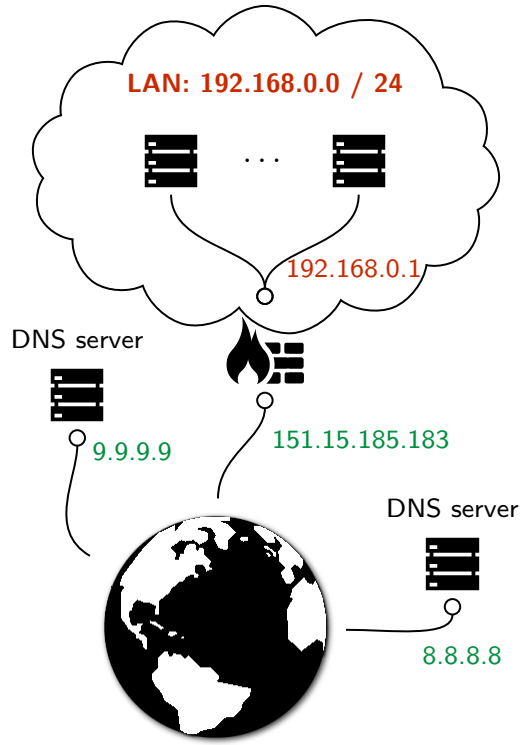
- **accepts or drops** it
- possibly **modifies** its source or destination (NAT)



Firewall — example

Packets flow

- freely among local nodes, e.g. between 192.168.0.3 and 192.168.0.23
- from local to external nodes, e.g. from 192.168.0.3 to 8.8.8.8, provided its source address is modified in the external one of the firewall 151.15.185.183 (SNAT)



(The firewall has *Self Addresses*
$\mathcal{S} = \{$`192.168.0.1`, `151.15.185.183`, `127.0.0.1`$\}$
for local, external and self reference)

Firewall Configuration Example (IPTABLES)

```
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]

-A PREROUTING -p udp --dport 123 -j DNAT --to 193.204.114.232
-A OUTPUT -p udp --dport 123 -j DNAT --to 193.204.114.232
-A PREROUTING -p tcp -d 151.15.185.183 --dport 80 -j DNAT --to 10.0.0.8
-A OUTPUT -p tcp -d 151.15.185.183 --dport 80 -j DNAT --to 10.0.0.8

-A POSTROUTING -d 192.168.0.0/16 -j ACCEPT
-A INPUT -d 192.168.0.0/16 -j ACCEPT
-A POSTROUTING -d 10.0.0.0/8 -j ACCEPT
-A INPUT -d 10.0.0.0/8 -j ACCEPT
-A POSTROUTING -j SNAT --to 151.15.185.183
-A INPUT -j SNAT --to 151.15.185.183

COMMIT

*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]

-A INPUT -m state --state ESTABLISHED -j ACCEPT
-A INPUT -p tcp -d 10.0.0.8 --dport 80 -j ACCEPT
-A INPUT -s 10.0.0.0/8 -d 10.0.0.0/8 -j ACCEPT
-A INPUT -s 192.168.0.0/16 ! -d 10.0.0.0/8 -j ACCEPT
-A INPUT -p udp -d 193.204.114.232 --dport 123 -j ACCEPT

-A FORWARD -m state --state ESTABLISHED -j ACCEPT
-A FORWARD -p tcp -d 10.0.0.8 --dport 80 -j ACCEPT
-A FORWARD -s 10.0.0.0/8 -d 10.0.0.0/8 -j ACCEPT
-A FORWARD -s 192.168.0.0/16 ! -d 10.0.0.0/8 -j ACCEPT
-A FORWARD -p udp -d 193.204.114.232 --dport 123 -j ACCEPT

-A OUTPUT -m state --state ESTABLISHED -j ACCEPT
-A OUTPUT -p tcp -d 10.0.0.8 --dport 80 -j ACCEPT
-A OUTPUT -s 10.0.0.0/8 -d 10.0.0.0/8 -j ACCEPT
-A OUTPUT -s 192.168.0.0/16 ! -d 10.0.0.0/8 -j ACCEPT
-A OUTPUT -p udp -d 193.204.114.232 --dport 123 -j ACCEPT

COMMIT
```

Firewall Configurations – a Mess

Decision of the firewall $\rightarrow$ based on the **configuration** (list of rules)

Difficult to read

- No **semantics** — just manuals
- Intricate **evaluation order**
- **Interaction** among rules (Shadowing)
- **Goto's** (and call-return)
- **OS dependent**
- Other **low level** details
- **Nonsense** like $\neg(p \vee q)$ meaning $\neg p \vee \neg q$

Difficult to manage

- Configuration
- Cross-system porting
- Test
- Verification

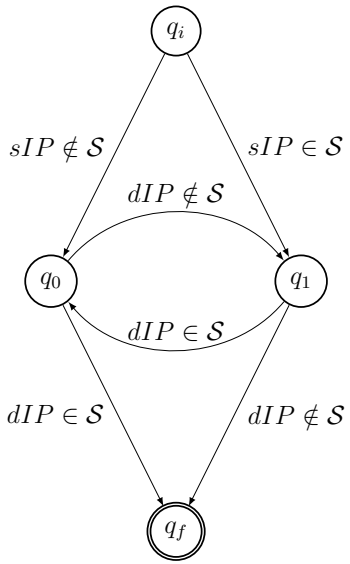
are **error-prone tasks**

Formalizing Firewall Configurations [EuroS&P, POST]

Firewall = evaluating procedure of the language + set of rules

Control Diagram

Accept a packet if it flows from q_i to q_f visiting each node at most once



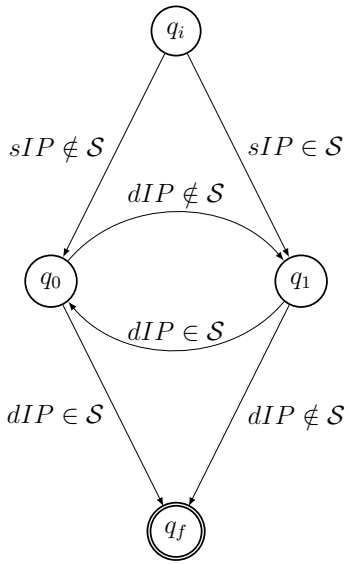
$\mathcal{S}$ are the addresses of the firewall

Formalizing Firewall Configurations [EuroS&P, POST]

Firewall = evaluating procedure of the language + set of rules

Control Diagram

Accept a packet if it flows from q_i to q_f visiting each node at most once



$\mathcal{S}$ are the addresses of the firewall

Configuration

Assigns a ruleset R to each node

Ruleset : list of **rules** $r = (\phi, a)$

- $\phi(p)$: **condition** e.g. $dport = 80$ (HTTP)
- a : **action**
  - `ACCEPT`
  - `DROP`
  - `NAT`(d_n, s_n)
  - `GOTO(R)`
  - `CALL(R)`
  - `RETURN`

With that in mind ...

Transcompilation pipeline between firewall languages

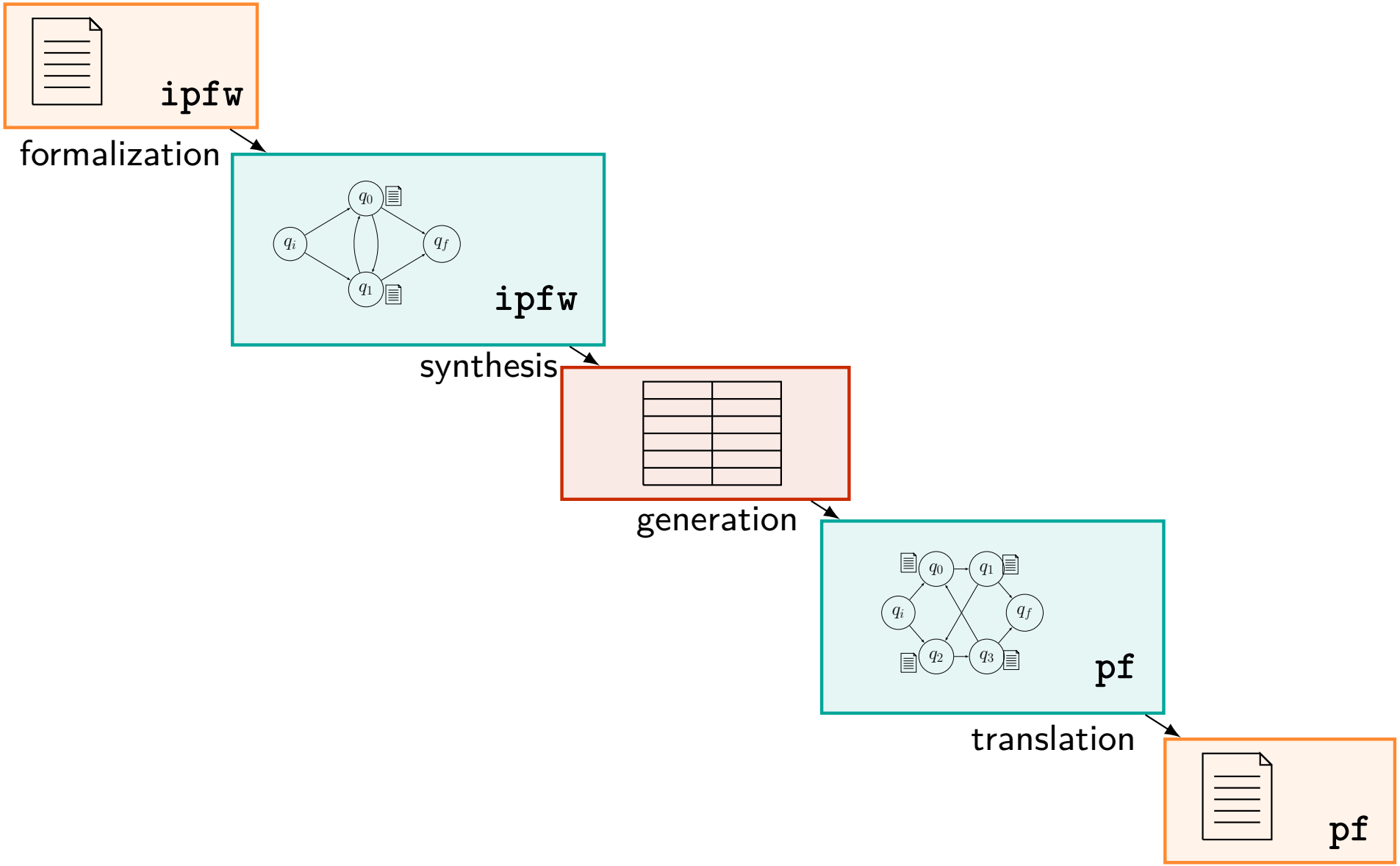
1. Decompile a configuration c from the source language to **Intermediate Firewall Configuration Language (IFCL)**
2. **Extract the meaning** of the policy as a function f describing how the accepted packets are translated $\Leftarrow$ SEMANTICS $(\!|\ |\!)$
3. **Compile** the function $f = (\!|c|\!)$ into the target language

Supports `iptables`, `pf`, `ipfw` and (partially) `CISCO-`*ios*

Helps

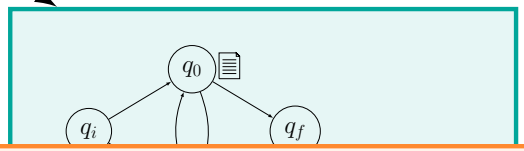
- **porting** configurations from a system to another
- **verifying** properties
- **updating** configurations
- **refactoring** configurations

