

# SpaceSearch: A Library for Building and Verifying Solver-Aided Tools



**Konstantin  
Weitz**

Steven S.  
Lyubomirsky

Stefan  
Heule

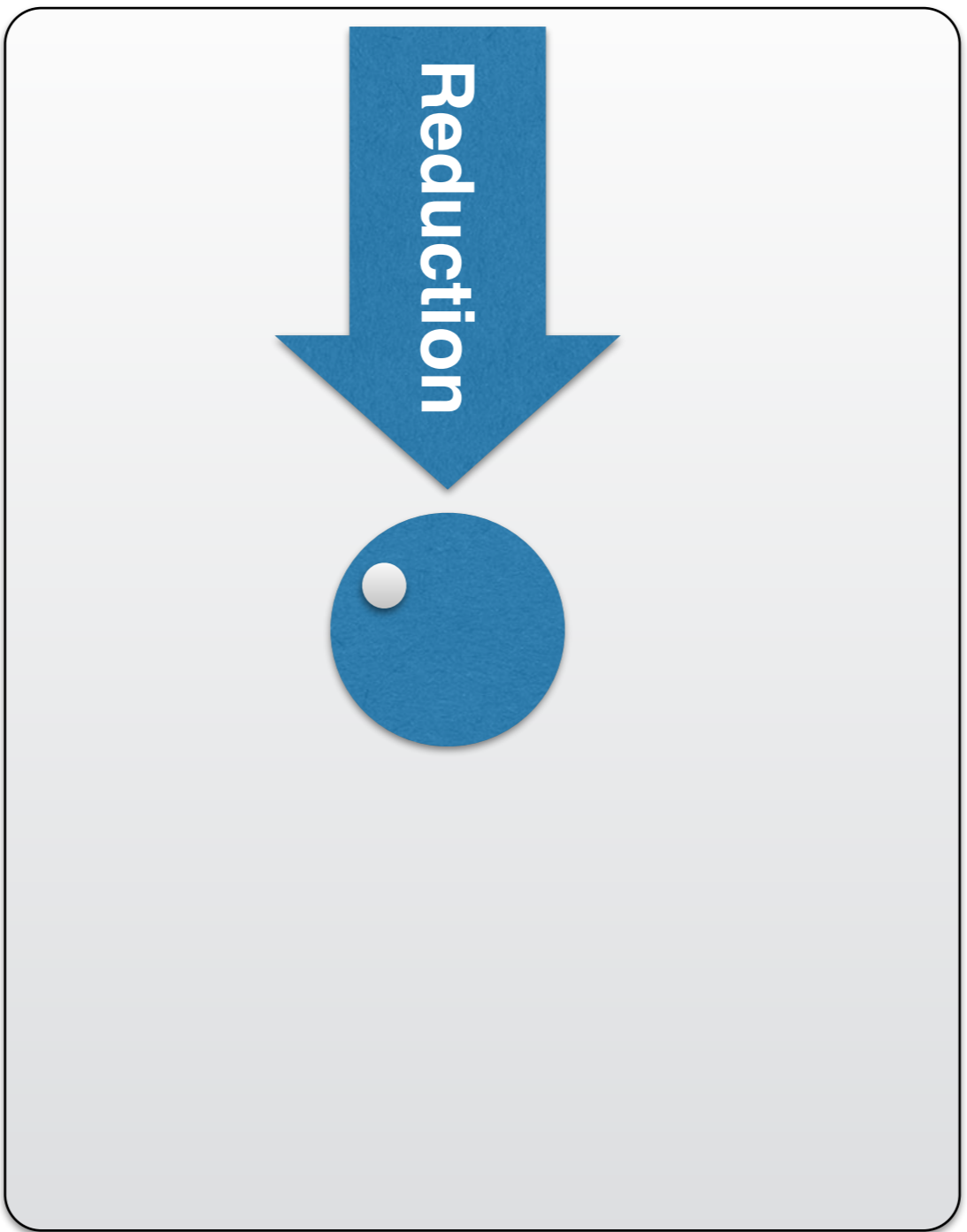
Emina  
Torlak

Michael  
D. Ernst

Zachary  
Tatlock

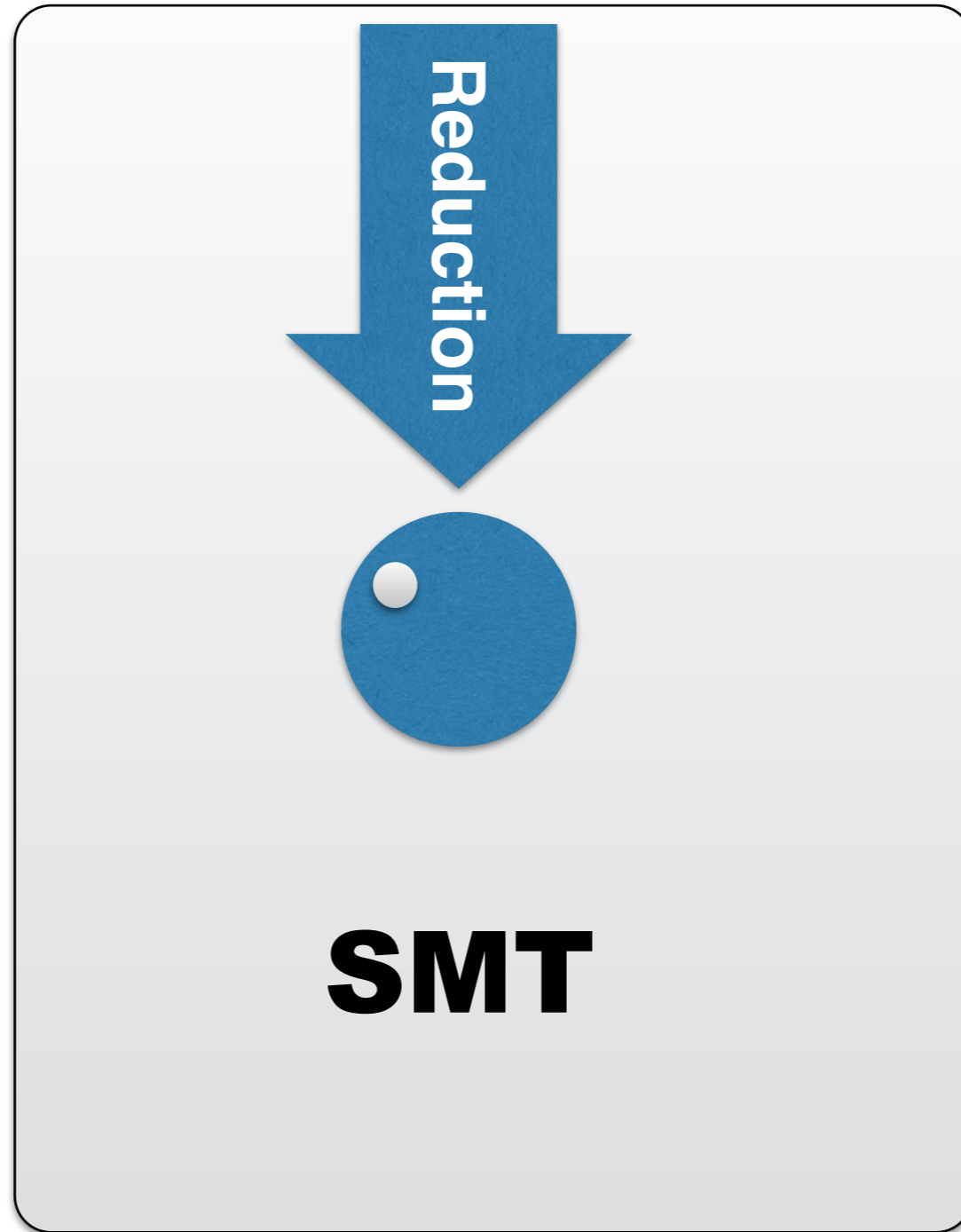




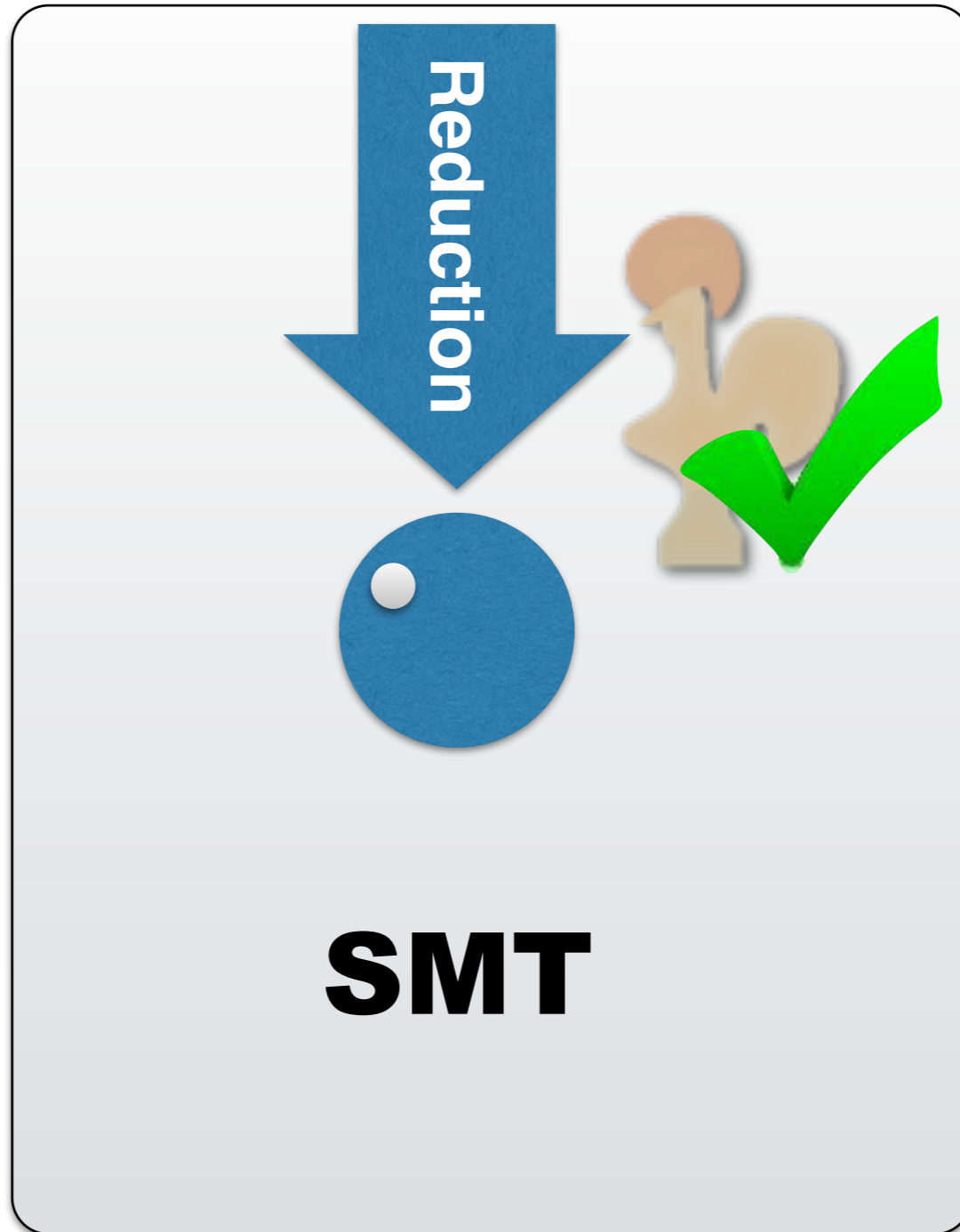




