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

Konstantin Weitz, Steven S. Lyubomirsky, Stefan Heule, Emina Torlak, Michael D. Ernst, Zachary Tatlock
Reduction
Reduction

SMT
SpaceSearch

Reduction

SMT
The Border Gateway Protocol
The Border Gateway Protocol
The Border Gateway Protocol
The Border Gateway Protocol
The Border Gateway Protocol
The Border Gateway Protocol

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
outCustomer permit 0.0.0.0/0 le 32
ip prefix-list outPeer permit
8.212.226.0/24 le p prefix-list
outProvider permit
The Border Gateway Protocol

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
outCustomer permit 0.0.0.0/0 le 32
ip prefix-list outPeer permit
8.212.226.0/24 le p prefix-list
outProvider permit
The Border Gateway Protocol

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
outCustomer permit 0.0.0.0/0 le 32
ip prefix-list outPeer permit
8.212.226.0/24 le p prefix-list
outProvider permit
The Border Gateway Protocol

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
outCustomer permit 0.0.0.0/0 le 32
ip prefix-list outPeer permit
8.212.226.0/24 le p prefix-list
outProvider permit
The Border Gateway Protocol

```
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
outCustomer permit 0.0.0.0/0 le 32
ip prefix-list outPeer permit 8.212.226.0/24 le p
prefix-list
outProvider permit
```
The Border Gateway Protocol

```
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
outCustomer permit 0.0.0.0/0 le 32
ip prefix-list outPeer permit
8.212.226.0/24 le p
ip prefix-list outProvider permit
```
The Border Gateway Protocol

```bash
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 outCustomer permit 0.0.0.0/0 le 32
ip prefix-list outPeer permit
ip prefix-list outProvider permit
```
Bagpipe

Specification

Configuration

Spec Holds

Spec Violation
Bagpipe

Configuration

Specification

Spec Holds

Spec Violation
∀t:trace( ), check( , t)

Spec

Configuration

Configuration

Spec Violation

Spec Holds
∀ \text{trace}, \text{check}, t

Configuration

Specification

Bagpipe

Configuration

Specification

∀ t: \text{trace}(), \text{check}(\text{file}, t)

∞

Spec Violation

Spec Holds
∀t:trace(), check(, t)
∀ t: trace(), check(t)

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 outC
permit 0.0.0.0/0 le 32
ip prefix-list outPeer permit 8.212.226.0/24 le p
∀t:trace(), check(t)

Bagpipe

configure terminal
router bgp 1

∀t:initTrace(), check(t)

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 outC
permit 0.0.0.0/0 le 32
ip prefix-list outPeer permit 8.212.226.0/24 le p

SMT

Specify
Holds

Spec Violation

Configuration

Specification

[OOPSLA'16]
∀ t: initTrace( ), check( , t)
∀ t: initTrace(), check( ), t

initTrace()  { t: initTrace() | ¬check( ) }
∀ t: initTrace( ), check( , t)

initTrace( )

{ t: initTrace( ) | ¬ check( , t) }

search( ) = None
SpaceSearch Interface

empty = 

singleton(a) = a

union(, ) = 
SpaceSearch Interface

empty = 

singleton(a) = a

union(            ,            ) =

bind(S,f) = U\_{x:S} f(x) = bind(            ,                                 ) =
SpaceSearch Interface

empty = \emptyset

singleton(a) = \{a\}

union(\emptyset, \{a\}) = \{a\}

search(\emptyset) = None

search(\{a\}) = Some a

bind(S, f) = \bigcup_{x \in S} f(x) = \bigcup_{x \in S} \bigcup_{y \in f(x)} \{y\}
∀ \( t:trace(\text{Configuration}) \), check(\text{Specification}, t)

∀ \( t:initTrace(\text{Configuration}) \), check(\text{Specification}, t)
∀ t:trace( ), check( , t)

∀ t:initTrace( ), check( , t)

Bagpipe

Configuration

Specification

∀ t:trace( ), check( , t)

∞ → Reduce

fin

∀ t:initTrace( ), check( , t)

search(
  bind(initTraceSpace( ), ( λ t.
    if check( , t)
    then empty
    else singleton(t))))) = None

Spec Holds

Spec Violation
∀ t:trace(), check(t)

∀ t:initTrace(), check(t)

reduce

configuration

specification

∀ t:trace(), check(t)

∀ t:initTrace(), check(t)

search(

bind(initTraceSpace(), (λ t. if check(t) then empty else singleton(t)))) = None

reduce

extract ?

spec holds

spec violation

SMT

Bagpipe
Meet Rosette
∀ x y. (x ∧ y) ⇔ ¬(¬x ∨ ¬y)

De Morgan's Law
Meet ROSETTE

\[ \forall x, y. (x \land y) \iff \neg(x \lor \neg y) \]
De Morgan's Law

(let ((x (symbolic-bool))
     (y (symbolic-bool)))
)
Meet ROSETTE

∀ x y. (x ∧ y) ↔ ¬(¬ x ∨ ¬ y)  De Morgan's Law

(let ((x (symbolic-bool))
  (y (symbolic-bool)))
  (eq? (and x y)
       (not (or (not x) (not y))))
∀ x y. (x ∧ y) ⇔ ¬(¬x ∨ ¬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)))
∀ x y. (x ∧ y) ⇔ ¬(¬ x ∨ ¬ 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

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))
∀ t:trace(), check(    , t)

∀ t:initTrace(), check(    , t)

Bagpipe

∀ t:trace(), check(    , t)

Reduce

fin

∀ t:initTrace(), check(    , t)

Reduce

search(
  bind(initTraceSpace(), (λ t.
    if check(    , t)
    then empty
    else singleton(t))) = None

Spec
Holds

Spec
Violation
∀ \( t : \text{trace}() \), check(\( \square \), \( t \))

∀ \( t : \text{initTrace}() \), check(\( \square \), \( t \))

Bagpipe

\[ \infty \]

fin

search(
bind(initTraceSpace(\( \square \)), (\( \lambda \)\( t \).
if check(\( \square \), \( t \))
then empty
else singleton(\( t \)))) = \text{None} \]

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 outC
permit 0.0.0.0/0 le 32
ip prefix-list outPeer permit 8.212.226.0/24 le p

Specify Hold

\[ \text{Spec Holds} \]

Spec Violation
Summary

SpaceSearch

- Interface & Semantics
- Extraction
Summary

SpaceSearch

- Interface & Semantics
- Extraction

More in the paper:
- Infinite Search Spaces
- Other Backends
- Parallelization
- Incrementalization
Evaluation

1. BGP Verification

SpaceSearch
- Interface & Semantics
- Extraction

Bagpipe
Evaluation

1. BGP Verification
2. SQL Rewrite
Evaluation

1. BGP Verification
2. SQL Rewrite
3. x86 Semantics
1. BGP Verification

**Specification**

**Configuration**

**Correct**

**Counter Example**
1. BGP Verification

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

- Internet2 >100K
- BelWü >200K
- Selfnet >50
2. SQL Rewrite Verification

Query A

Query B

\[
\text{search}(\text{counterexamples}()) = \text{SMT}
\]

Reduce

Extract

Equal

Counter Example
3. x86 Semantics Validation

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

Instruction

ADD x, y

Extract

SMT

Equal

Counter Example
3. x86 Semantics Validation

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

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

1. BGP Verification
2. SQL Rewrite
3. x86 Semantics

Konstantin Weitz
Steven S. Lyubomirsky
Stefan Heule
Emina Torlak
Michael D. Ernst
Zachary Tatlock