Transcompilation Pipeline: *FireWallSynthesizer*



Transcompilation Pipeline: *FireWallSynthesizer*

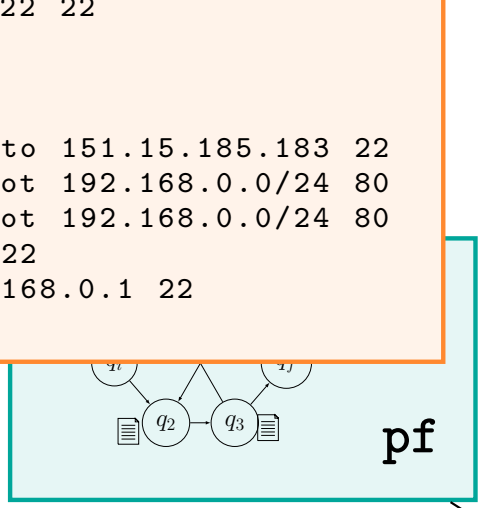
formalization



ipfw source configuration

```
ipfw -q nat 1 config redirect_port tcp 192.168.0.8:22 22
ipfw -q nat 2 config ip 151.15.185.183

ipfw -q add 0010 deny tcp from any to 8.8.8.8
ipfw -q add 0020 nat 1 tcp from not 192.168.0.0/24 to 151.15.185.183 22
ipfw -q add 0030 nat 2 tcp from 192.168.0.0/24 to not 192.168.0.0/24 80
ipfw -q add 0040 allow tcp from 151.15.185.183 to not 192.168.0.0/24 80
ipfw -q add 0050 allow tcp from any to 192.168.0.8 22
ipfw -q add 0060 allow tcp from 192.168.0.8 to 192.168.0.1 22
ipfw -q add 0070 deny all from any to any
```



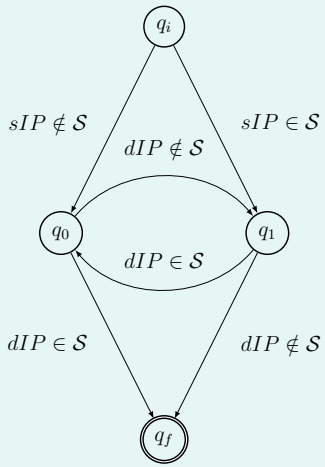
translation

Transcompilation Pipeline: *FireWallSynthesizer*

formalizati

IFCL source configuration

```
R(q_0) = R(q_1):
    (dstIP = 8.8.8.8, DROP);
    (srcIP != 192.168.0.0/24 and dstIP = 151.15.185.183 and
        dstPort = 22, NAT(192.168.0.8, *));
    (srcIP = 192.168.0.0/24 and dstIP != 192.168.0.0/24 and
        dstPort = 80, NAT(*, 151.15.185.183));
    (srcIP = 151.15.185.183 and dstIP != 192.168.0.0/24 and
        dstPort = 80, ACCEPT);
    (dstIP = 192.168.0.8 and dstPort = 22, ACCEPT);
    (srcIP = 192.168.0.8 and dstIP = 192.168.0.1 and
        dstPort = 22, ACCEPT);
    (true, DROP);
```



pf

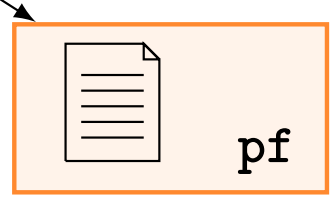
Transcompilation Pipeline: *FireWallSynthesizer*

formalization

Table representing the *accepted* packets and their transformations

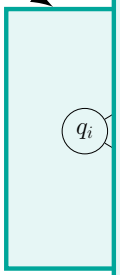
Received packets				Accepted packets			
sIP	**sPort**	**dIP**	**dPort**	**sIP**	**sPort**	**dIP**	**dPort**
192.168.0.8	*	192.168.0.1	22	-	-	-	-
*	*	192.168.0.8	22	-	-	-	-
151.15.185.183	*	* \{ 8.8.8.8 192.168.0.0/24 }	80	-	-	-	-
192.168.0.0/24	*	* \{ 8.8.8.8 192.168.0.0/24 }	80	151.15.185.183	-	-	-
* \{ 192.168.0.0/24 }	*	151.15.185.183	22	-	-	192.168.0.8	-

translation



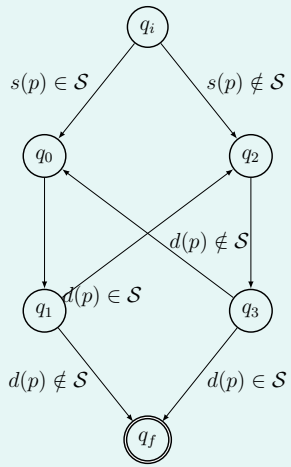
Transcompilation Pipeline: *FireWallSynthesizer*

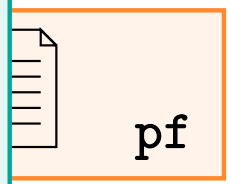
formalization



IFCL target configuration

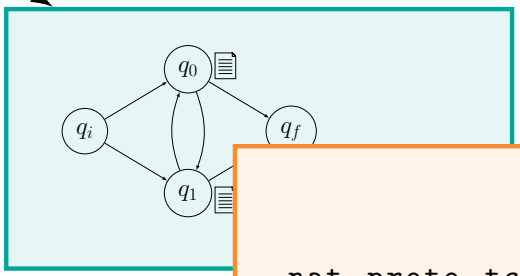
```
R(q_0):
    (srcIP = 192.168.0.0/24 and dstIP != 192.168.0.0/24 and
        dstPort = 80, NAT(*, 151.15.185.183));
    (true, ACCEPT);
R(q_2):
    (srcIP != 192.168.0.0/24 and dstIP = 151.15.185.183 and
        dstPort = 22, NAT(192.168.0.8, *));
    (true, ACCEPT);
R(q_1) = R(q_3):
    (dstIP = 8.8.8.8, DROP);
    (srcIP = 151.15.185.183 and dstIP != 192.168.0.0/24 and
        dstPort = 80, ACCEPT);
    (dstIP = 192.168.0.8 and dstPort = 22, ACCEPT);
    (srcIP = 192.168.0.8 and dstIP = 192.168.0.1 and
        dstPort = 22, ACCEPT);
    (true, DROP);
```





Transcompilation Pipeline: *FireWallSynthesizer*

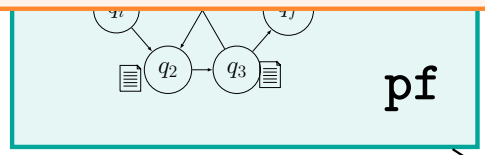
formalization



pf target configuration

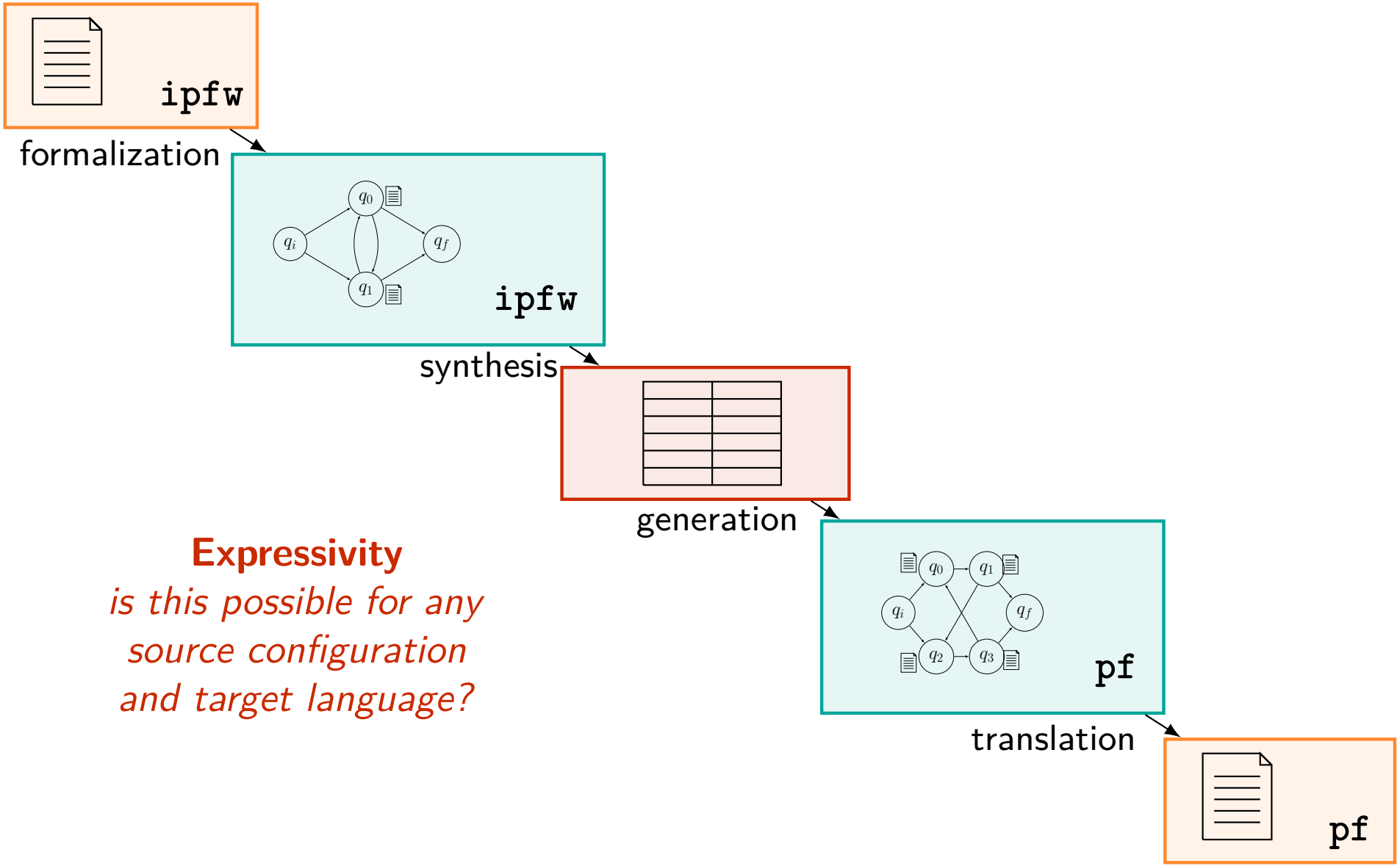
```
nat proto tcp from 192.168.0.0/24 to ! 192.168.0.0/24 port 80
                                          -> 151.15.185.183
rdr proto tcp from ! 192.168.0.0/24 to 151.15.185.183 port 22
                                          -> 192.168.0.8

block all
pass from 192.168.0.8 to 192.168.0.1 port 22
pass from any to 192.168.0.8 port 22
pass from 151.15.185.183 to ! 192.168.0.0/24 port 80
block from any to 8.8.8.8
```



translation

Transcompilation Pipeline: *FireWallSynthesizer*



Expressivity

is this possible for any source configuration and target language?

Checking expressivity of firewall languages

A **general approach** that

- works for **any firewall language**
- detects **corner cases** and idiosyncrasies
- helps in designing **automatic tools for generating configurations**

Pair Expressivity of firewall language $\mathcal{L}$

$\mathbb{P}$ set of **packets** $p = (dstIP : dstPort,\ srcIP : srcPort)$

$\mathcal{T}_{\mathbb{P}}$ set of **transformations** t

$$p_1 = (192.168.0.1 : 1,\ 192.168.0.1 : 1)$$
$$t_1 = (\lambda_{1.1.1.1} : id,\ id : id)$$
$$t_1(p_1) = (1.1.1.1 : 1,\ 192.168.0.1 : 1)$$

Pair Expressivity

Given a packet p and a transformation t
does it exist a configuration in $\mathcal{L}$ that associates p with t?