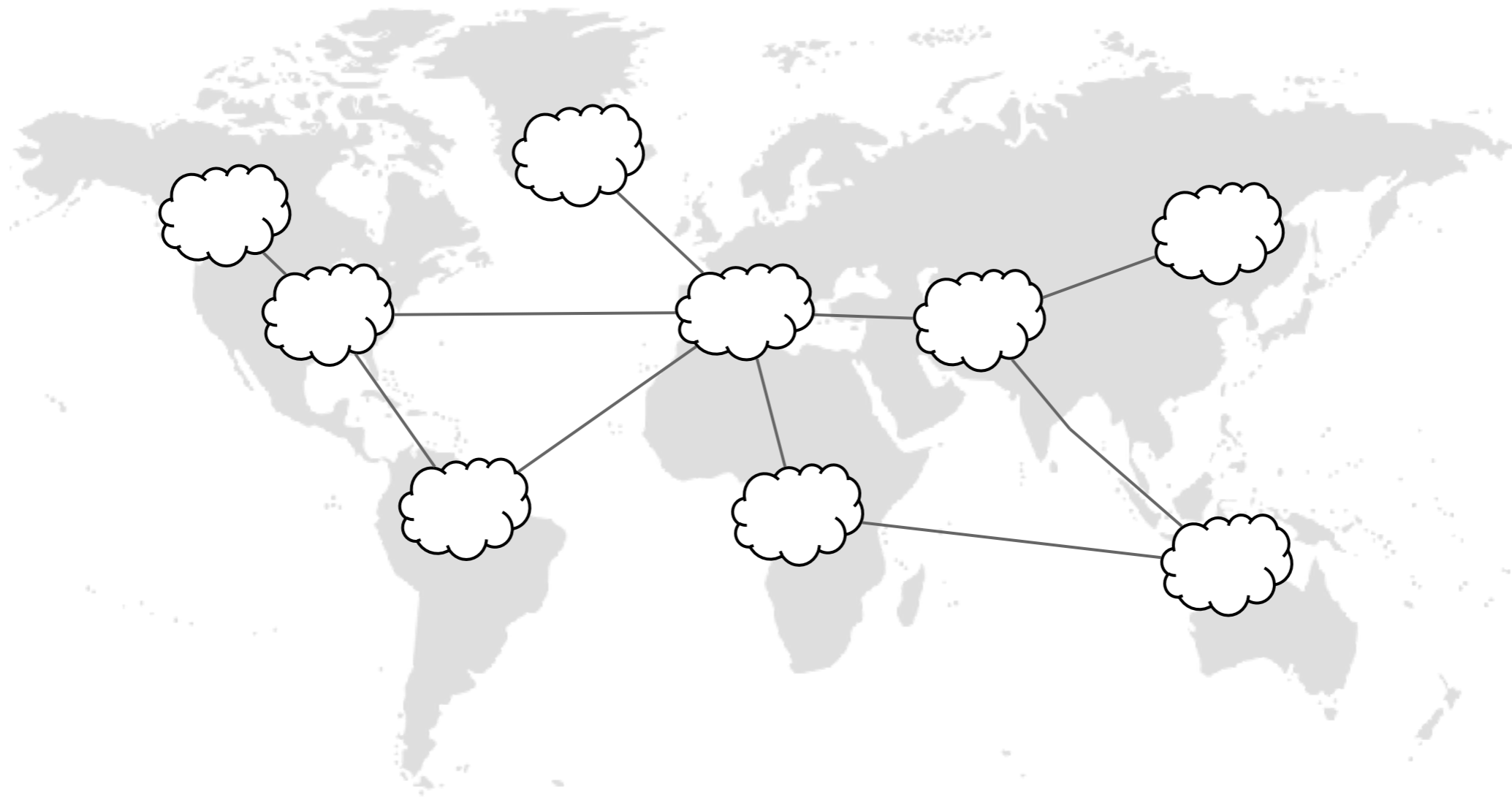
**SMT**



# SpaceSearch

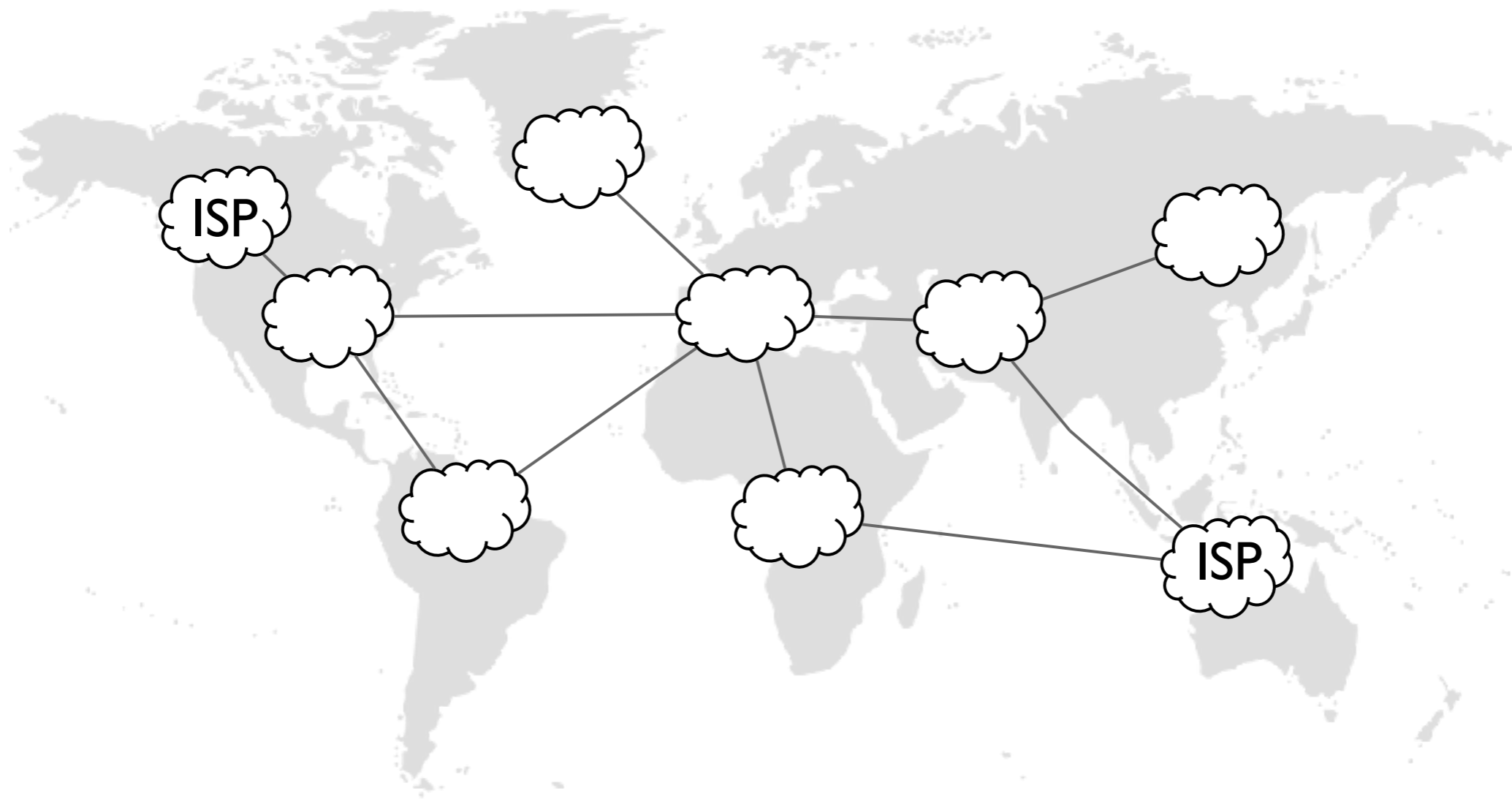


# The Border Gateway Protocol

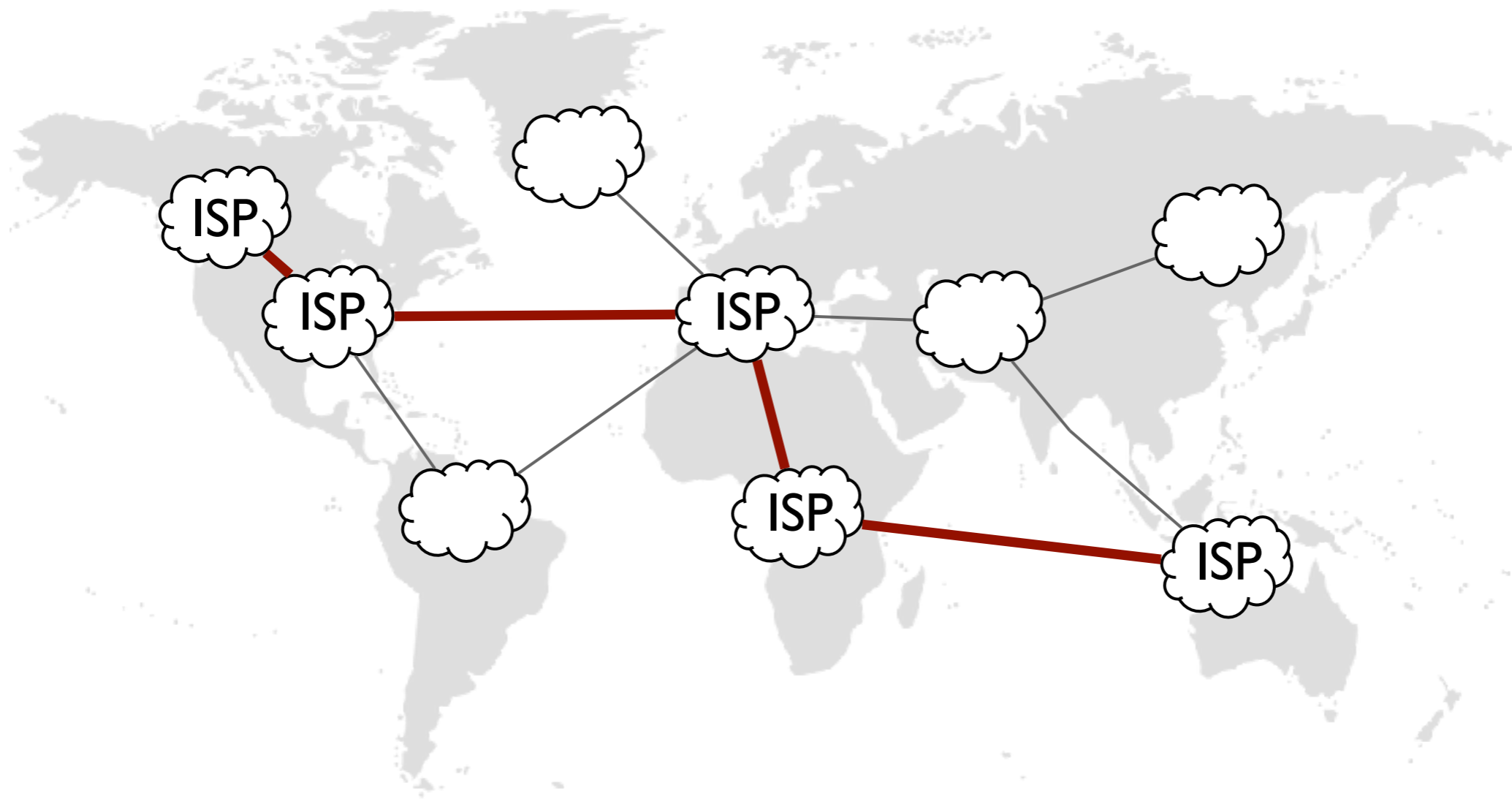




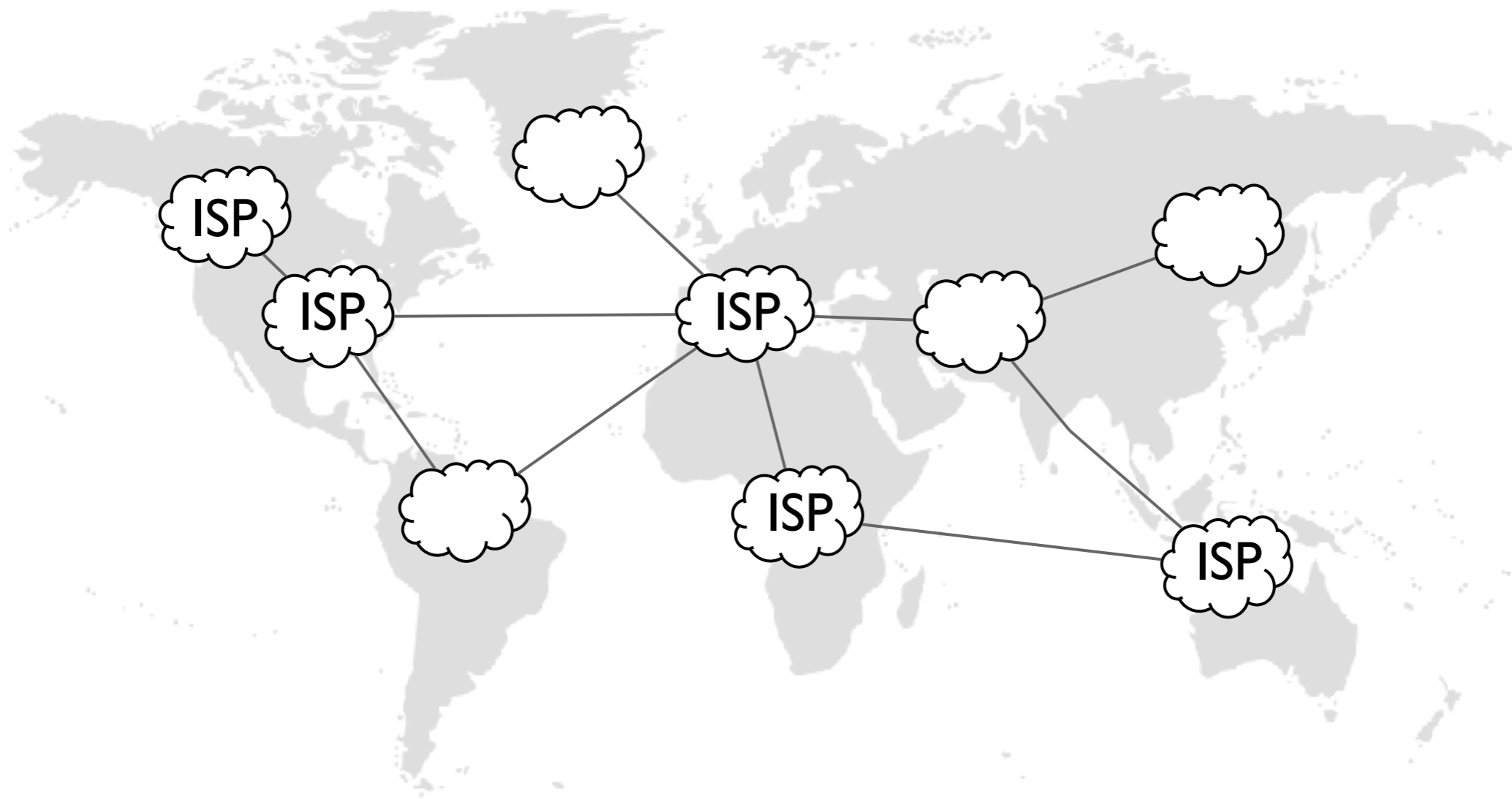
# The Border Gateway Protocol



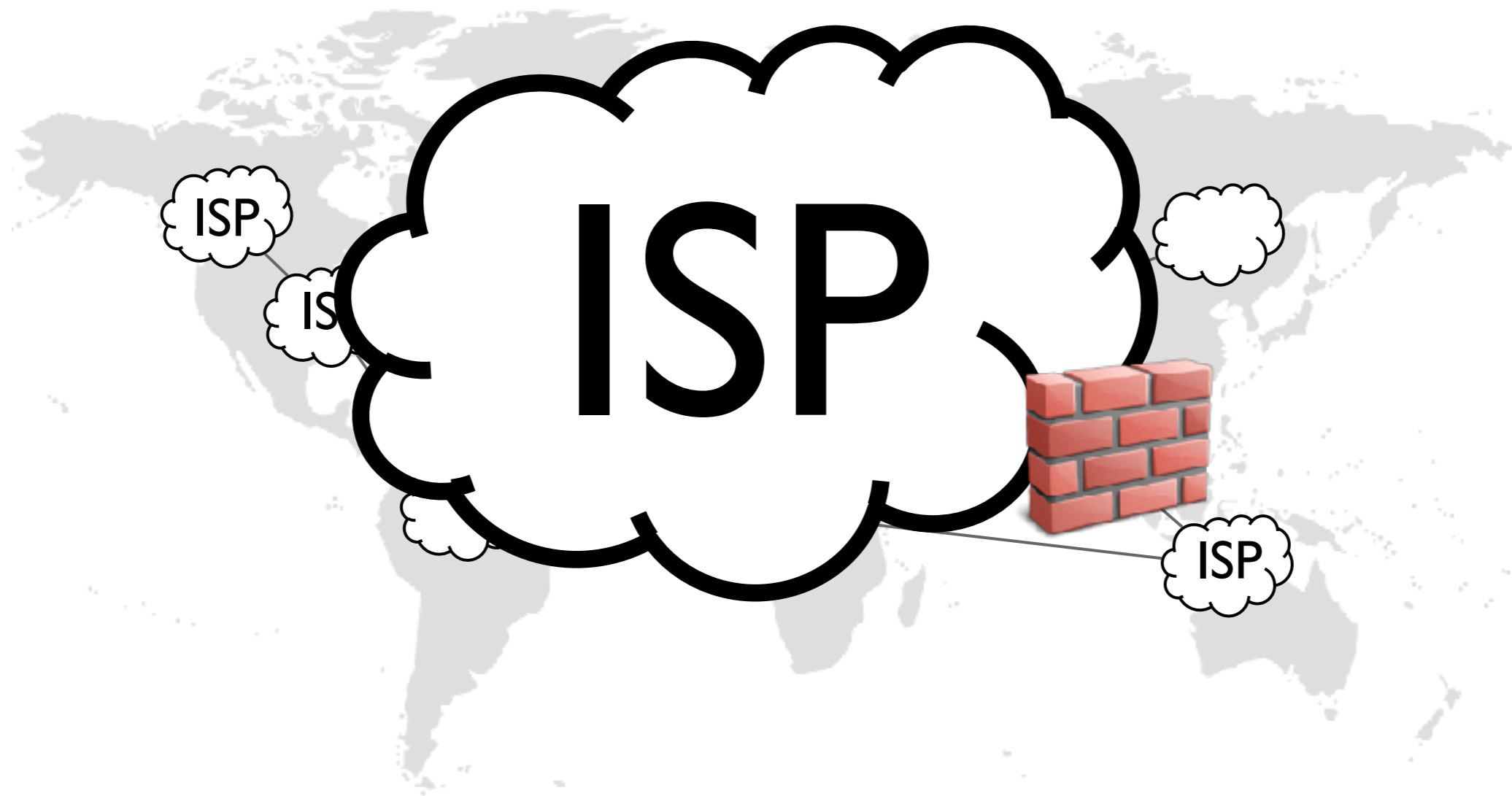