Key observation

Only IFCL configurations obtainable **from a source configuration**,

... computed **directly on** the control diagram!

Legal IFCL configurations

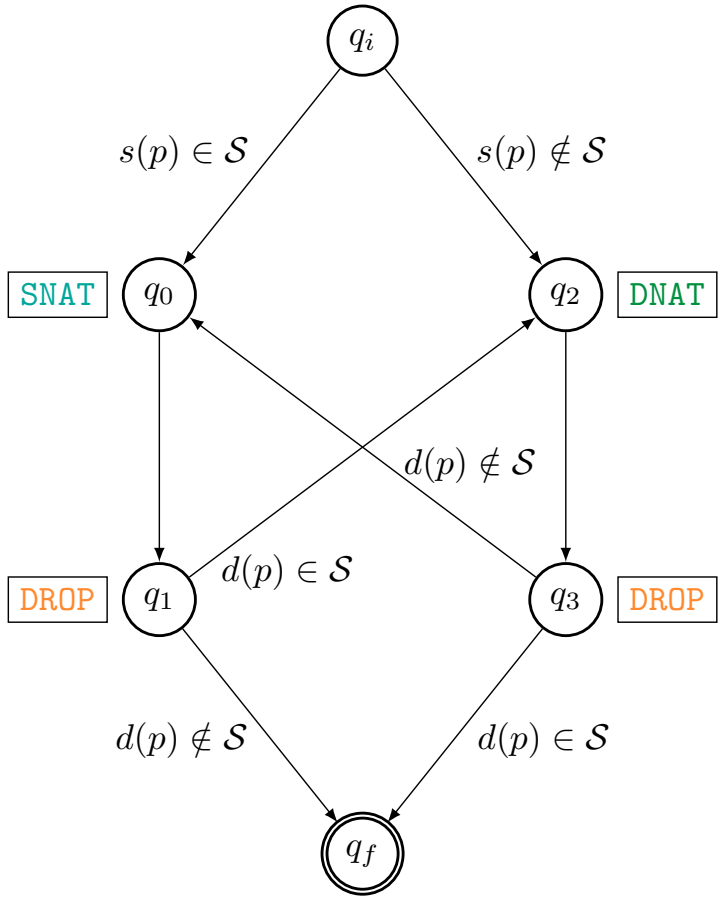
Not every ruleset can be assigned to each node!

Assign ***cap-labels*** to nodes

- `DROP` :
  can **discard** the packet
- `SNAT` :
  can **change the source** address
- `DNAT` :
  can **change the destination** address

We restrict to cap-labels **compliant** configurations

The case of `pf`



$\mathcal{S}$ are the addresses of the firewall

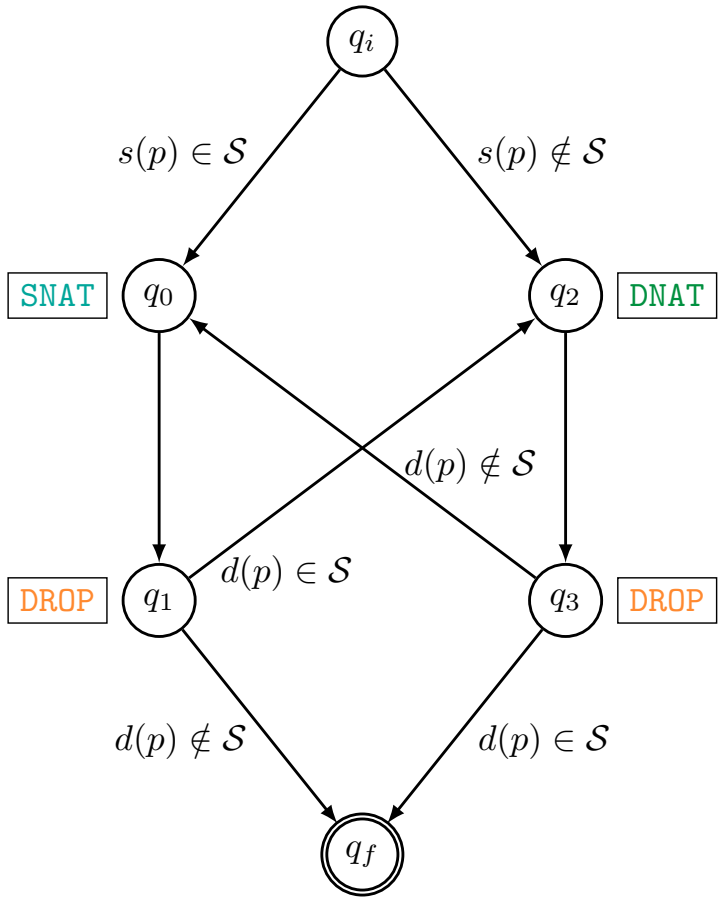
Enumerating expressible pairs (p, t) — $O(N^2)$: quotienting

Given a control diagram with labels

Returns all the expressible pairs (p, t)

- Take two subsets of **arcs predicates** X_1, X_2 –
  $X_1 = \{\ d(p) \in \mathcal{S},\ s(p) \in \mathcal{S}\ \}$
  $X_2 = \{\ d(p) \notin \mathcal{S},\ s(p) \in \mathcal{S}\ \}$
- Take a subset of **transformations** Y –
  $Y = \Lambda \times \Lambda$ (change source and destination)
- Take **a pair** (p, t) such that
  p satisfy X_1 – self source and destination
  t is inside Y – SNAT and DNAT
  $t(p)$ satisfy X_2 – not self destination

  E.g. $p = (\texttt{192.168.0.1} : 1,\ \texttt{192.168.0.1} : 1)$
  $t = (\lambda_{\texttt{1.1.1.1}} : id,\ \lambda_{\texttt{151.15.185.183}} : id)$
- **Check** if (p, t) is expressible – No!
  Then **every pair** for X_1, Y, X_2 is not



$\mathcal{S}$ are the addresses of the firewall
$\{\texttt{192.168.0.1}, \texttt{151.15.185.183}, \texttt{127.0.0.1}\}$

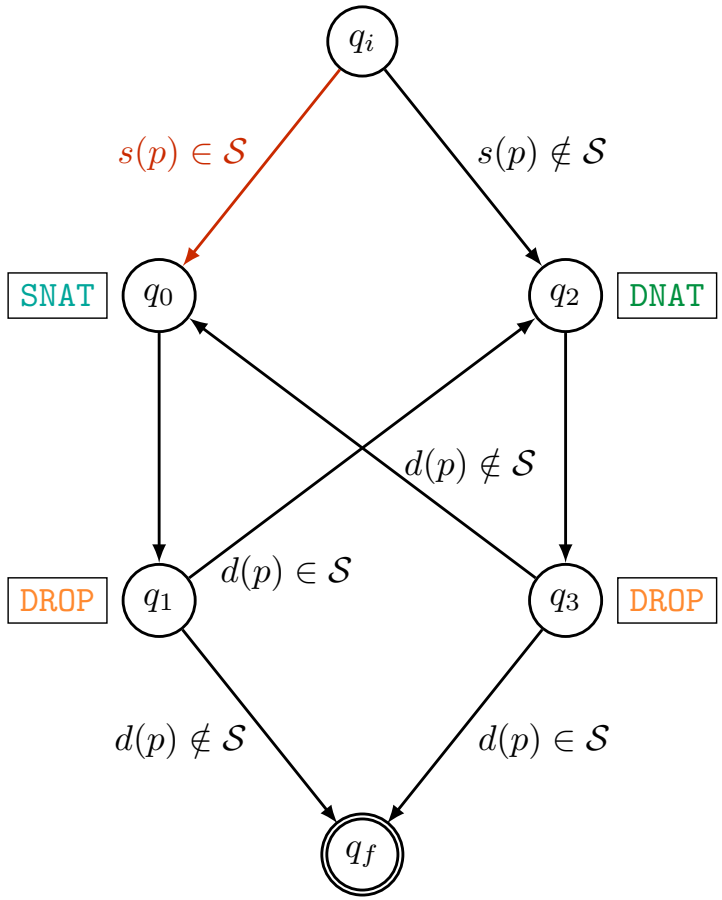
Enumerating expressible pairs (p, t) — $O(N^2)$: quotienting

Given a control diagram with labels

Returns all the expressible pairs (p, t)

- Take two subsets of **arcs predicates** X_1, X_2 –
  $X_1 = \{\ d(p) \in \mathcal{S},\ s(p) \in \mathcal{S}\ \}$
  $X_2 = \{\ d(p) \notin \mathcal{S},\ s(p) \in \mathcal{S}\ \}$
- Take a subset of **transformations** Y –
  $Y = \Lambda \times \Lambda$ (change source and destination)
- Take **a pair** (p, t) such that
  p satisfy X_1 – self source and destination
  t is inside Y – SNAT and DNAT
  $t(p)$ satisfy X_2 – not self destination

  E.g. $p = (\texttt{192.168.0.1} : 1,\ \texttt{192.168.0.1} : 1)$
  $t = (\lambda_{\texttt{1.1.1.1}} : id,\ \lambda_{\texttt{151.15.185.183}} : id)$
- **Check** if (p, t) is expressible – No!
  Then **every pair** for X_1, Y, X_2 is not



$\mathcal{S}$ are the addresses of the firewall
$\{\texttt{192.168.0.1}, \texttt{151.15.185.183}, \texttt{127.0.0.1}\}$

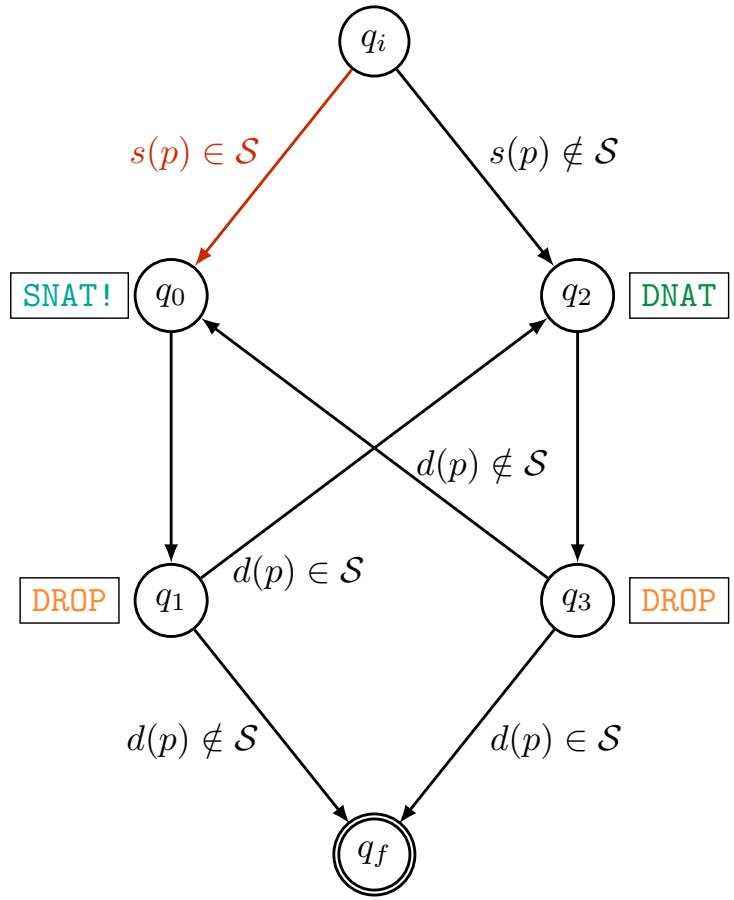
Enumerating expressible pairs (p, t) — $O(N^2)$: quotienting