# The Border Gateway Protocol



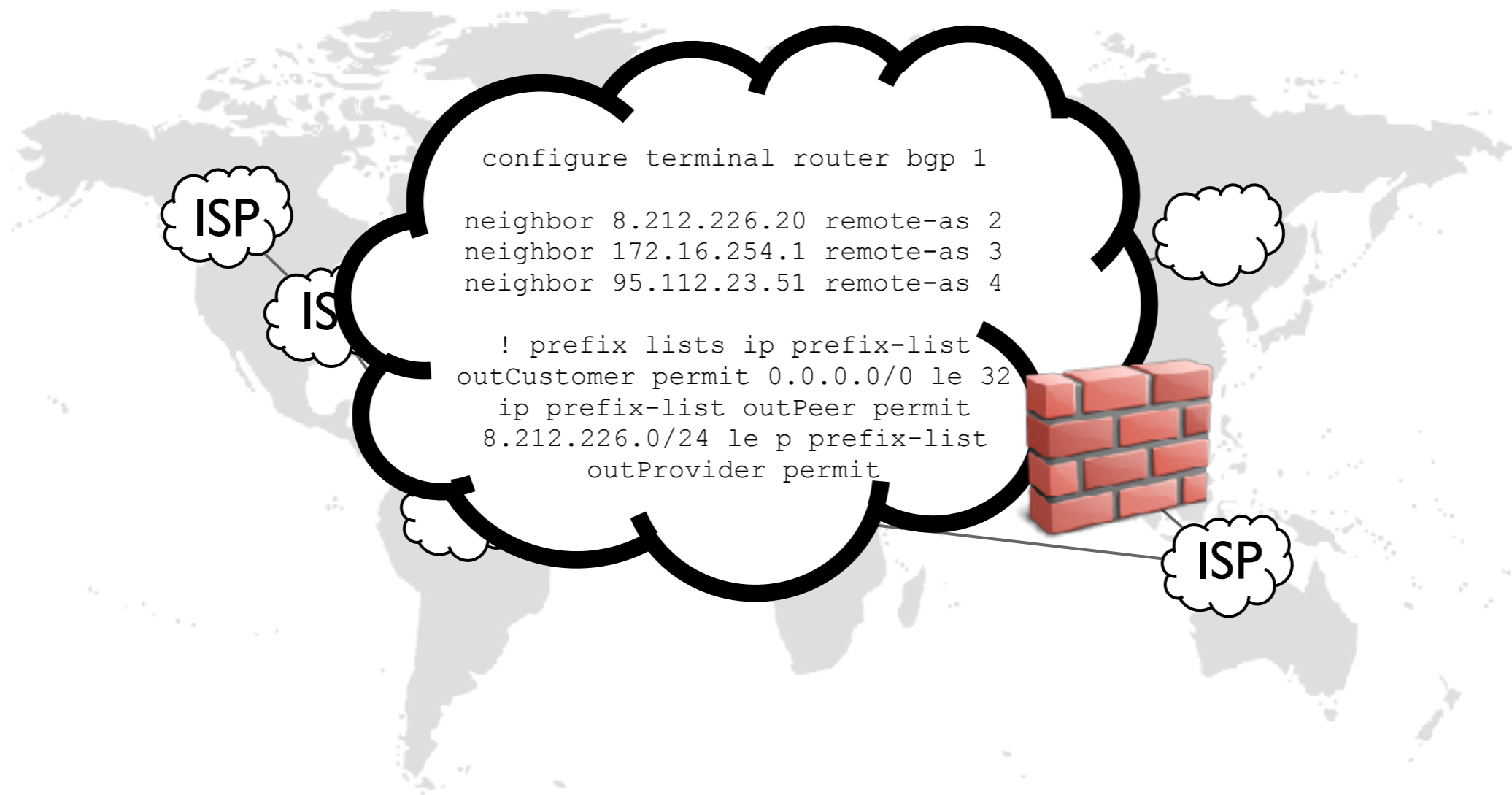
# The Border Gateway Protocol



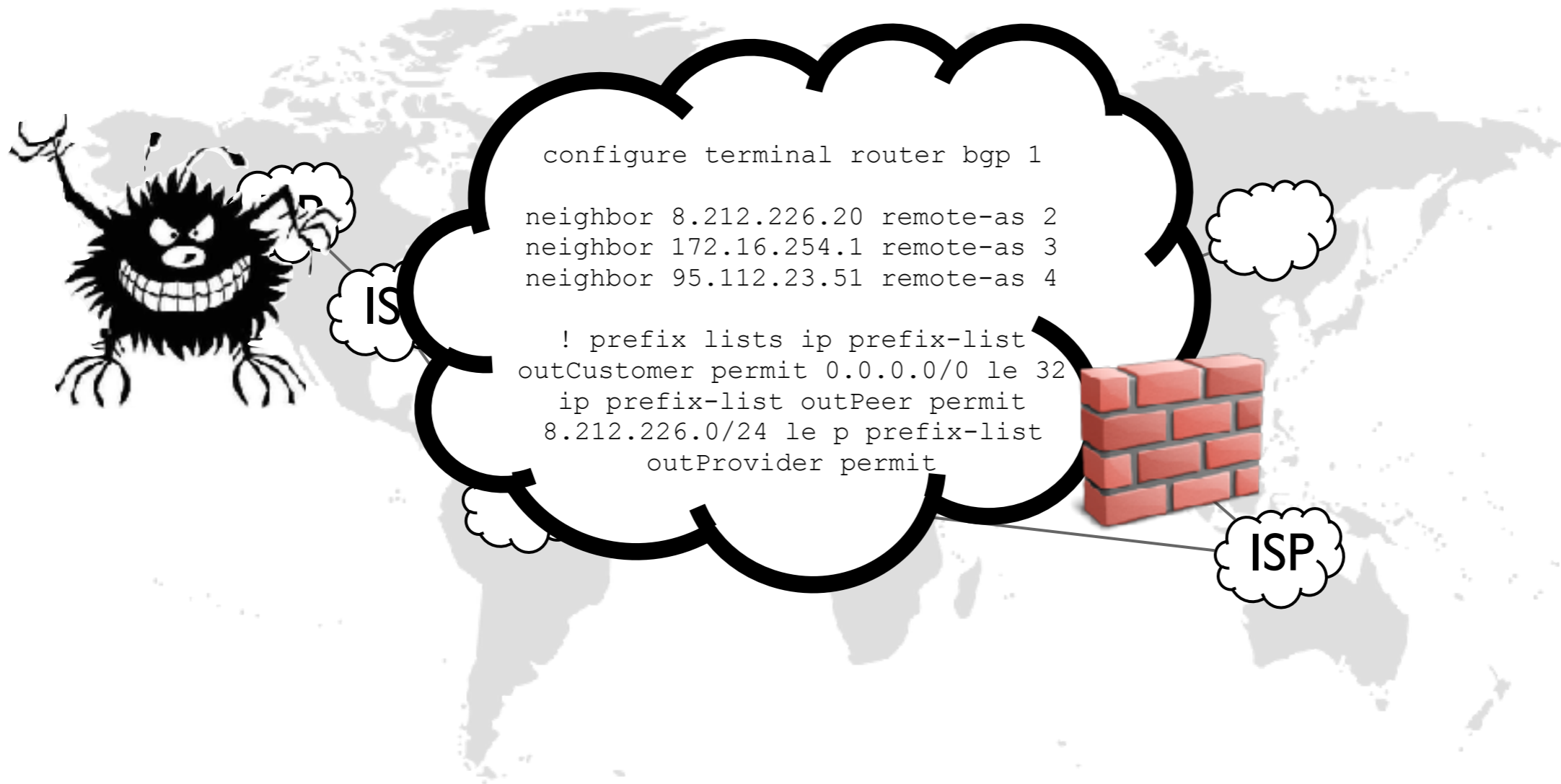
# The Border Gateway Protocol



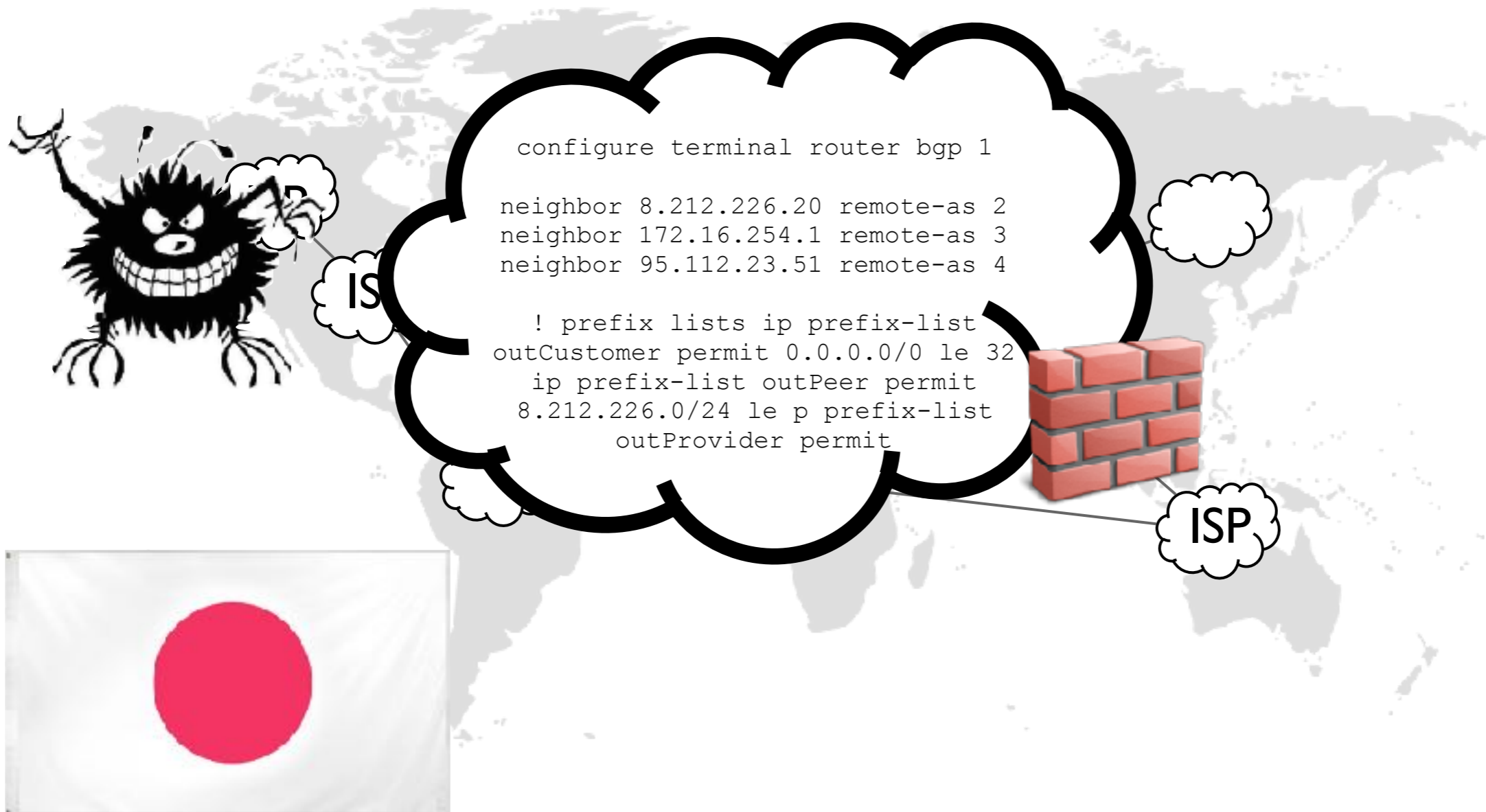
# The Border Gateway Protocol



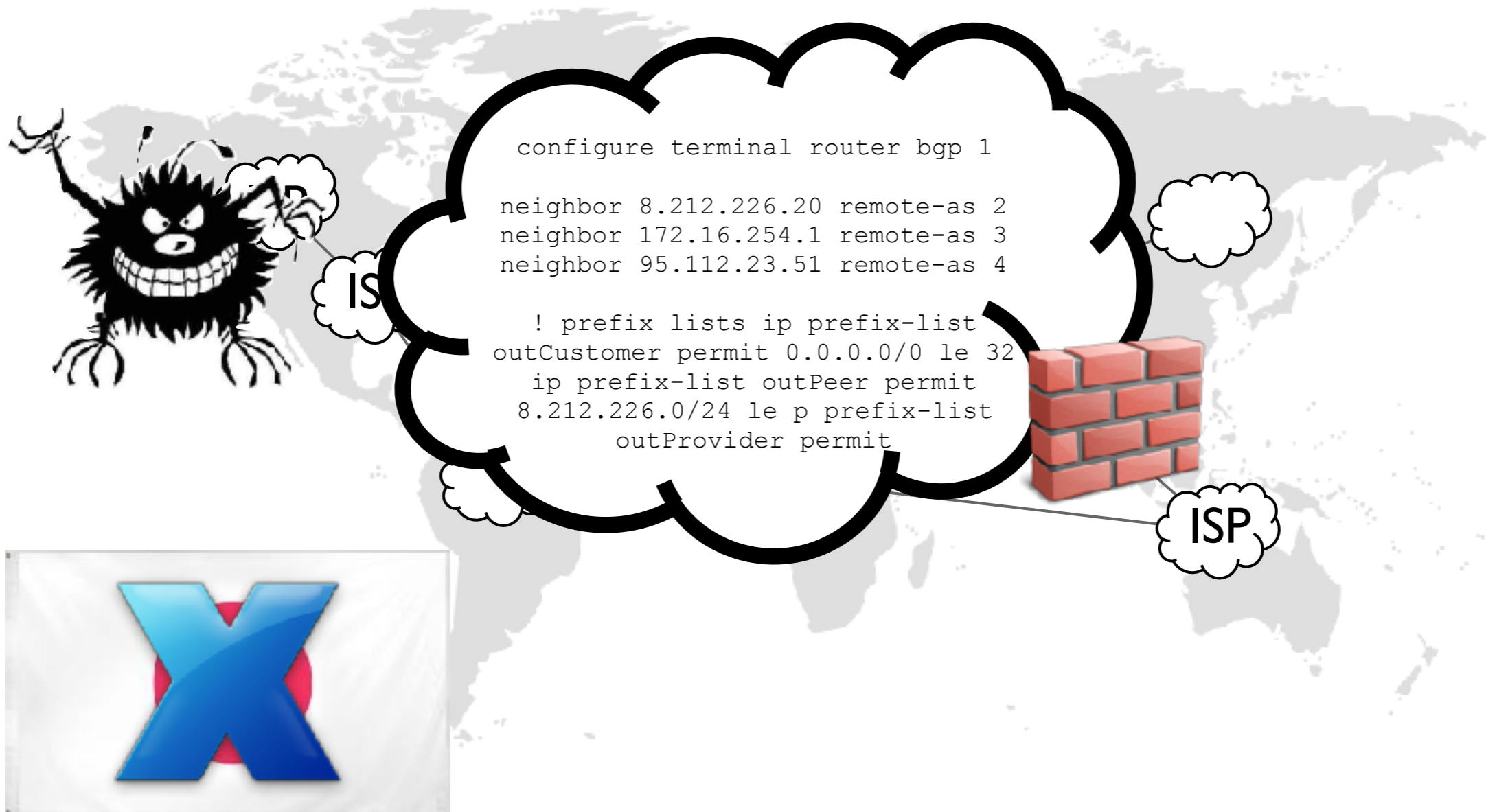