Given a control diagram with labels

Returns all the expressible pairs (p, t)

- Take two subsets of **arcs predicates** X_1, X_2 –
  $X_1 = \{\ d(p) \in \mathcal{S},\ s(p) \in \mathcal{S}\ \}$
  $X_2 = \{\ d(p) \notin \mathcal{S},\ s(p) \in \mathcal{S}\ \}$
- Take a subset of **transformations** Y –
  $Y = \Lambda \times \Lambda$ (change source and destination)
- Take **a pair** (p, t) such that
  p satisfy X_1 – self source and destination
  t is inside Y – SNAT and DNAT
  $t(p)$ satisfy X_2 – not self destination

  E.g. $p = (\texttt{192.168.0.1} : 1,\ \texttt{192.168.0.1} : 1)$
  $t = (\lambda_{\texttt{1.1.1.1}} : id,\ \lambda_{\texttt{151.15.185.183}} : id)$
- **Check** if (p, t) is expressible – No!
  Then **every pair** for X_1, Y, X_2 is not



$\mathcal{S}$ are the addresses of the firewall
$\{\texttt{192.168.0.1}, \texttt{151.15.185.183}, \texttt{127.0.0.1}\}$

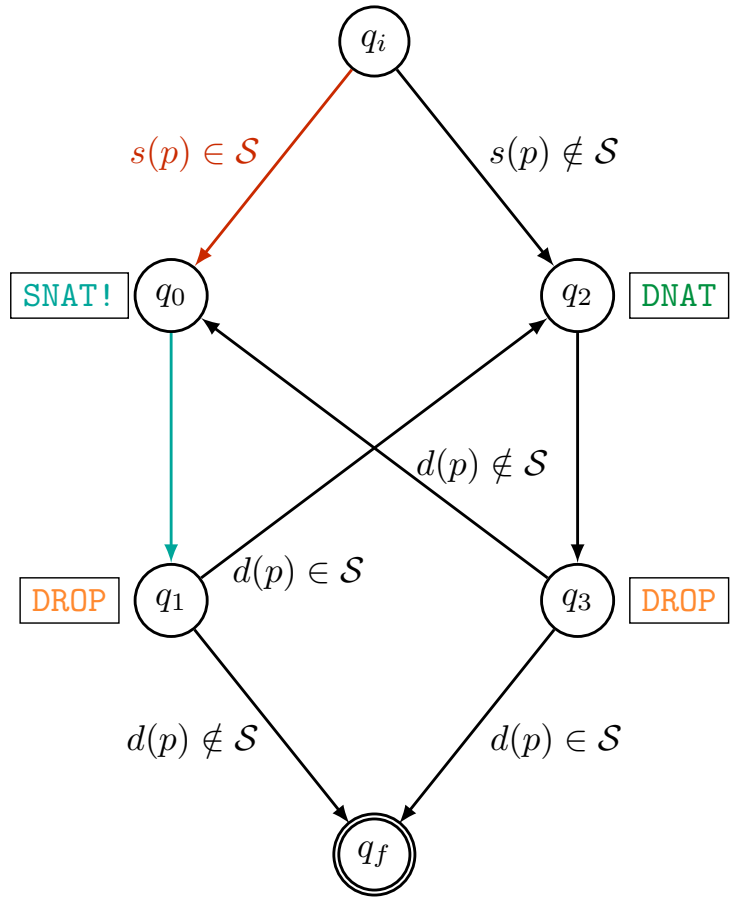
Enumerating expressible pairs (p, t) — $O(N^2)$: quotienting

Given a control diagram with labels

Returns all the expressible pairs (p, t)

- Take two subsets of **arcs predicates** X_1, X_2 –
  $X_1 = \{\ d(p) \in \mathcal{S},\ s(p) \in \mathcal{S}\ \}$
  $X_2 = \{\ d(p) \notin \mathcal{S},\ s(p) \in \mathcal{S}\ \}$
- Take a subset of **transformations** Y –
  $Y = \Lambda \times \Lambda$ (change source and destination)
- Take **a pair** (p, t) such that
  p satisfy X_1 – self source and destination
  t is inside Y – SNAT and DNAT
  $t(p)$ satisfy X_2 – not self destination

  E.g. $p = (\texttt{192.168.0.1} : 1,\ \texttt{192.168.0.1} : 1)$
  $t = (\lambda_{\texttt{1.1.1.1}} : id,\ \lambda_{\texttt{151.15.185.183}} : id)$
- **Check** if (p, t) is expressible – No!
  Then **every pair** for X_1, Y, X_2 is not



$\mathcal{S}$ are the addresses of the firewall
$\{\texttt{192.168.0.1}, \texttt{151.15.185.183}, \texttt{127.0.0.1}\}$

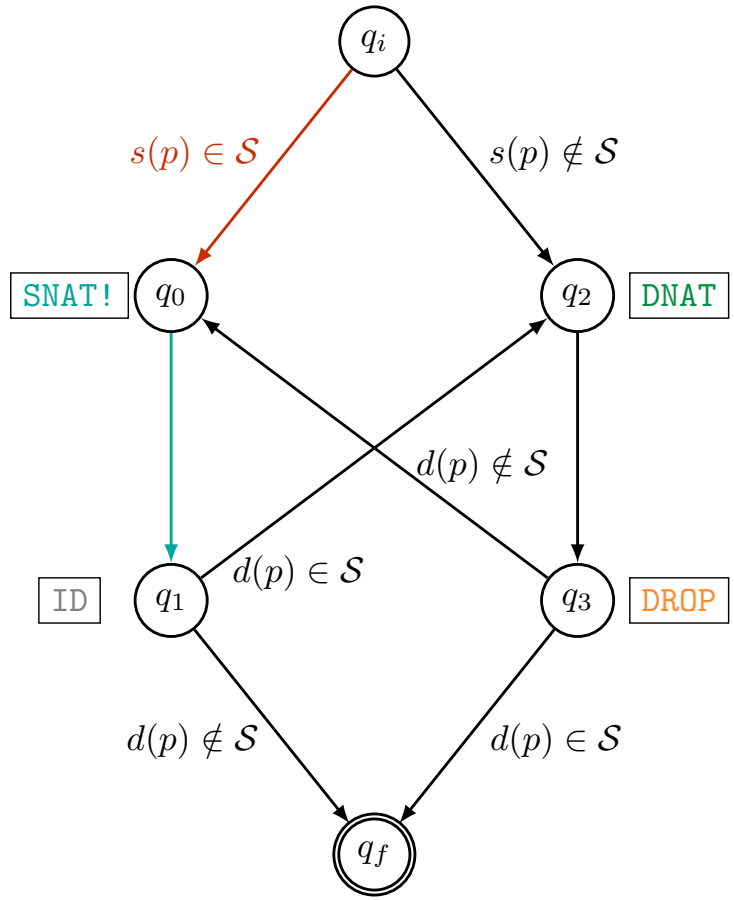
Enumerating expressible pairs (p, t) — $O(N^2)$: quotienting

Given a control diagram with labels

Returns all the expressible pairs (p, t)

- Take two subsets of **arcs predicates** X_1, X_2 –
  $X_1 = \{\ d(p) \in \mathcal{S},\ s(p) \in \mathcal{S}\ \}$
  $X_2 = \{\ d(p) \notin \mathcal{S},\ s(p) \in \mathcal{S}\ \}$
- Take a subset of **transformations** Y –
  $Y = \Lambda \times \Lambda$ (change source and destination)
- Take **a pair** (p, t) such that
  p satisfy X_1 – self source and destination
  t is inside Y – SNAT and DNAT
  $t(p)$ satisfy X_2 – not self destination

  E.g. $p = (\texttt{192.168.0.1} : 1,\ \texttt{192.168.0.1} : 1)$
  $t = (\lambda_{\texttt{1.1.1.1}} : id,\ \lambda_{\texttt{151.15.185.183}} : id)$
- **Check** if (p, t) is expressible – No!
  Then **every pair** for X_1, Y, X_2 is not



$\mathcal{S}$ are the addresses of the firewall
$\{\texttt{192.168.0.1}, \texttt{151.15.185.183}, \texttt{127.0.0.1}\}$

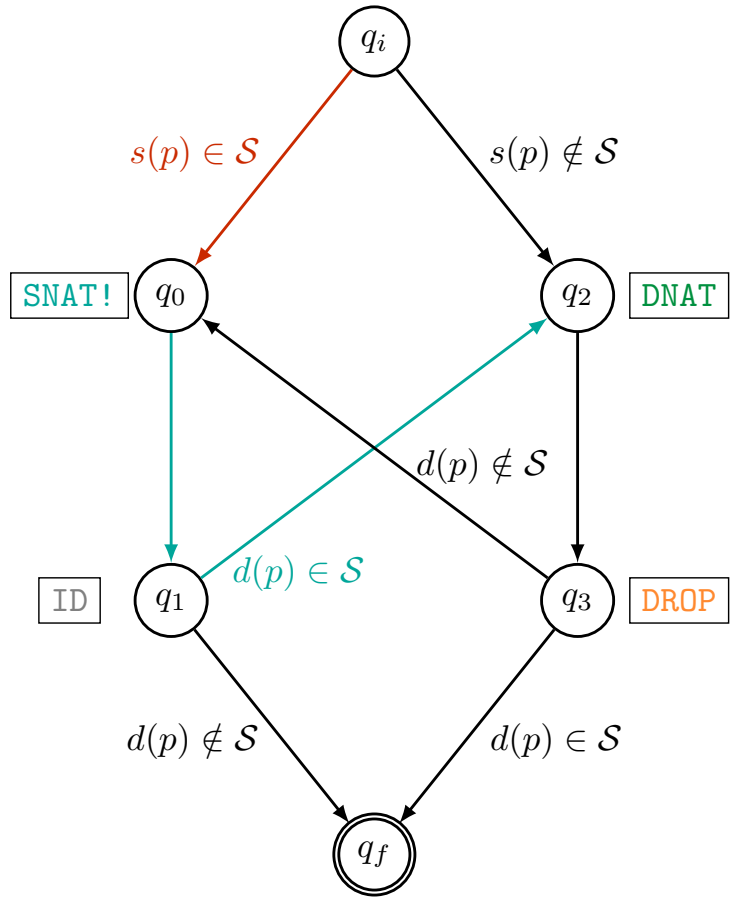
Enumerating expressible pairs (p, t) — $O(N^2)$: quotienting

Given a control diagram with labels

Returns all the expressible pairs (p, t)

- Take two subsets of **arcs predicates** X_1, X_2 –
  $X_1 = \{\ d(p) \in \mathcal{S},\ s(p) \in \mathcal{S}\ \}$
  $X_2 = \{\ d(p) \notin \mathcal{S},\ s(p) \in \mathcal{S}\ \}$
- Take a subset of **transformations** Y –
  $Y = \Lambda \times \Lambda$ (change source and destination)
- Take **a pair** (p, t) such that
  p satisfy X_1 – self source and destination
  t is inside Y – SNAT and DNAT
  $t(p)$ satisfy X_2 – not self destination

  E.g. $p = (\texttt{192.168.0.1} : 1,\ \texttt{192.168.0.1} : 1)$
  $t = (\lambda_{\texttt{1.1.1.1}} : id,\ \lambda_{\texttt{151.15.185.183}} : id)$
- **Check** if (p, t) is expressible – No!
  Then **every pair** for X_1, Y, X_2 is not



$\mathcal{S}$ are the addresses of the firewall
$\{$192.168.0.1, 151.15.185.183, 127.0.0.1$\}$