# The Border Gateway Protocol



# The Border Gateway Protocol

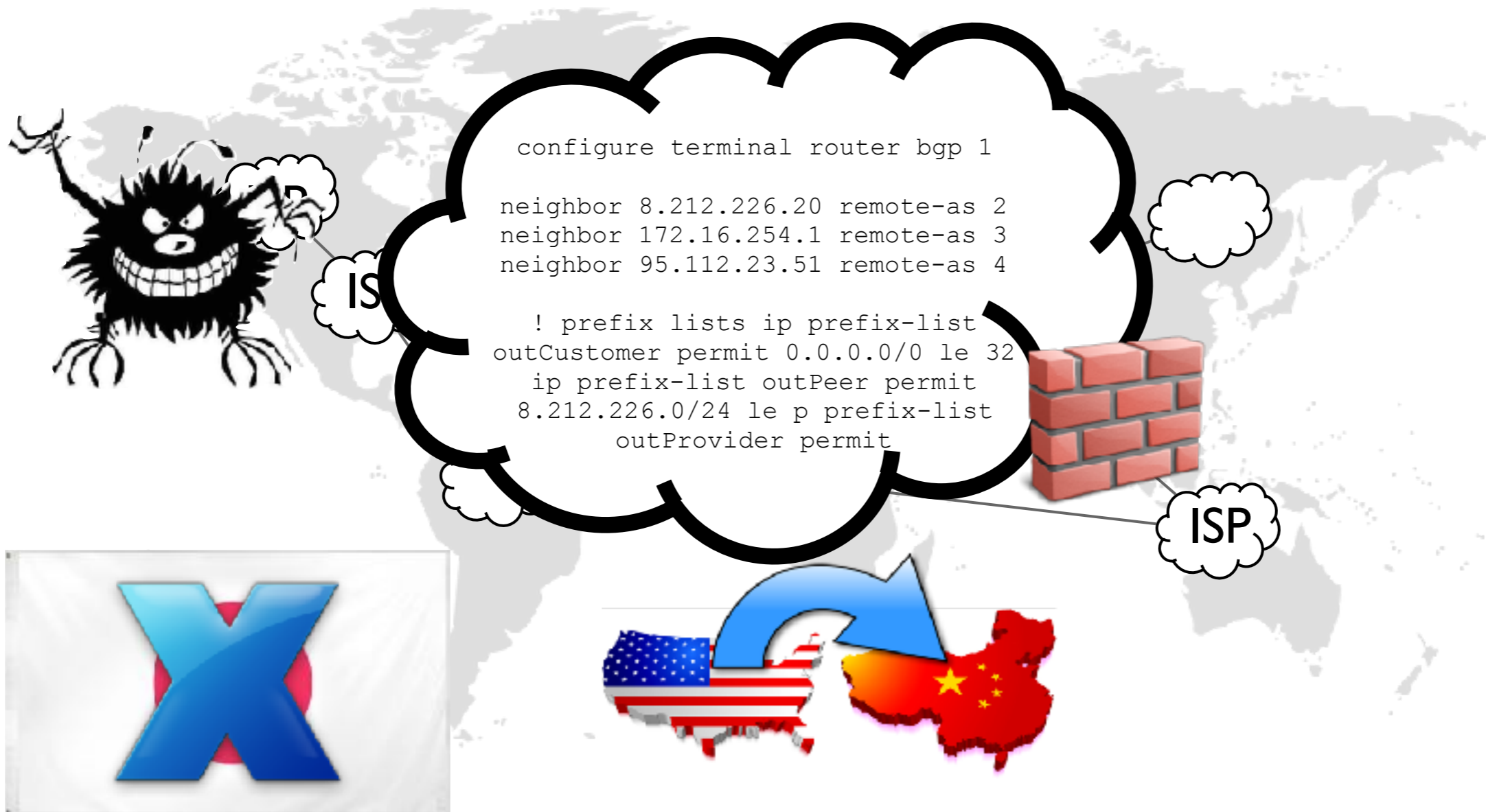


# The Border Gateway Protocol

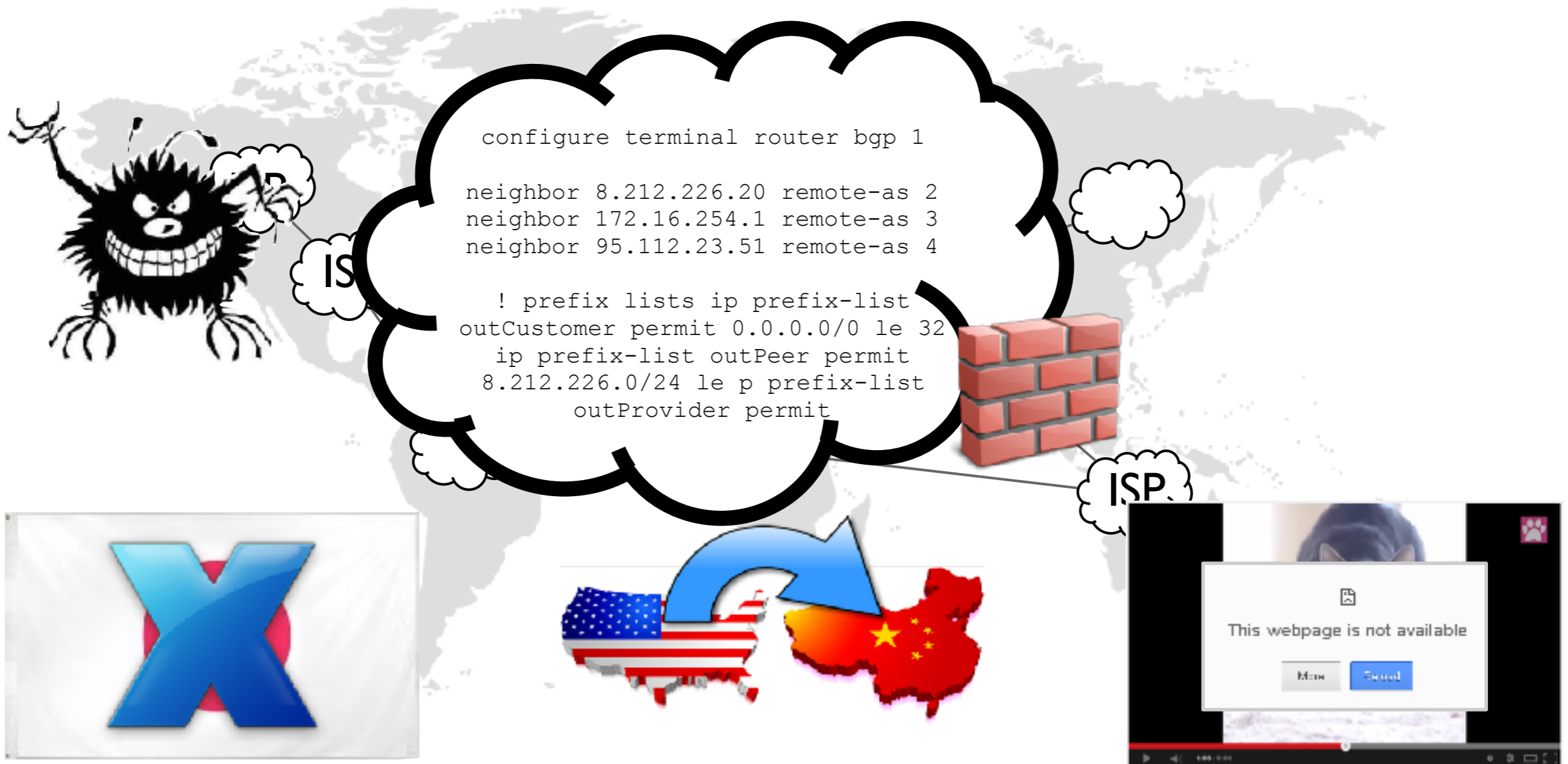




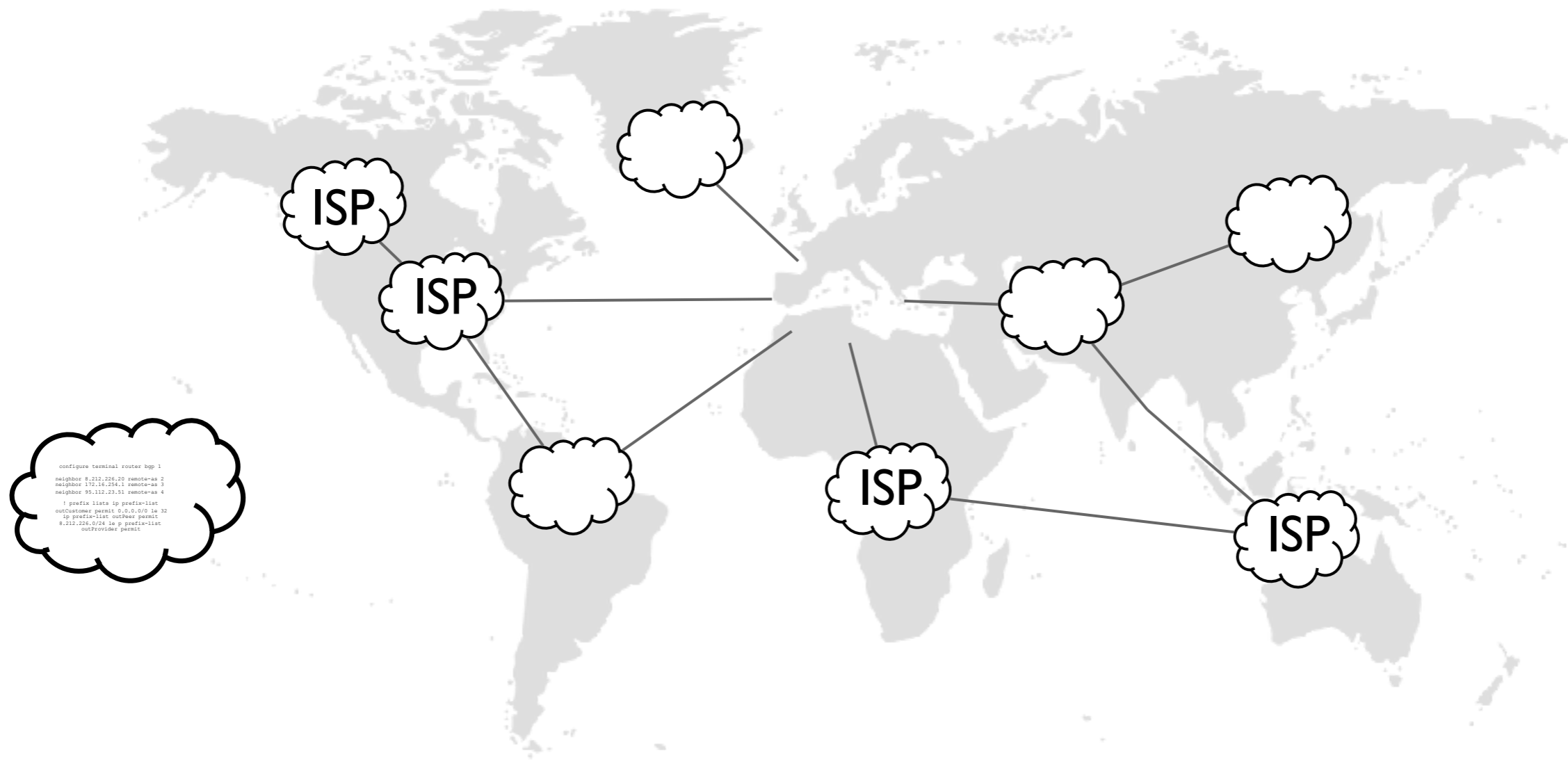
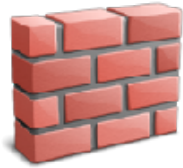
# The Border Gateway Protocol

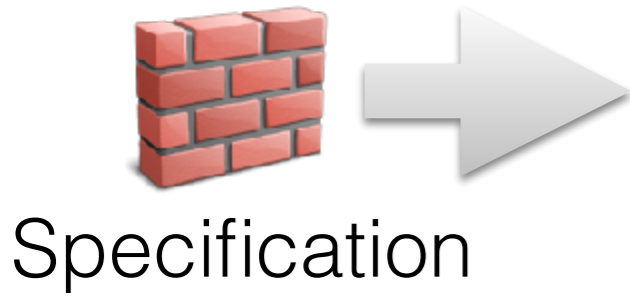


# The Border Gateway Protocol

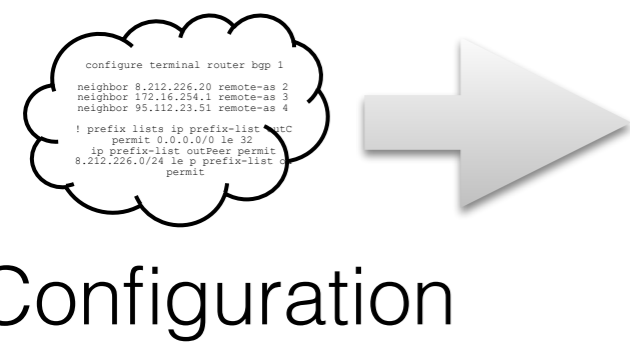


# The Border Gateway Protocol

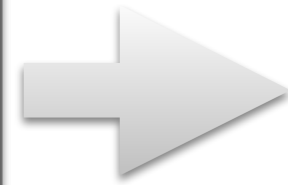




Specification



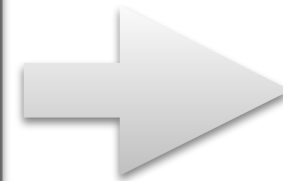
Configuration



Spec Violation



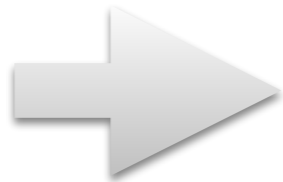
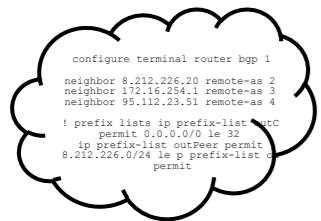
Specification



Spec  
Holds



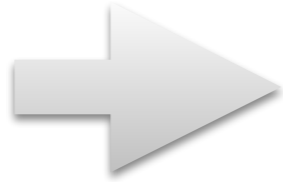
Spec  
Violation



Configuration

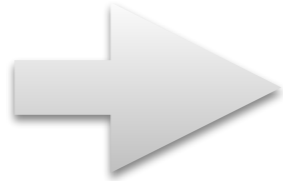
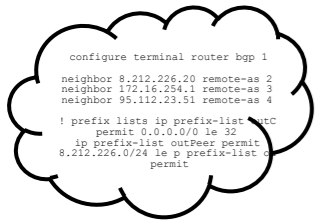
**Bagpipe**

# Bagpipe

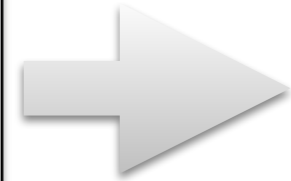


Specification

$\forall t: \text{trace}(\text{cloud icon}), \text{check}(\text{document icon}, t)$



Configuration

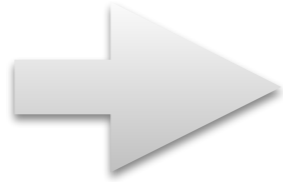


Spec Holds



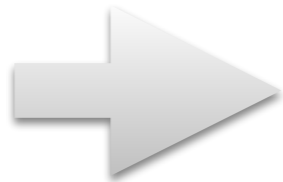
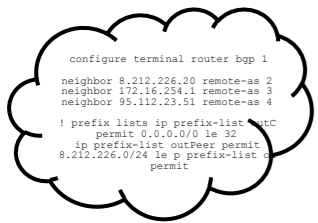
Spec Violation

# Bagpipe

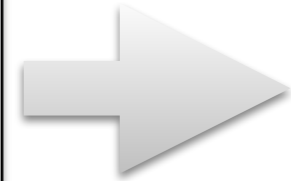


Specification

$\forall t: \text{trace}(\text{cloud icon}, \text{check}(\text{document icon}, t))$



Configuration



Spec Holds

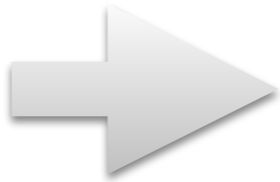


Spec Violation

# Bagpipe



Specification



$\forall t:\text{trace}(\text{configure terminal router bgp}), \text{check}(\text{document icon}, t)$



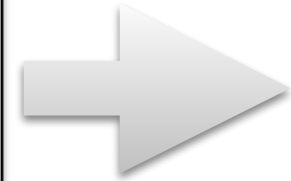
[OOPSLA'16]

fin

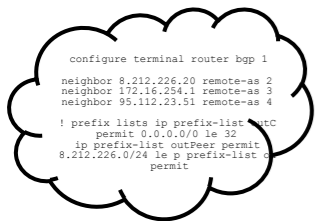
$\forall t:\text{initTrace}(\text{configure terminal router bgp}), \text{check}(\text{document icon}, t)$



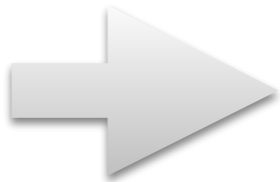
Spec Holds



Spec Violation



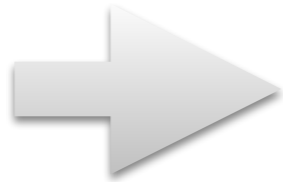
Configuration



```
configure terminal router bgp 1
neighbor 8.212.226.20 remote-as 2
neighbor 172.16.234.1 remote-as 3
neighbor 95.110.23.51 remote-as 4
! prefix-lists ip prefix-list tcc
  permit 0.0.0.0/0 le 32
ip prefix-list outPeer permit
8.212.226.0/24 le p prefix-list o
  permit
```



# Bagpipe



Specification

$\forall t:\text{trace}(\text{configure terminal router bgp}, t)$



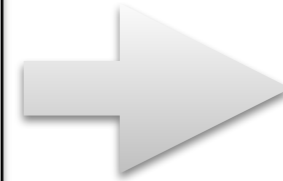
[OOPSLA'16]



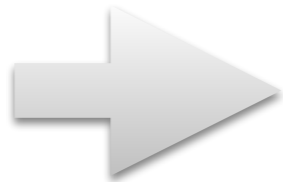
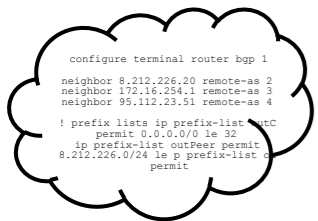
fin



$\forall t:\text{initTrace}(\text{configure terminal router bgp}, t)$



Spec Holds



Configuration

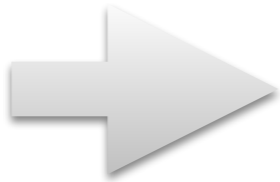


Spec Violation

# Bagpipe



Specification



$\forall t:\text{trace}(\text{cloud icon}, t)$ ,  $\text{check}(\text{document icon}, t)$



fin



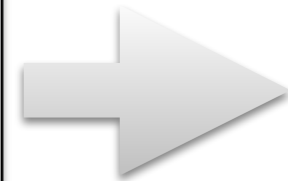
[OOPSLA'16]



$\forall t:\text{initTrace}(\text{cloud icon}, t)$ ,  $\text{check}(\text{document icon}, t)$



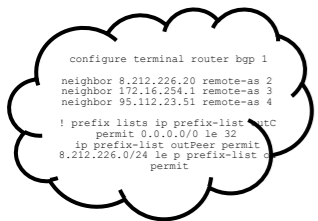
**SMT**



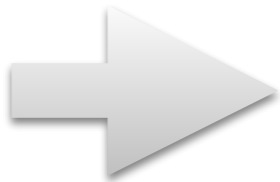
Spec Holds



Spec Violation



Configuration

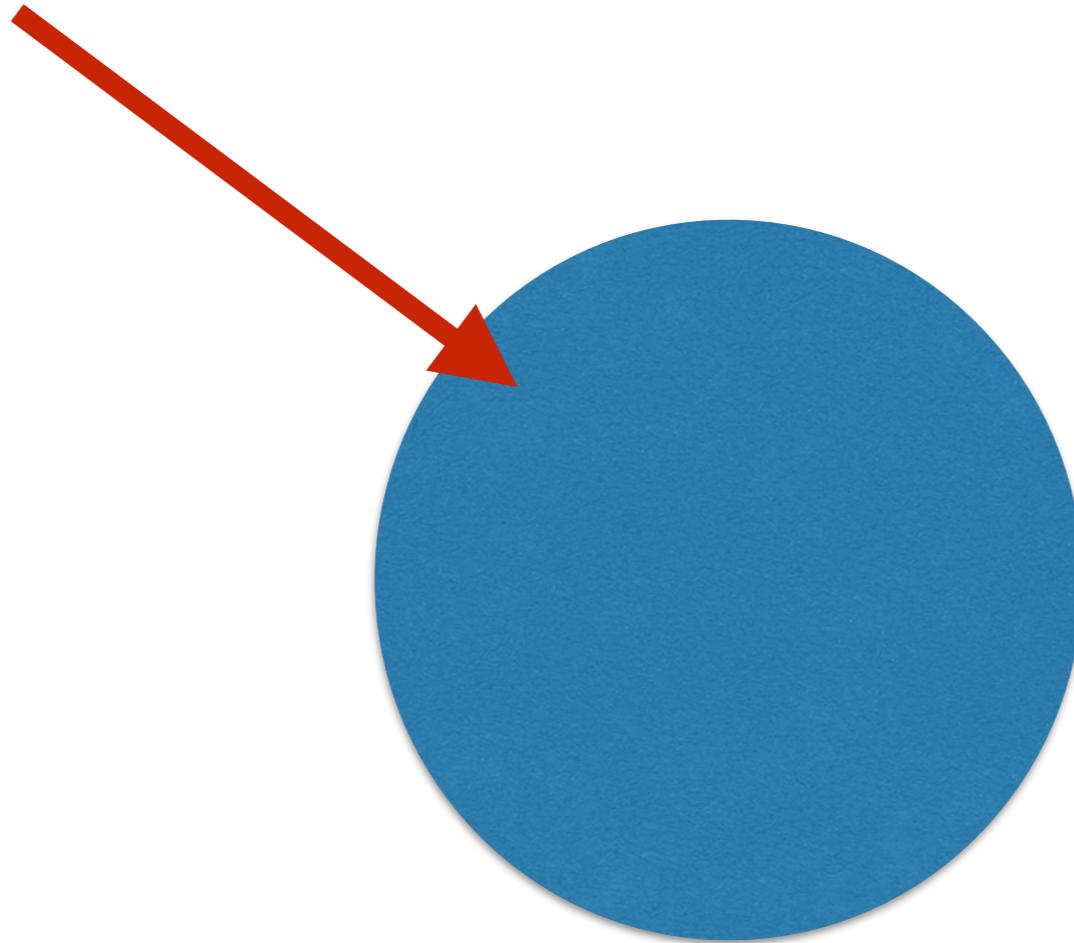


```
configure terminal router bgp 1
neighbor 8.212.226.20 remote-as 2
neighbor 172.16.234.1 remote-as 3
neighbor 95.110.23.51 remote-as 4
! prefix-lists
ip prefix-list outPeer permit 0.0.0.0/0 le 32
ip prefix-list outPeer permit 8.212.226.0/24 le 32
ip prefix-list outPeer permit
```

∇ t:initTrace(, check(, t)

$\forall t: \text{initTrace}(\text{cloud icon}), \text{check}(\text{document icon}, t)$

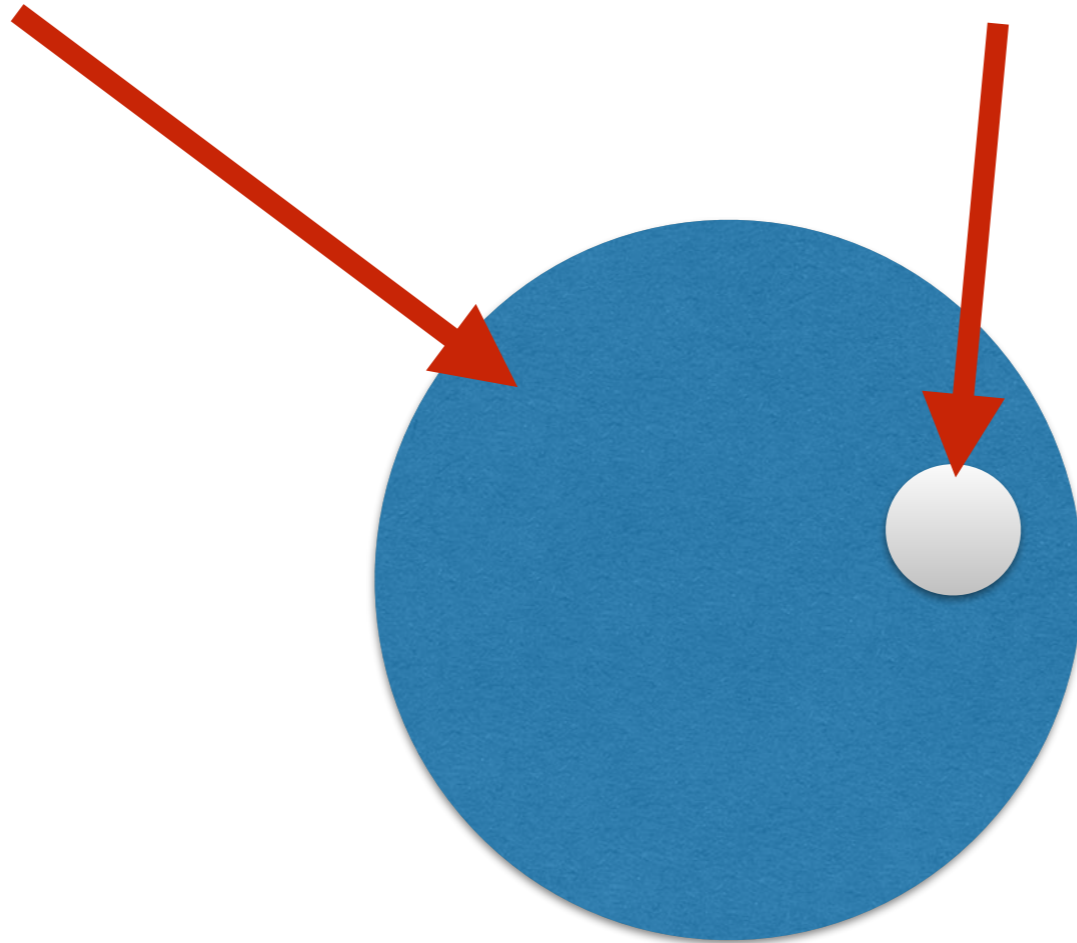
$\text{initTrace}(\text{cloud icon})$



$\forall t:\text{initTrace}(\text{cloud icon}), \text{check}(\text{file icon}, t)$

$\text{initTrace}(\text{cloud icon})$

$\{ t:\text{initTrace}(\text{cloud icon}) \mid \neg\text{check}(\text{file icon}, t) \}$