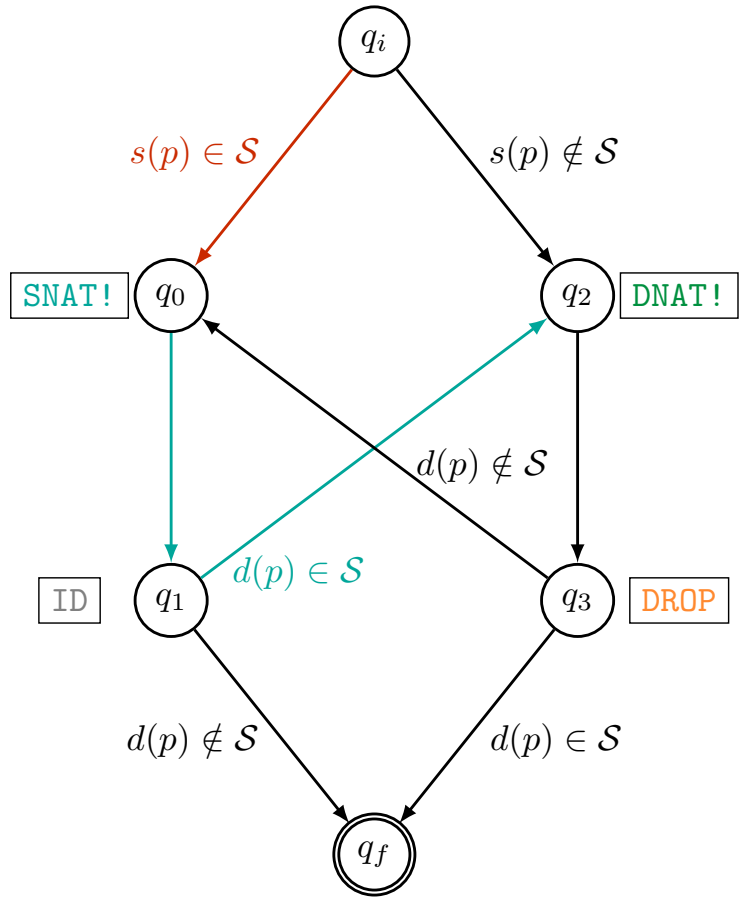
Enumerating expressible pairs (p, t) — $O(N^2)$: quotienting

Given a control diagram with labels

Returns all the expressible pairs (p, t)

- Take two subsets of **arcs predicates** X_1, X_2 –
  $X_1 = \{\ d(p) \in \mathcal{S},\ s(p) \in \mathcal{S}\ \}$
  $X_2 = \{\ d(p) \notin \mathcal{S},\ s(p) \in \mathcal{S}\ \}$
- Take a subset of **transformations** Y –
  $Y = \Lambda \times \Lambda$ (change source and destination)
- Take **a pair** (p, t) such that
  p satisfy X_1 – self source and destination
  t is inside Y – SNAT and DNAT
  $t(p)$ satisfy X_2 – not self destination

  E.g. $p = (\texttt{192.168.0.1} : 1,\ \texttt{192.168.0.1} : 1)$
  $t = (\lambda_{\texttt{1.1.1.1}} : id,\ \lambda_{\texttt{151.15.185.183}} : id)$
- **Check** if (p, t) is expressible – No!
  Then **every pair** for X_1, Y, X_2 is not



$\mathcal{S}$ are the addresses of the firewall
$\{$192.168.0.1, 151.15.185.183, 127.0.0.1$\}$

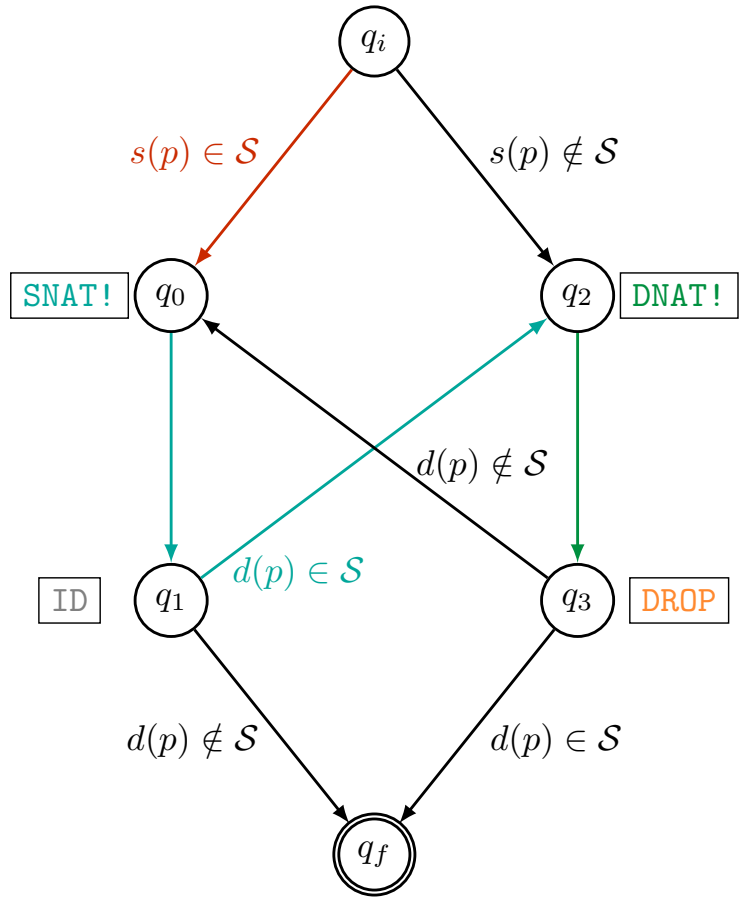
Enumerating expressible pairs (p, t) — $O(N^2)$: quotienting

Given a control diagram with labels

Returns all the expressible pairs (p, t)

- Take two subsets of **arcs predicates** X_1, X_2 –
  $X_1 = \{\ d(p) \in \mathcal{S},\ s(p) \in \mathcal{S}\ \}$
  $X_2 = \{\ d(p) \notin \mathcal{S},\ s(p) \in \mathcal{S}\ \}$
- Take a subset of **transformations** Y –
  $Y = \Lambda \times \Lambda$ (change source and destination)
- Take **a pair** (p, t) such that
  p satisfy X_1 – self source and destination
  t is inside Y – SNAT and DNAT
  $t(p)$ satisfy X_2 – not self destination

  E.g. $p = (\texttt{192.168.0.1} : 1,\ \texttt{192.168.0.1} : 1)$
  $t = (\lambda_{\texttt{1.1.1.1}} : id,\ \lambda_{\texttt{151.15.185.183}} : id)$
- **Check** if (p, t) is expressible – No!
  Then **every pair** for X_1, Y, X_2 is not



$\mathcal{S}$ are the addresses of the firewall
$\{$192.168.0.1, 151.15.185.183, 127.0.0.1$\}$

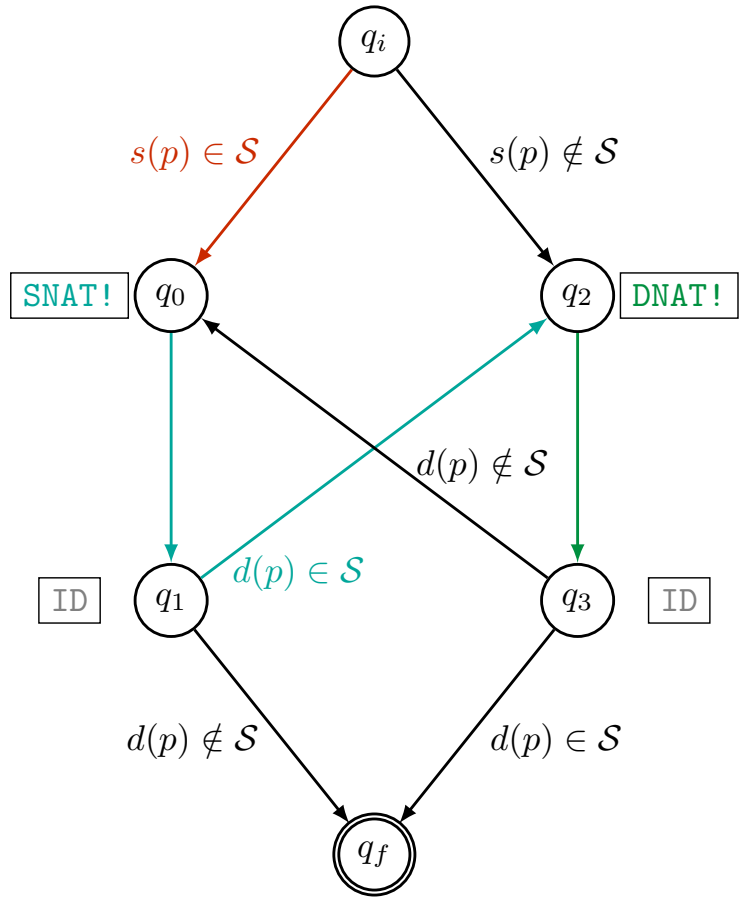
Enumerating expressible pairs (p, t) — $O(N^2)$: quotienting

Given a control diagram with labels

Returns all the expressible pairs (p, t)

- Take two subsets of **arcs predicates** X_1, X_2 –
  $X_1 = \{\ d(p) \in \mathcal{S},\ s(p) \in \mathcal{S}\ \}$
  $X_2 = \{\ d(p) \notin \mathcal{S},\ s(p) \in \mathcal{S}\ \}$
- Take a subset of **transformations** Y –
  $Y = \Lambda \times \Lambda$ (change source and destination)
- Take **a pair** (p, t) such that
  p satisfy X_1 – self source and destination
  t is inside Y – SNAT and DNAT
  $t(p)$ satisfy X_2 – not self destination

  E.g. $p = (\texttt{192.168.0.1} : 1,\ \texttt{192.168.0.1} : 1)$
  $t = (\lambda_{\texttt{1.1.1.1}} : id,\ \lambda_{\texttt{151.15.185.183}} : id)$
- **Check** if (p, t) is expressible – No!
  Then **every pair** for X_1, Y, X_2 is not



$\mathcal{S}$ are the addresses of the firewall
$\{\texttt{192.168.0.1}, \texttt{151.15.185.183}, \texttt{127.0.0.1}\}$

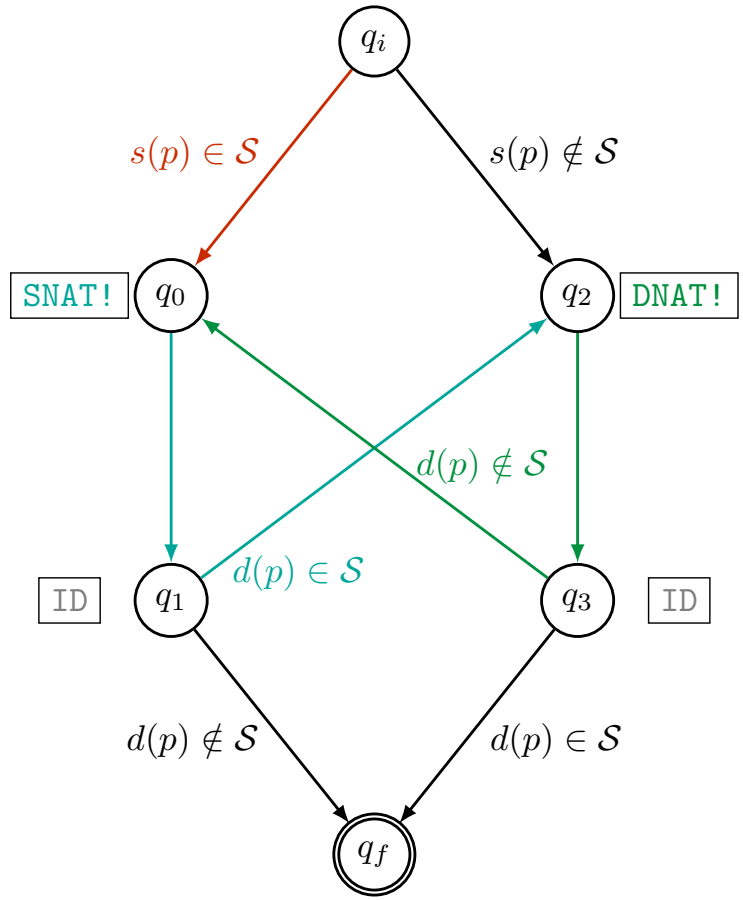
Enumerating expressible pairs (p, t) — $O(N^2)$: quotienting