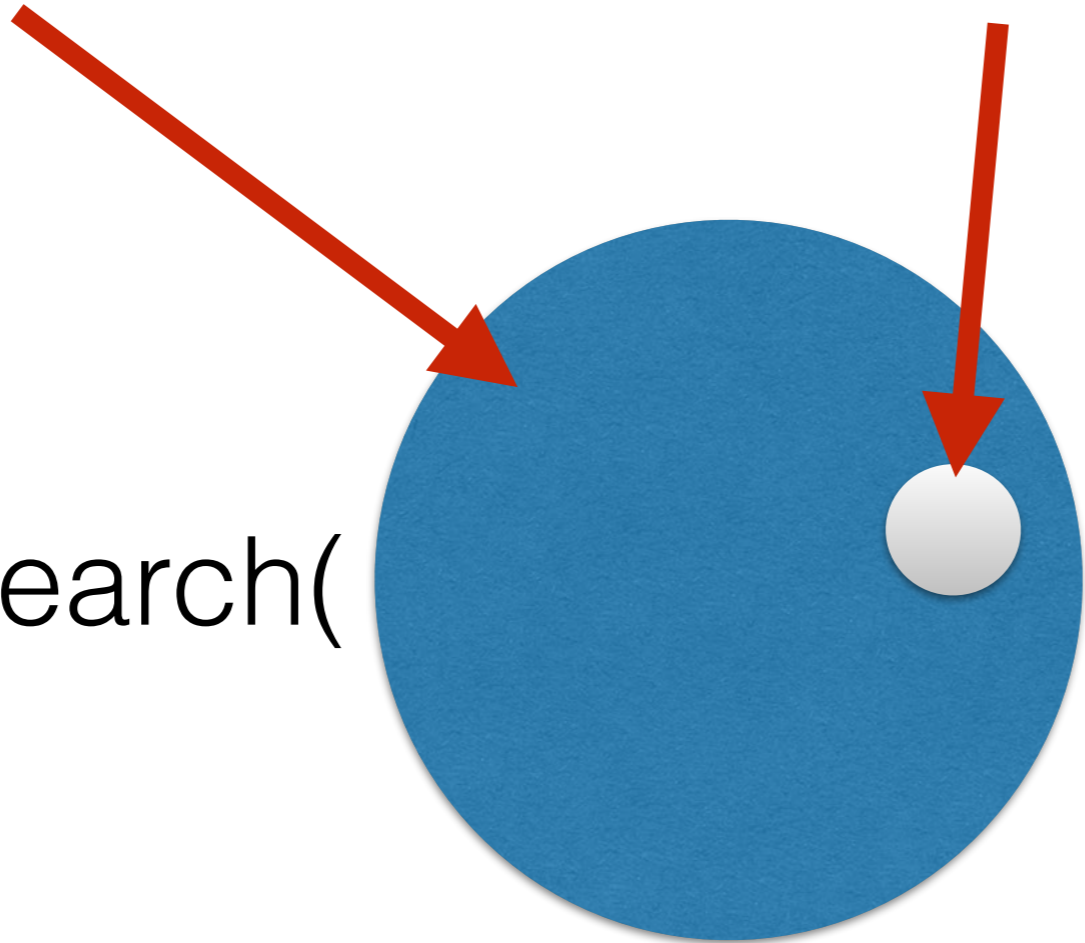


$\forall t:\text{initTrace}(\text{cloud icon}), \text{check}(\text{file icon}, t)$


$\text{initTrace}(\text{cloud icon})$

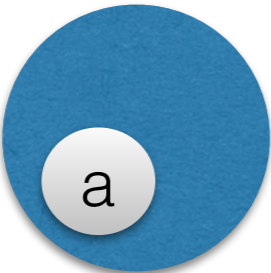
$\{ t:\text{initTrace}(\text{cloud icon}) \mid \neg\text{check}(\text{file icon}, t) \}$




$\text{search}(\text{blue circle}) = \text{None}$




# SpaceSearch Interface

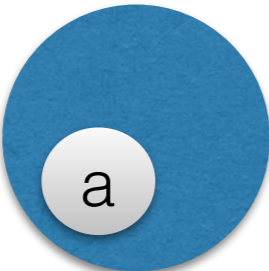
empty = 




singleton(a) = 

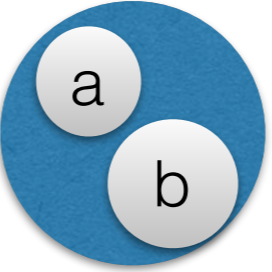
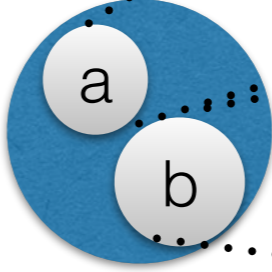
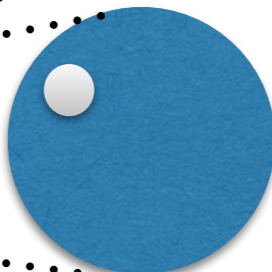

union(, ) = 

# SpaceSearch Interface

empty = 

singleton(a) = 


union(, ) = 


bind(S,f) =  $\bigcup_{x \in S} f(x)$  = bind(, , ) = 

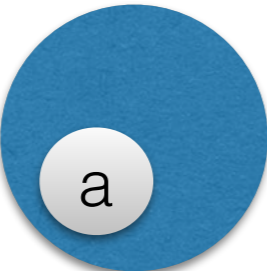
The diagram illustrates the bind operation. It shows a set S containing elements a and b. A function f maps each element x in S to a set f(x). For x=a, f(a) is the set with the irregular shape. For x=b, f(b) is the set with the small dot. The result of bind(S,f) is the union of f(a) and f(b), which is the set containing both the irregular shape and the small dot.







# SpaceSearch Interface

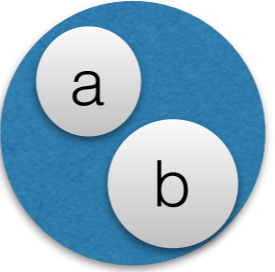
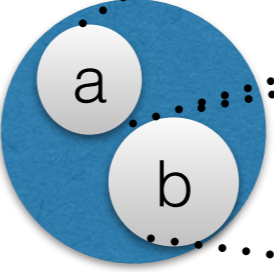
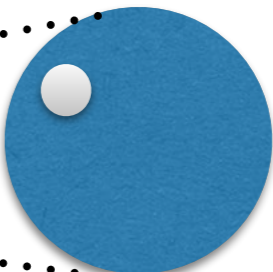

empty = 

search() = None

singleton(a) = 

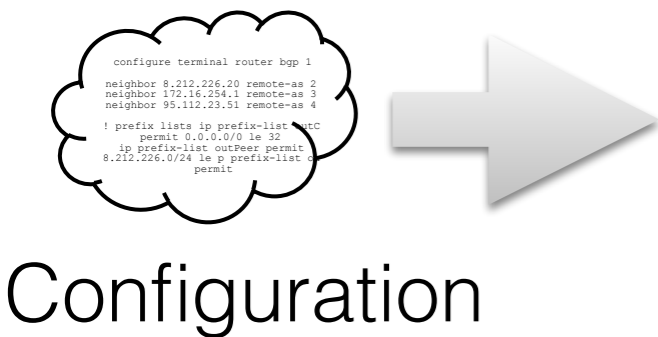
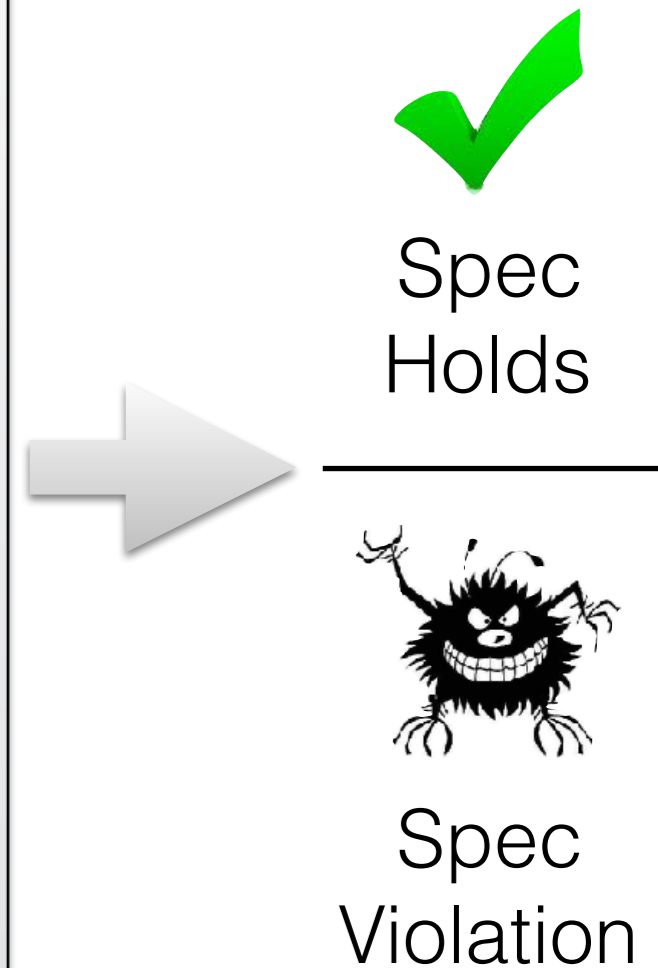
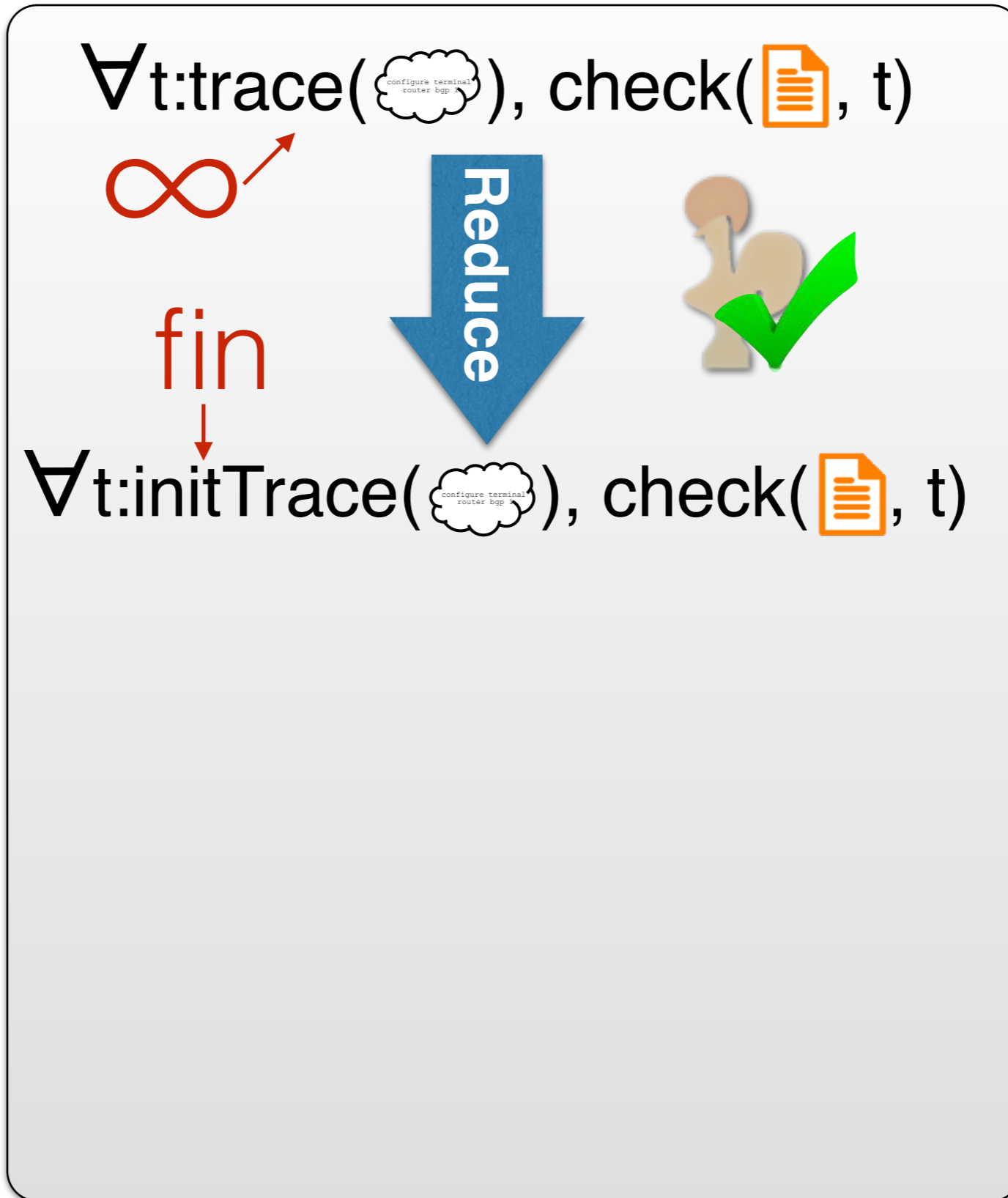
search() = Some a