Given a control diagram with labels

Returns all the expressible pairs (p, t)

- Take two subsets of **arcs predicates** X_1, X_2 –
  $X_1 = \{\ d(p) \in \mathcal{S},\ s(p) \in \mathcal{S}\ \}$
  $X_2 = \{\ d(p) \notin \mathcal{S},\ s(p) \in \mathcal{S}\ \}$
- Take a subset of **transformations** Y –
  $Y = \Lambda \times \Lambda$ (change source and destination)
- Take **a pair** (p, t) such that
  p satisfy X_1 – self source and destination
  t is inside Y – SNAT and DNAT
  $t(p)$ satisfy X_2 – not self destination

  E.g. $p = (\texttt{192.168.0.1} : 1,\ \texttt{192.168.0.1} : 1)$
  $t = (\lambda_{\texttt{1.1.1.1}} : id,\ \lambda_{\texttt{151.15.185.183}} : id)$
- **Check** if (p, t) is expressible – No!
  Then **every pair** for X_1, Y, X_2 is not



$\mathcal{S}$ are the addresses of the firewall
$\{\texttt{192.168.0.1}, \texttt{151.15.185.183}, \texttt{127.0.0.1}\}$

Results

X_1		Y	X_2		(p,t)	$E_{\mathcal{L}}$	
$d(p)$	$s(p)$		$d(t(p))$	$s(t(p))$	$(\,(d(p), s(p)),\ t\,)$	`pf/ipfw`	`iptables`
$\notin \mathcal{S}$	$\in \mathcal{S}$	$\varepsilon(\texttt{DNAT})$	$\in \mathcal{S}$	$\in \mathcal{S}$	$((b:r, a:r), (\lambda_a:\lambda_r, id:id))$	✗	✓
$\notin \mathcal{S}$	$\in \mathcal{S}$	$\varepsilon(\texttt{DNAT})$	$\notin \mathcal{S}$	$\in \mathcal{S}$	$((b:r, a:r), (\lambda_b:\lambda_r, id:id))$	✗	✓
$\in \mathcal{S}$	$\notin \mathcal{S}$	$\varepsilon(\texttt{SNAT})$	$\in \mathcal{S}$	$\in \mathcal{S}$	$((a:r, b:r), (id:id, \lambda_a:\lambda_r))$	✗	✓
$\in \mathcal{S}$	$\notin \mathcal{S}$	$\varepsilon(\texttt{SNAT})$	$\in \mathcal{S}$	$\notin \mathcal{S}$	$((a:r, b:r), (id:id, \lambda_b:\lambda_r))$	✗	✓
$\in \mathcal{S}$	$\in \mathcal{S}$	$\Lambda \times \Lambda$	$\notin \mathcal{S}$	$\in \mathcal{S}$	$((a:r, a:r), (\lambda_b:\lambda_r, \lambda_a:\lambda_r))$	✗	✓
$\in \mathcal{S}$	$\in \mathcal{S}$	$\Lambda \times \Lambda$	$\notin \mathcal{S}$	$\notin \mathcal{S}$	$((a:r, a:r), (\lambda_b:\lambda_r, \lambda_b:\lambda_r))$	✗	✓
$\in \mathcal{S}$	$\notin \mathcal{S}$	$\Lambda \times \Lambda$	$\in \mathcal{S}$	$\in \mathcal{S}$	$((a:r, b:r), (\lambda_a:\lambda_r, \lambda_a:\lambda_r))$	✗	✓
$\in \mathcal{S}$	$\notin \mathcal{S}$	$\Lambda \times \Lambda$	$\in \mathcal{S}$	$\notin \mathcal{S}$	$((a:r, b:r), (\lambda_a:\lambda_r, \lambda_b:\lambda_r))$	✗	✓
$\notin \mathcal{S}$	$\in \mathcal{S}$	$\Lambda \times \Lambda$	$\in \mathcal{S}$	$\in \mathcal{S}$	$((b:r, a:r), (\lambda_a:\lambda_r, \lambda_a:\lambda_r))$	✗	✓
$\notin \mathcal{S}$	$\in \mathcal{S}$	$\Lambda \times \Lambda$	$\in \mathcal{S}$	$\notin \mathcal{S}$	$((b:r, a:r), (\lambda_a:\lambda_r, \lambda_b:\lambda_r))$	✗	✓
$\notin \mathcal{S}$	$\in \mathcal{S}$	$\Lambda \times \Lambda$	$\notin \mathcal{S}$	$\in \mathcal{S}$	$((b:r, a:r), (\lambda_b:\lambda_r, \lambda_a:\lambda_r))$	✗	✓
$\notin \mathcal{S}$	$\in \mathcal{S}$	$\Lambda \times \Lambda$	$\notin \mathcal{S}$	$\notin \mathcal{S}$	$((b:r, a:r), (\lambda_b:\lambda_r, \lambda_b:\lambda_r))$	✗	✓
$\notin \mathcal{S}$	$\notin \mathcal{S}$	$\Lambda \times \Lambda$	$\in \mathcal{S}$	$\in \mathcal{S}$	$((b:r, b:r), (\lambda_a:\lambda_r, \lambda_a:\lambda_r))$	✗	✓
$\notin \mathcal{S}$	$\notin \mathcal{S}$	$\Lambda \times \Lambda$	$\in \mathcal{S}$	$\notin \mathcal{S}$	$((b:r, b:r), (\lambda_a:\lambda_r, \lambda_b:\lambda_r))$	✗	✓
Otherwise					...	✓	✓

Results

X_1		Y	X_2		(p,t)	$E_{\mathcal{L}}$	
$d(p)$	$s(p)$		$d(t(p))$	$s(t(p))$	$(\,(d(p), s(p)),\ t\,)$	`pf/ipfw`	`iptables`
$\notin \mathcal{S}$	$\in \mathcal{S}$	$\varepsilon(\texttt{DNAT})$	$\in \mathcal{S}$	$\in \mathcal{S}$	$((b:r, a:r), (\lambda_a:\lambda_r, id:id))$	✗	✓
$\notin \mathcal{S}$	$\in \mathcal{S}$	$\varepsilon(\texttt{DNAT})$	$\notin \mathcal{S}$	$\in \mathcal{S}$	$((b:r, a:r), (\lambda_b:\lambda_r, id:id))$	✗	✓
$\in \mathcal{S}$	$\notin \mathcal{S}$	$\varepsilon(\texttt{SNAT})$	$\in \mathcal{S}$	$\in \mathcal{S}$	$((a:r, b:r), (id:id, \lambda_a:\lambda_r))$	✗	✓
$\in \mathcal{S}$	$\notin \mathcal{S}$	$\varepsilon(\texttt{SNAT})$	$\in \mathcal{S}$	$\notin \mathcal{S}$	$((a:r, b:r), (id:id, \lambda_b:\lambda_r))$	✗	✓
$\in \mathcal{S}$	$\in \mathcal{S}$	$\Lambda \times \Lambda$	$\notin \mathcal{S}$	$\in \mathcal{S}$	$((a:r, a:r), (\lambda_b:\lambda_r, \lambda_a:\lambda_r))$	✗	✓
$\in \mathcal{S}$	$\in \mathcal{S}$	$\Lambda \times \Lambda$	$\notin \mathcal{S}$	$\notin \mathcal{S}$	$((a:r, a:r), (\lambda_b:\lambda_r, \lambda_b:\lambda_r))$	✗	✓
$\in \mathcal{S}$	$\notin \mathcal{S}$	$\Lambda \times \Lambda$	$\in \mathcal{S}$	$\in \mathcal{S}$	$((a:r, b:r), (\lambda_a:\lambda_r, \lambda_a:\lambda_r))$	✗	✓
$\in \mathcal{S}$	$\notin \mathcal{S}$	$\Lambda \times \Lambda$	$\in \mathcal{S}$	$\notin \mathcal{S}$	$((a:r, b:r), (\lambda_a:\lambda_r, \lambda_b:\lambda_r))$	✗	✓
$\notin \mathcal{S}$	$\in \mathcal{S}$	$\Lambda \times \Lambda$	$\in \mathcal{S}$	$\in \mathcal{S}$	$((b:r, a:r), (\lambda_a:\lambda_r, \lambda_a:\lambda_r))$	✗	✓
$\notin \mathcal{S}$	$\in \mathcal{S}$	$\Lambda \times \Lambda$	$\in \mathcal{S}$	$\notin \mathcal{S}$	$((b:r, a:r), (\lambda_a:\lambda_r, \lambda_b:\lambda_r))$	✗	✓
$\notin \mathcal{S}$	$\in \mathcal{S}$	$\Lambda \times \Lambda$	$\notin \mathcal{S}$	$\in \mathcal{S}$	$((b:r, a:r), (\lambda_b:\lambda_r, \lambda_a:\lambda_r))$	✗	✓
$\notin \mathcal{S}$	$\in \mathcal{S}$	$\Lambda \times \Lambda$	$\notin \mathcal{S}$	$\notin \mathcal{S}$	$((b:r, a:r), (\lambda_b:\lambda_r, \lambda_b:\lambda_r))$	✗	✓
$\notin \mathcal{S}$	$\notin \mathcal{S}$	$\Lambda \times \Lambda$	$\in \mathcal{S}$	$\in \mathcal{S}$	$((b:r, b:r), (\lambda_a:\lambda_r, \lambda_a:\lambda_r))$	✗	✓
$\notin \mathcal{S}$	$\notin \mathcal{S}$	$\Lambda \times \Lambda$	$\in \mathcal{S}$	$\notin \mathcal{S}$	$((b:r, b:r), (\lambda_a:\lambda_r, \lambda_b:\lambda_r))$	✗	✓
Otherwise					...	✓	✓

In practice

`iptables` universal, `ipfw` and `pf` not universal and equally expressive

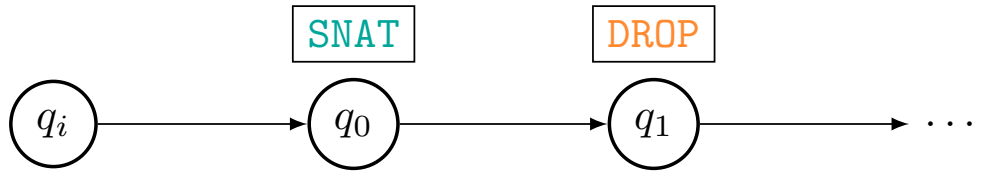
Function Expressivity of firewall language $\mathcal{L}$

- The semantics of a firewall is a **function** $f : \mathbb{P} \to \mathcal{T}_{\mathbb{P}}$
- Is function expressivity **the same** of pairs expressivity?

Function Expressivity of firewall language $\mathcal{L}$

- The semantics of a firewall is a **function** $f : \mathbb{P} \to \mathcal{T}_{\mathbb{P}}$
- Is function expressivity **the same** of pairs expressivity? **NO!**
- The management of different pairs may **interfere** one with the others

$$(p_1, t_1) = ((\texttt{1.1.1.1} : 1,\ \texttt{151.15.185.183} : 1),\ \bot)$$
$$(p_2, t_2) = ((\texttt{1.1.1.1} : 1,\ \texttt{192.168.0.1} : 1),\ (\lambda_{\texttt{151.15.185.183}} : id,\ id : id))$$
$$t_2(p_2) = p_1$$



$p_1 = (\texttt{1.1.1.1} : 1,\ \texttt{151.15.185.183} : 1)$
$p_2 = (\texttt{1.1.1.1} : 1,\ \texttt{192.168.0.1} : 1)$

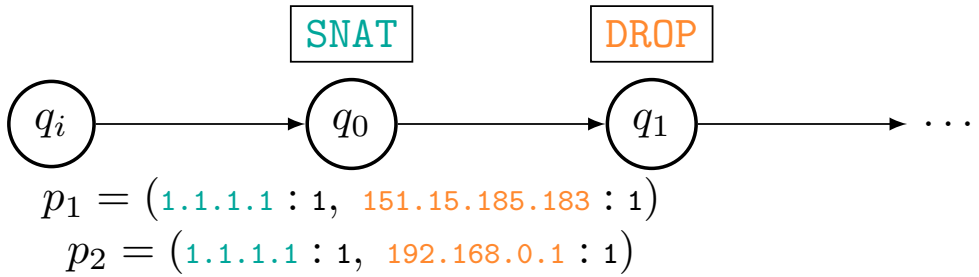
Function Expressivity

Given a function $f : \mathbb{P} \to \mathcal{T}_{\mathbb{P}}$
does it exist a configuration in $\mathcal{L}$ having f as semantics?

Function Expressivity of firewall language $\mathcal{L}$

- The semantics of a firewall is a **function** $f : \mathbb{P} \to \mathcal{T}_{\mathbb{P}}$
- Is function expressivity **the same** of pairs expressivity? **NO!**
- The management of different pairs may **interfere** one with the others

$$(p_1, t_1) = ((\texttt{1.1.1.1} : 1,\ \texttt{151.15.185.183} : 1),\ \bot)$$
$$(p_2, t_2) = ((\texttt{1.1.1.1} : 1,\ \texttt{192.168.0.1} : 1),\ (\lambda_{\texttt{151.15.185.183}} : id,\ id : id))$$
$$t_2(p_2) = p_1$$



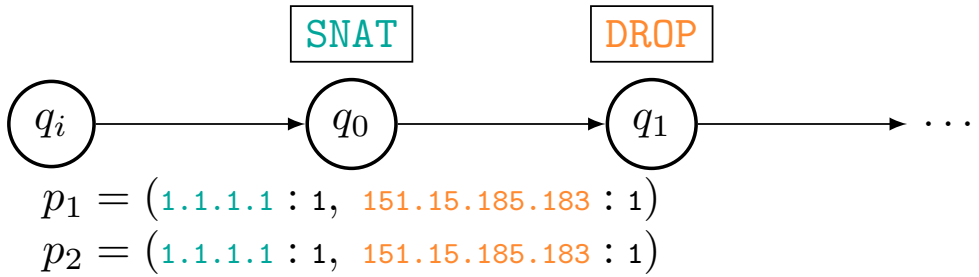
Function Expressivity

Given a function $f : \mathbb{P} \to \mathcal{T}_{\mathbb{P}}$
does it exist a configuration in $\mathcal{L}$ having f as semantics?

Function Expressivity of firewall language $\mathcal{L}$

- The semantics of a firewall is a **function** $f : \mathbb{P} \to \mathcal{T}_{\mathbb{P}}$
- Is function expressivity **the same** of pairs expressivity? **NO!**
- The management of different pairs may **interfere** one with the others

$$(p_1, t_1) = ((\texttt{1.1.1.1} : 1,\ \texttt{151.15.185.183} : 1),\ \bot)$$
$$(p_2, t_2) = ((\texttt{1.1.1.1} : 1,\ \texttt{192.168.0.1} : 1),\ (\lambda_{\texttt{151.15.185.183}} : id,\ id : id))$$
$$t_2(p_2) = p_1$$



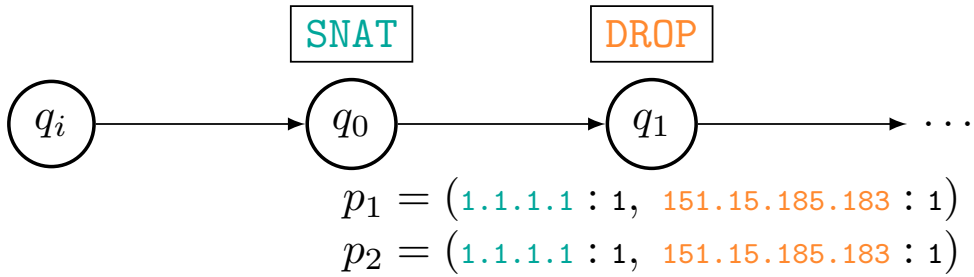
Function Expressivity

Given a function $f : \mathbb{P} \to \mathcal{T}_{\mathbb{P}}$
does it exist a configuration in $\mathcal{L}$ having f as semantics?

Function Expressivity of firewall language $\mathcal{L}$

- The semantics of a firewall is a **function** $f : \mathbb{P} \to \mathcal{T}_{\mathbb{P}}$
- Is function expressivity **the same** of pairs expressivity? **NO!**
- The management of different pairs may **interfere** one with the others

$$(p_1, t_1) = ((\texttt{1.1.1.1} : 1,\ \texttt{151.15.185.183} : 1),\ \bot)$$
$$(p_2, t_2) = ((\texttt{1.1.1.1} : 1,\ \texttt{192.168.0.1} : 1),\ (\lambda_{\texttt{151.15.185.183}} : id,\ id : id))$$
$$t_2(p_2) = p_1$$



Function Expressivity

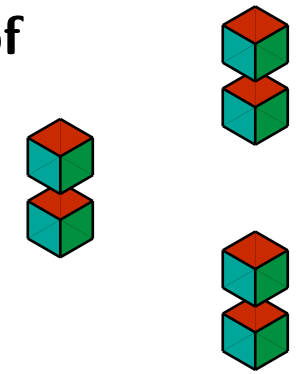
Given a function $f : \mathbb{P} \to \mathcal{T}_{\mathbb{P}}$
does it exist a configuration in $\mathcal{L}$ having f as semantics?

Checking expressible functions f [ITASEC19]

Function f represented as **sets of pairs** (P, t)

P is a multi-cube of packets

t is a transformation



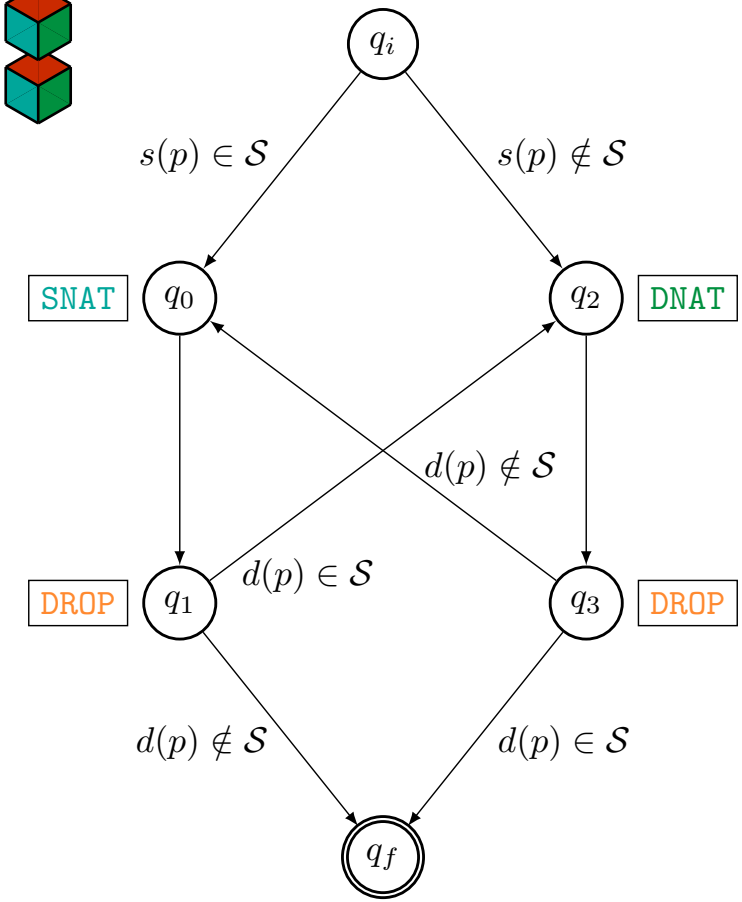
Algorithm

Given a control diagram with labels

Returns true if f the expressible

- **For each** pair (P, t) with $t \neq \perp$
  - Find the **path**
  - **For each** node q
    - Preceding nodes $\rightarrow \mathbf{P_q}$
    - Labels in $q \rightarrow \mathbf{t_q}$
- Special management for pairs $(P, \perp)$

The case of `pf`



$\mathcal{S}$ are the addresses of the firewall

Results

`iptables` not universal and incomparable with others,
`ipfw` more expressive than `pf`

Results

`iptables` not universal and incomparable with others,
`ipfw` more expressive than `pf`

Tags are not considered

Pair expressivity not affected, function expressivity may be

guess: the two expressivity coincide if tags are considered $\rightarrow$ function expressivity express when tags are **really needed**

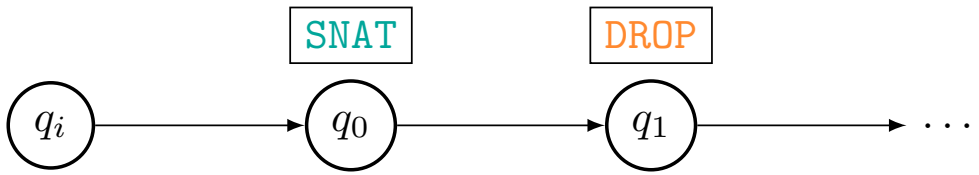
Results

`iptables` not universal and incomparable with others,
`ipfw` more expressive than `pf`

Tags are not considered

Pair expressivity not affected, function expressivity may be

guess: the two expressivity coincide if tags are considered $\rightarrow$ function expressivity express when tags are **really needed**



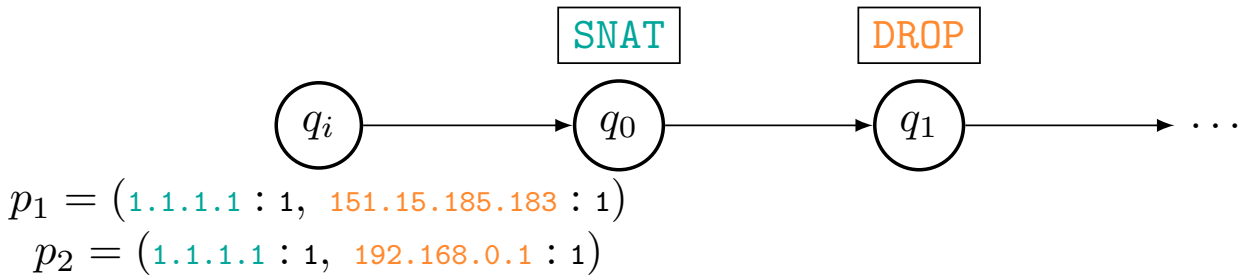
Results

`iptables` not universal and incomparable with others,
`ipfw` more expressive than `pf`