union(, ) = 

bind(S,f) =  $\bigcup_{x \in S} f(x)$  = bind(, , ) = 

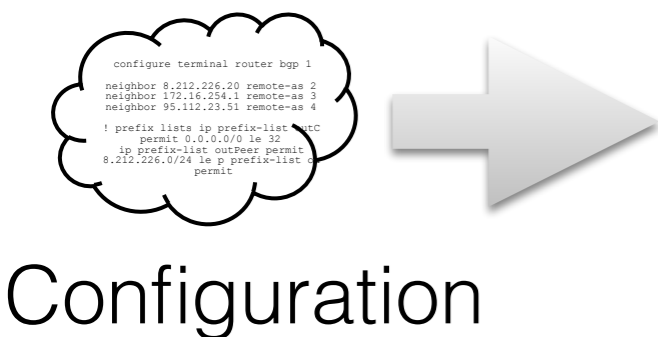
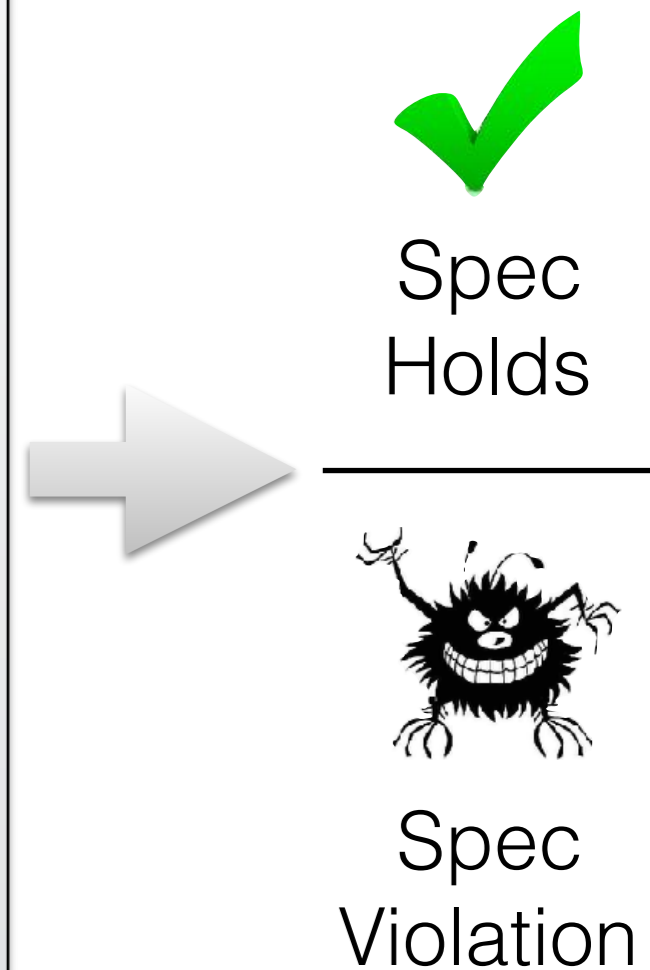
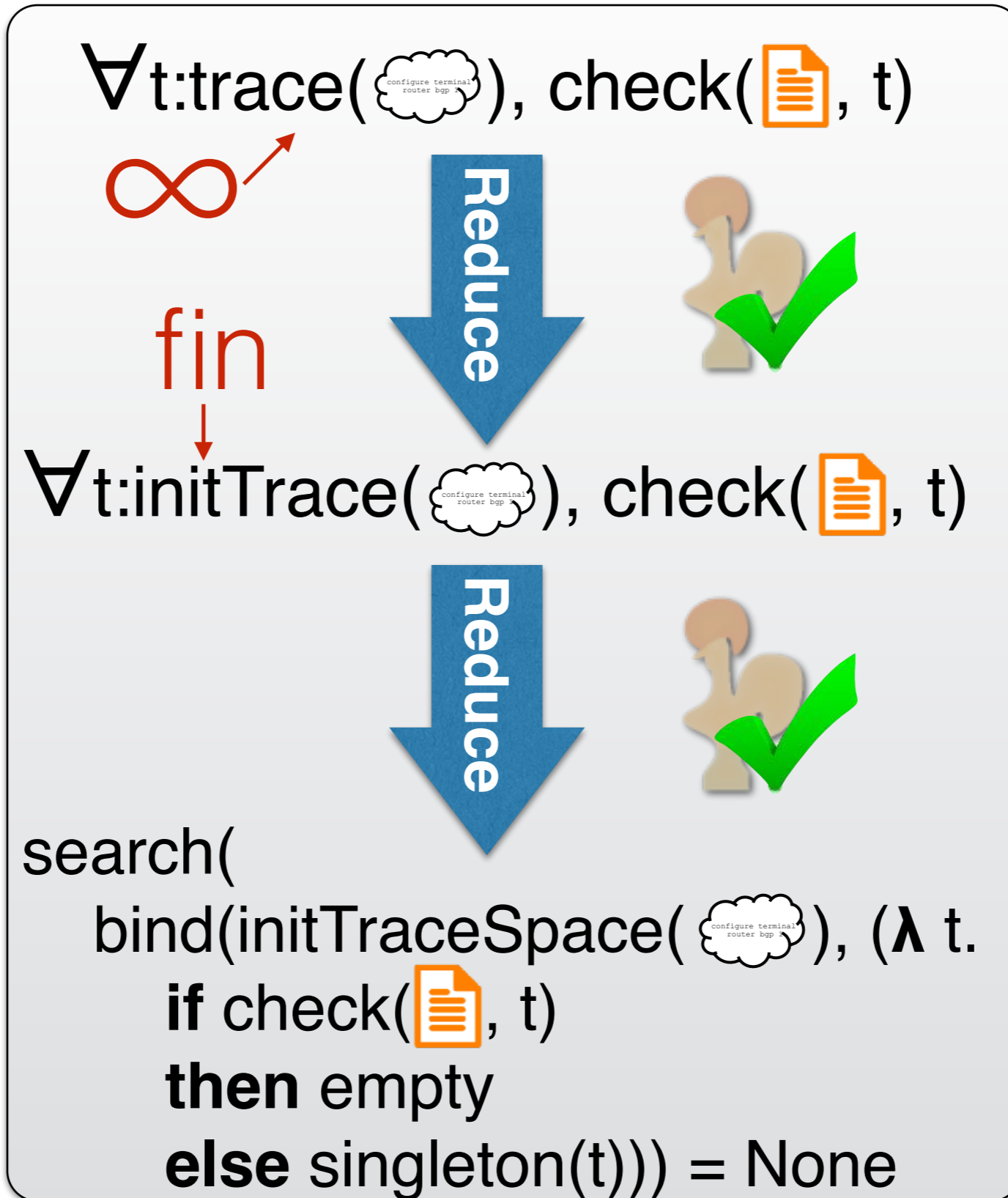
*Diagram details:* Dotted lines connect the 'a' and 'b' elements from the second set to the 'a' and 'b' elements in the first set. A dotted line labeled  $f(a)$  connects the 'a' element from the first set to the white irregular shape in the final result. A dotted line labeled  $f(b)$  connects the 'b' element from the first set to the small white dot in the final result.

# Bagpipe



```
configure terminal router bgp 1
neighbor 8.212.226.20 remote-as 2
neighbor 172.16.254.1 remote-as 3
neighbor 95.112.23.51 remote-as 4
! prefix-lists ip prefix-list out
  permit 0.0.0.0/0 le 32
ip prefix-list outPermit
8.212.226.0/24 le p prefix-list
  permit
```

# Bagpipe



```
configure terminal router bgp 1
neighbor 8.212.226.20 remote-as 2
neighbor 172.16.254.1 remote-as 3
neighbor 95.112.23.51 remote-as 4
! prefix-list ip prefix-list out
  permit 0.0.0.0/0 le 32
ip prefix-list outPermit
8.212.226.0/24 le p prefix-list
  permit
```

# Bagpipe



$\forall t:\text{trace}(\text{configure terminal router bgp}), \text{check}(\text{spec}, t)$



fin



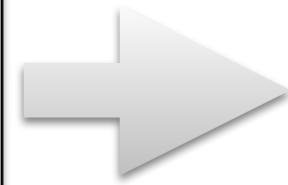
$\forall t:\text{initTrace}(\text{configure terminal router bgp}), \text{check}(\text{spec}, t)$



search(  
bind(initTraceSpace( $\text{configure terminal router bgp}$ ), ( $\lambda t.$   
if check(spec, t)  
then empty  
else singleton(t))) = None



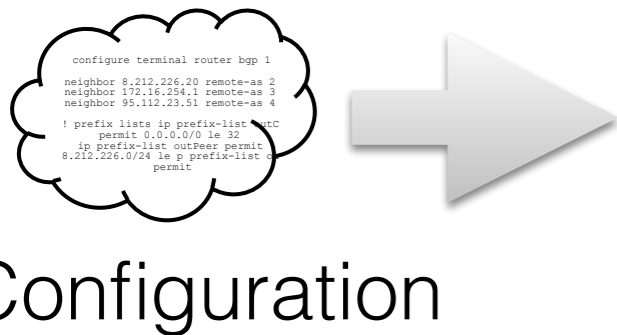
SMT



Spec Holds



Spec Violation



```
configure terminal router bgp 1
neighbor 8.212.226.20 remote-as 2
neighbor 172.16.254.1 remote-as 3
neighbor 95.112.23.51 remote-as 4
! prefix-lists
prefix-list outTree permit 0.0.0.0/0 le 32
ip prefix-list outTree permit 8.212.226.0/24 le 32
prefix-list inTree permit
```

Meet **ROSETTE**

# Meet **ROSETTE**

$$\forall x y. (x \wedge y) \iff \neg(\neg x \vee \neg y)$$

De Morgan's Law

# Meet **ROSETTE**

$\forall x y. (x \wedge y) \iff \neg(\neg x \vee \neg y)$

De Morgan's Law

```
(let ( (x (symbolic-bool))  
      (y (symbolic-bool)) )
```

# Meet **RO**SETTE

$$\forall x y. (x \wedge y) \iff \neg(\neg x \vee \neg y)$$

De Morgan's Law

```
(let ((x (symbolic-bool))  
      (y (symbolic-bool)))  
  (eq? (and x y)  
        (not (or (not x) (not y)))))
```



# Meet **ROSETTE**

$$\forall x y. (x \wedge y) \iff \neg(\neg x \vee \neg y)$$

De Morgan's Law

```
(solve
  (let ((x (symbolic-bool))
        (y (symbolic-bool)))
    (if (eq? (and x y)
            (not (or (not x) (not y))))
        (assert false)
        `counter-example))
```

# Meet **ROSETTE**

$\forall x y. (x \wedge y) \iff \neg(\neg x \vee \neg y)$

De Morgan's Law

```
(solve
  (let ((x (symbolic-bool))
        (y (symbolic-bool)))
    (if (eq? (and x y)
            (not (or (not x) (not y))))
        (assert false)
        `counter-example))
```



type-driven state merging

```
(declare-const x Bool)
(declare-const y Bool)
(define-const a Bool (and x y))
(define-const b Bool (not (or (not x) (not y))))
(assert (not (and (=> a b) (=> b a))))
(check-sat)
```

# SpaceSearch Extraction



=>

**ROSETTE**

union(s,t) => (lambda (v) (if (**symbolic-bool**)  
                                  (s v) (t v)))

empty => (lambda (v) (**assert** false))

search(s) => (**solve** s)

single(a) => (lambda (v) a)

bind(s,f) => (lambda (v) (f (s v) v))

# Bagpipe



$\forall t:\text{trace}(\text{configure terminal router bgp}), \text{check}(\text{Specification}, t)$



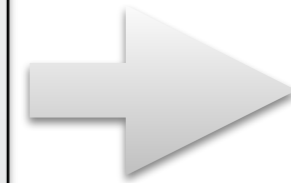
fin



$\forall t:\text{initTrace}(\text{configure terminal router bgp}), \text{check}(\text{Specification}, t)$



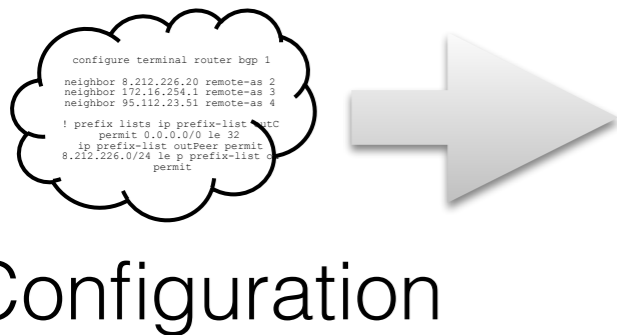
search(  
bind(initTraceSpace( $\text{configure terminal router bgp}$ ), ( $\lambda t.$   
if check( $\text{Specification}$ , t)  
then empty  
else singleton(t))) = None



Spec Holds



Spec Violation



```
configure terminal router bgp 1
neighbor 8.212.226.20 remote-as 2
neighbor 172.16.254.1 remote-as 3
neighbor 95.112.23.51 remote-as 4
! prefix-lists
prefix-list permit 0.0.0.0/0 le 32
ip prefix-list outTree permit
8.212.226.0/24 le p prefix-list
permit
```

# Bagpipe



$\forall t:\text{trace}(\text{configure terminal router bgp}), \text{check}(\text{spec}, t)$



fin



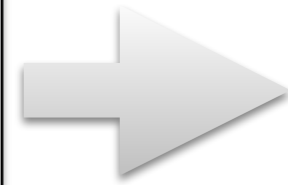
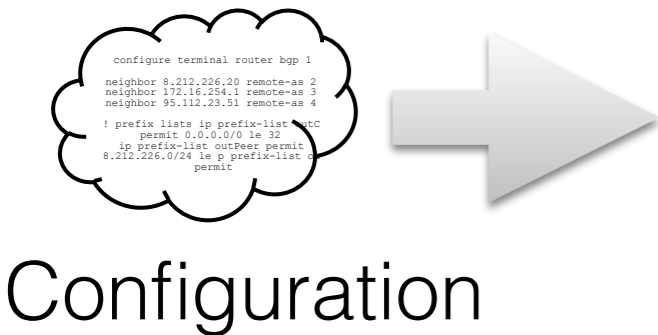
$\forall t:\text{initTrace}(\text{configure terminal router bgp}), \text{check}(\text{spec}, t)$



search(  
 bind(initTraceSpace( $\text{configure terminal router bgp}$ ), ( $\lambda$   
 if check(spec, t)  
 then empty  
 else singleton(t))) = None



**R**OS**ET**T**E**/**S**M**T**



Spec Holds



Spec Violation

# Summary

## SpaceSearch



- Interface & Semantics
- Extraction

# Summary

## SpaceSearch




- Interface & Semantics
- Extraction

More in the paper:


- Infinite Search Spaces
- Other Backends
- Parallelization
- Incrementalization

# Evaluation



SpaceSearch

- Interface & Semantics
- Extraction



Bagpipe

1. BGP Verification



# Evaluation




1. BGP Verification



2. SQL Rewrite

# Evaluation



SpaceSearch

- Interface & Semantics
- Extraction



1. BGP Verification



2. SQL Rewrite

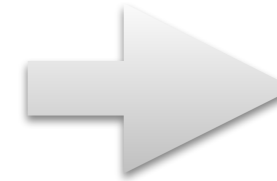
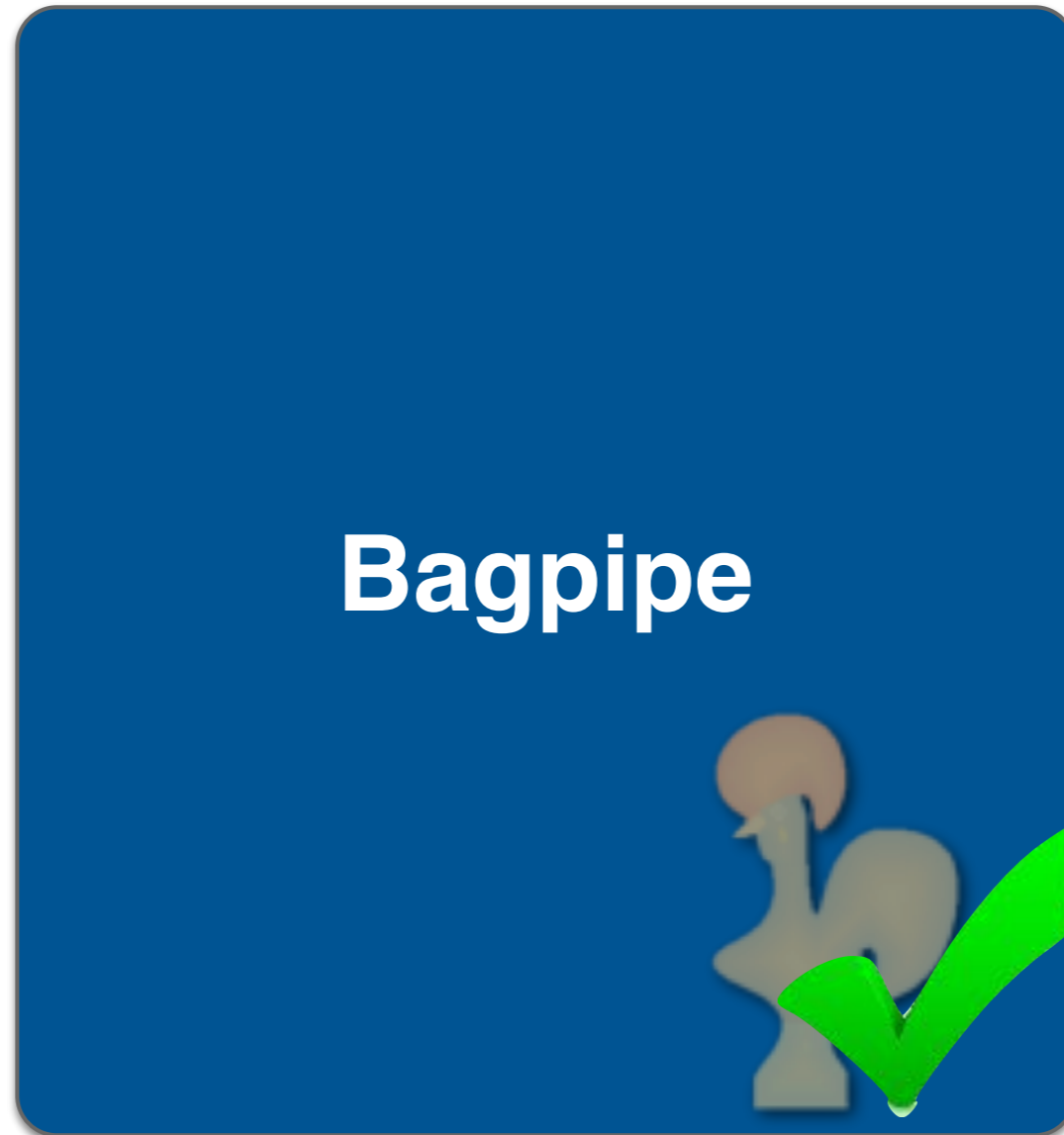
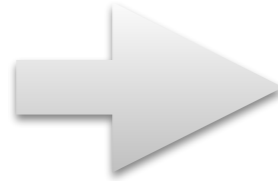


3. x86 Semantics

# 1. BGP Verification



Specification



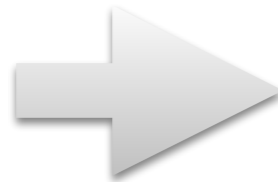
Correct



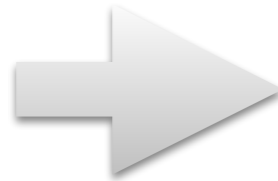
Counter Example

```
configure terminal router bgp 1
neighbor 8.212.226.20 remote-as 2
neighbor 172.16.254.1 remote-as 3
neighbor 95.112.23.51 remote-as 4
! prefix lists ip prefix-list out
  permit 0.0.0.0/0 le 32
ip prefix-list outPermit
8.212.226.0/24 le p prefix-list out
  permit
```

Configuration

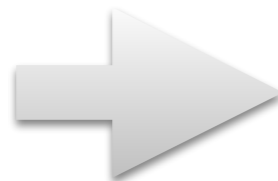


# 1. BGP Verification

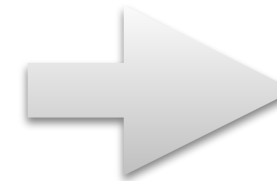


- 10 Juniper Scenarios
- No Martian
- Block To External
- Gao & Rexford

```
configure terminal router bgp 1
neighbor 8.212.226.20 remote-as 2
neighbor 172.16.254.1 remote-as 3
neighbor 95.112.23.51 remote-as 4
! prefix lists ip prefix-list out
  permit 0.0.0.0/0 le 32
ip prefix-list outPermit
8.212.226.0/24 le p prefix-list out
  permit
```



- Internet2 > 100K
- BelWü > 200K
- Selfnet > 50



14 Specs  
Hold

---



19 Spec  
Violations,  
No False  
Positives



## 2. SQL Rewrite Verification

```
SELECT DISTINCT x.name  
FROM Employee AS x, Employee AS y  
WHERE x.name = y.name
```

=

```
SELECT DISTINCT name  
FROM Employee
```



## 2. SQL Rewrite Verification

```
SELECT DISTINCT x.name  
FROM Employee AS x, Employee AS y  
WHERE x.name = y.name
```

=

```
SELECT DISTINCT name  
FROM Employee
```

ConjunctiveQuery:

```
SELECT DISTINCT a,b FROM A,B WHERE a = b, b = c
```



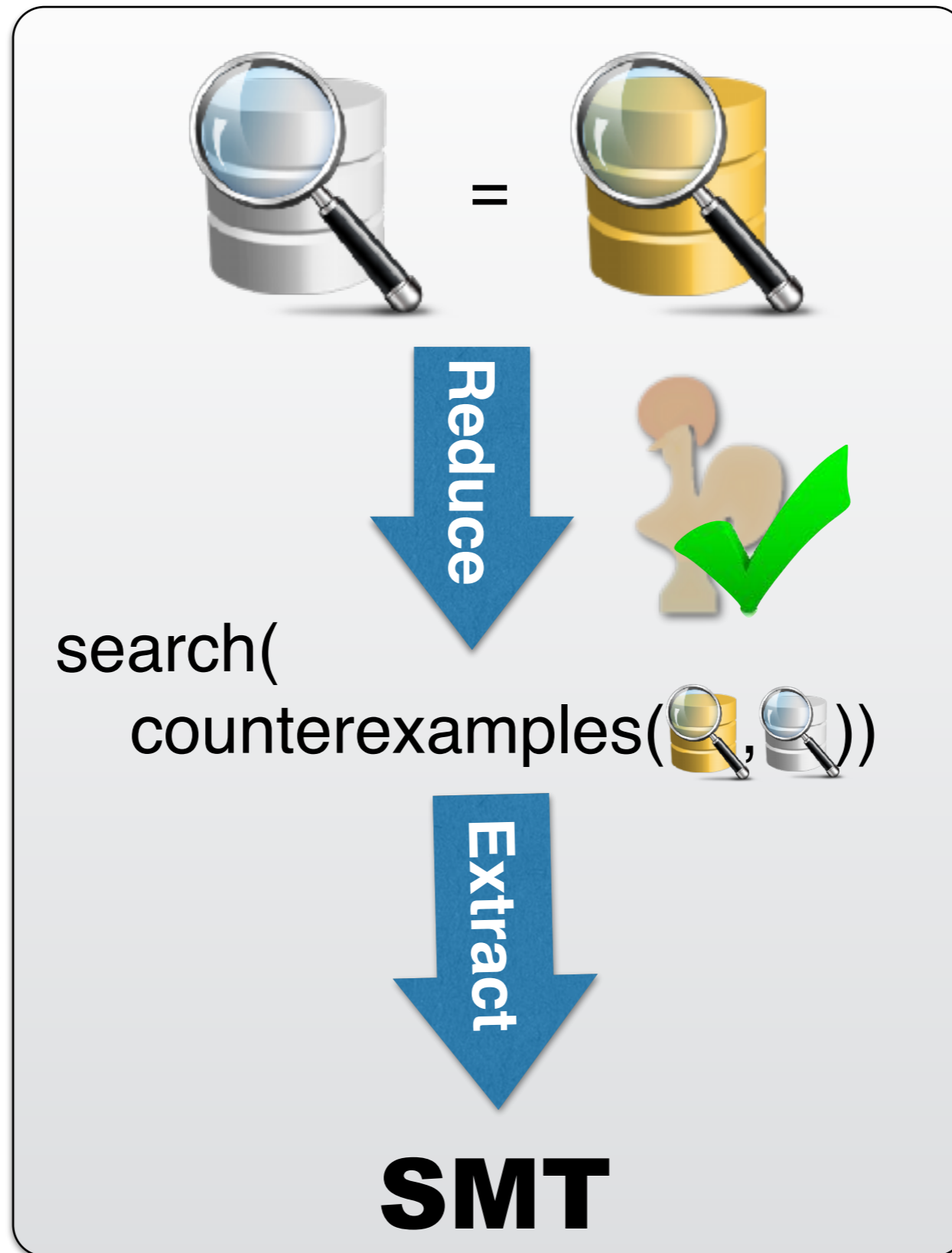
# 2. SQL Rewrite Verification



Query A



Query B



Equal



Counter Example



# 3. x86 Semantics Validation

```
search(  
  bind(int32, ( $\lambda$  x.  
    bind(int32, ( $\lambda$  y.  
      if stoke(ADD x, y) =  
        rocksalt(ADD x, y)  
      then empty  
      else singleton(x,y))))))
```



**SMT**

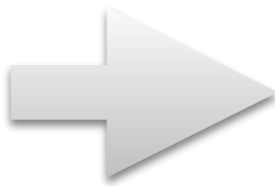


Equal

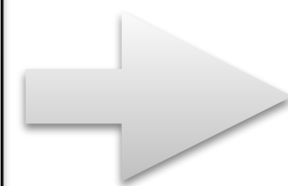


Counter  
Example

**ADD x, y**



Instruction







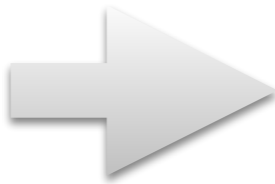
# 3. x86 Semantics Validation

```
search(  
  bind(int32, ( $\lambda$  x.  
    bind(int32, ( $\lambda$  y.  
      if stoke(ADD x, y) =  
        rocksalt(ADD x, y)  
      then empty  
      else singleton(x,y))))))
```

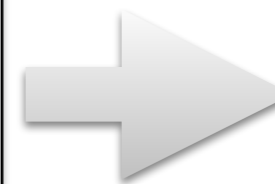


**SMT**

**ADD x, y**



Instruction



Equal



Counter  
Example

**Bugs found:**

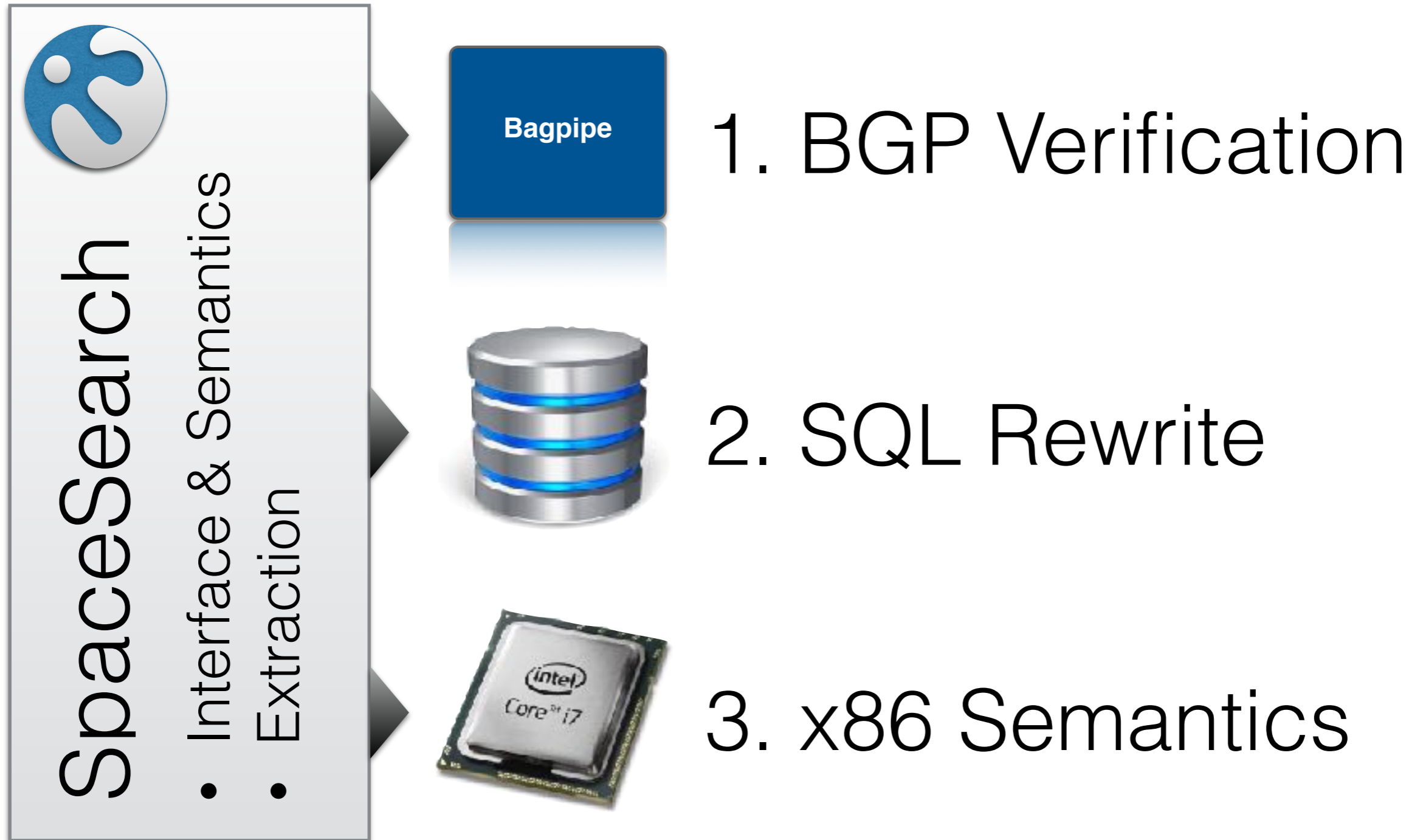
- 7 Rocksalt Bugs
- 1 Stoke Bug

# Related Work

- Solver Aided Languages:  
Rosette Torlak et al. PLDI'14  
Smten Uhler et al. CAV'13
- Solver Aided Tool Verification:  
XCert Tatlock et al. PLDI'10
- Verified SAT Solvers & SAT Tactics:  
Marić TCS'10  
Oe et al. VMCAI'12

# Thank You

[github.com/konne88/SpaceSearch](https://github.com/konne88/SpaceSearch)



**Konstantin  
Weitz**

Steven S.  
Lyubomirsky

Stefan  
Heule

Emina  
Torlak

Michael  
D. Ernst

Zachary  
Tatlock