Tags are not considered

Pair expressivity not affected, function expressivity may be

guess: the two expressivity coincide if tags are considered $\rightarrow$ function expressivity express when tags are **really needed**



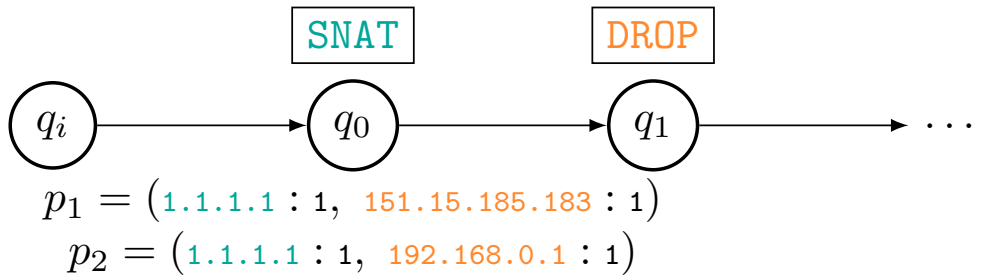
Results

`iptables` not universal and incomparable with others,
`ipfw` more expressive than `pf`

Tags are not considered

Pair expressivity not affected, function expressivity may be

guess: the two expressivity coincide if tags are considered $\rightarrow$ function expressivity express when tags are **really needed**



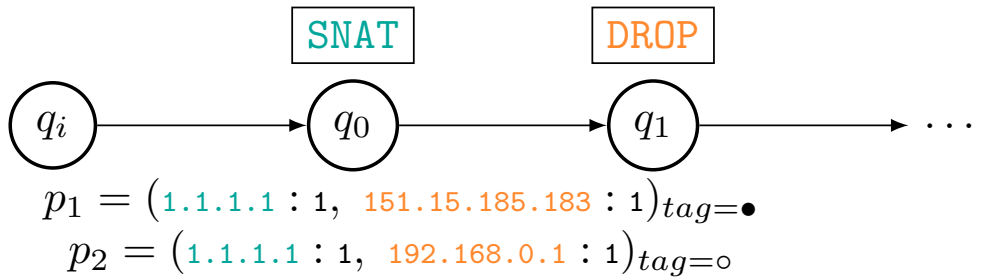
Results

`iptables` not universal and incomparable with others,
`ipfw` more expressive than `pf`

Tags are not considered

Pair expressivity not affected, function expressivity may be

guess: the two expressivity coincide if tags are considered $\rightarrow$ function expressivity express when tags are **really needed**



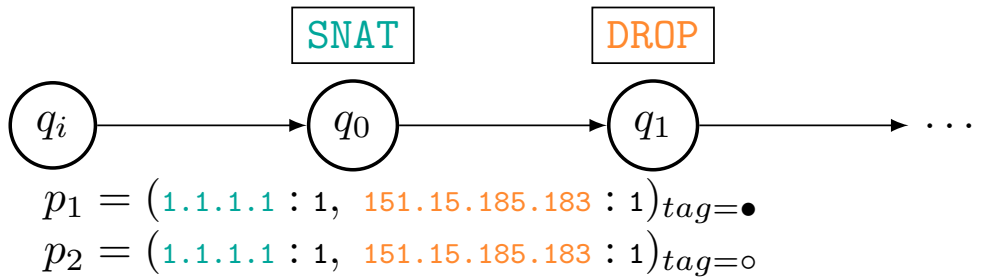
Results

`iptables` not universal and incomparable with others,
`ipfw` more expressive than `pf`

Tags are not considered

Pair expressivity not affected, function expressivity may be

guess: the two expressivity coincide if tags are considered $\rightarrow$ function expressivity express when tags are **really needed**



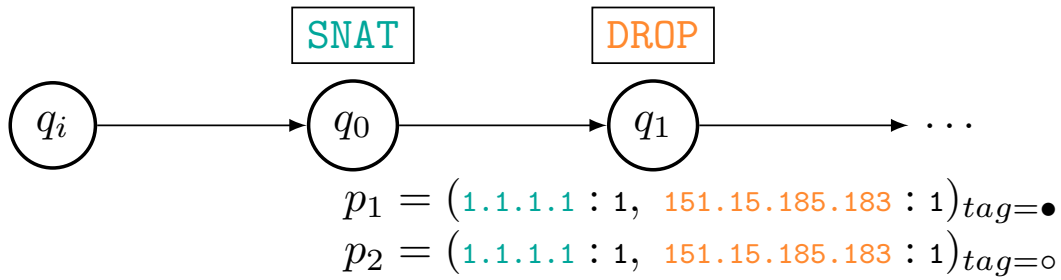
Results

`iptables` not universal and incomparable with others,
`ipfw` more expressive than `pf`

Tags are not considered

Pair expressivity not affected, function expressivity may be

guess: the two expressivity coincide if tags are considered $\rightarrow$ function expressivity express when tags are **really needed**



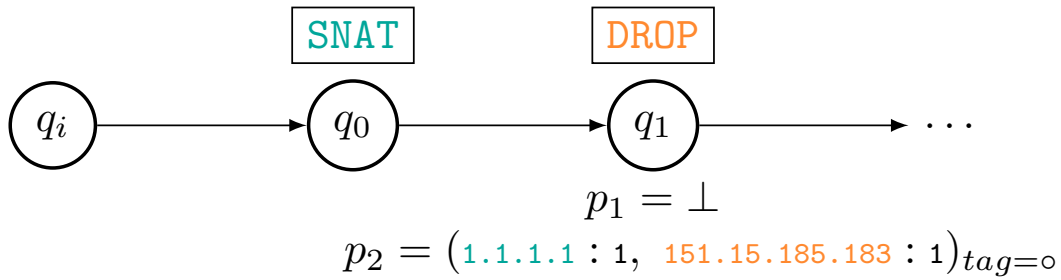
Results

`iptables` not universal and incomparable with others,
`ipfw` more expressive than `pf`

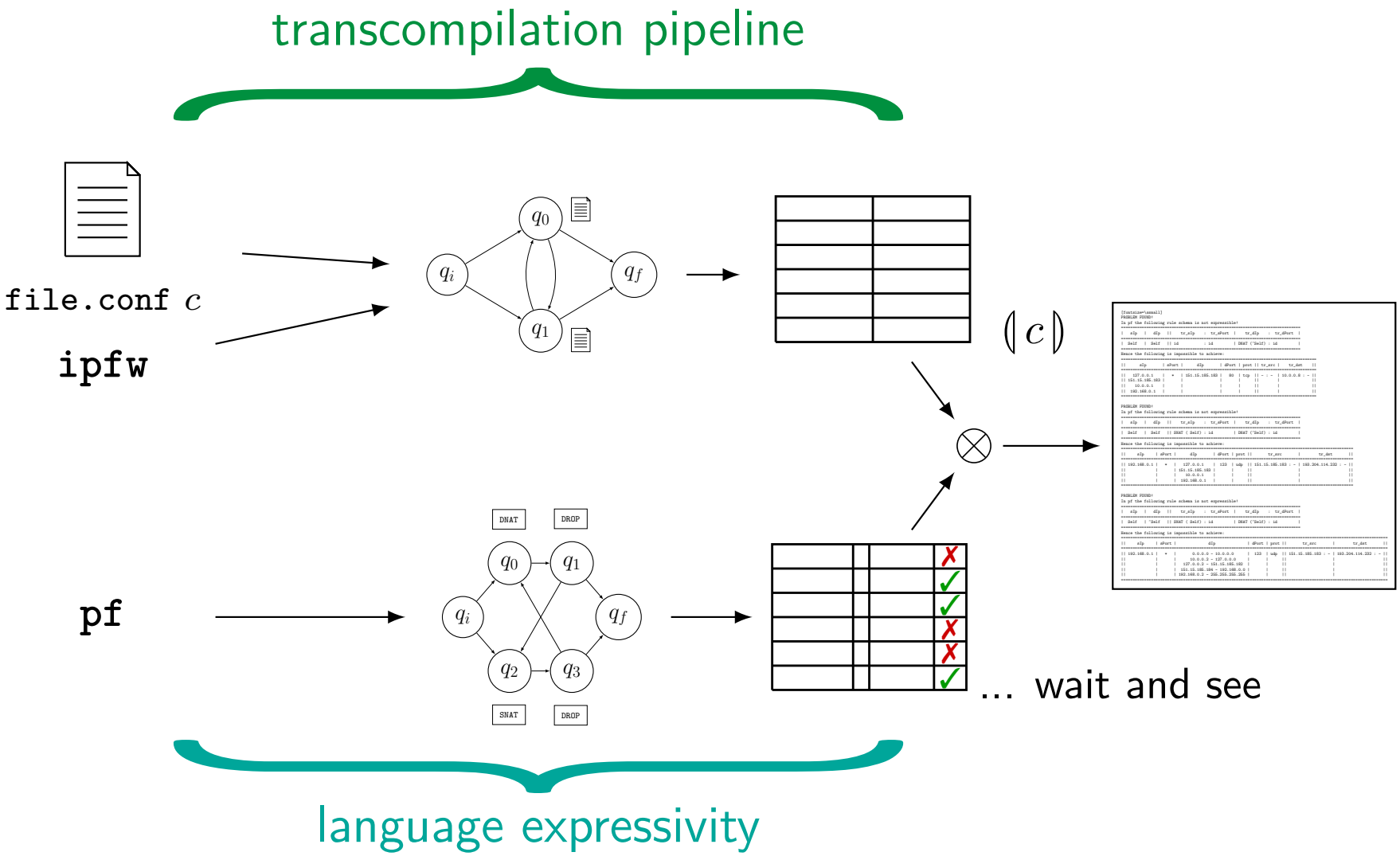
Tags are not considered

Pair expressivity not affected, function expressivity may be

guess: the two expressivity coincide if tags are considered $\rightarrow$ function expressivity express when tags are **really needed**



F2F checks expressivity



Extra – *F2F* at work

```
(venv) user@here:~/$ fwp iptables ~/interfaces ~/iptables.conf pf

PROBLEM FOUND!
In pf the following rule schema is not expressible!
```

slp	dlp	tr_slp : tr_sPort	tr_dlp : tr_dPort
Self	Self	id : id	DNAT (~Self) : id

Hence the following is impossible to achieve:

slp	sPort	dlp	dPort	prot	tr_src	tr_dst
127.0.0.1 151.15.185.183 10.0.0.1 192.168.0.1	*	151.15.185.183	80	tcp	– : –	10.0.0.8 : –

PROBLEM FOUND!
In pf the following rule schema is not expressible!

slp	dlp	tr_slp : tr_sPort	tr_dlp : tr_dPort
Self	Self	SNAT (Self) : id	DNAT (~Self) : id

Hence the following is impossible to achieve:

slp	sPort	dlp	dPort	prot	tr_src	tr_dst
192.168.0.1	*	127.0.0.1 151.15.185.183 10.0.0.1 192.168.0.1	123	udp	151.15.185.183 : –	193.204.114.232 : –

PROBLEM FOUND!
In pf the following rule schema is not expressible!

slp	dlp	tr_slp : tr_sPort	tr_dlp : tr_dPort
Self	~Self	SNAT (Self) : id	DNAT (~Self) : id

Hence the following is impossible to achieve:

slp	sPort	dlp	dPort	prot	tr_src	tr_dst
192.168.0.1	*	0.0.0.0 – 10.0.0.0 10.0.0.2 – 127.0.0.0 127.0.0.2 – 151.15.185.182 151.15.185.184 – 192.168.0.0 192.168.0.2 – 255.255.255.255	123	udp	151.15.185.183 : –	193.204.114.232 : –

```
(venv) user@here:~/$
```