

Formalizing Metarouting with PVS

Anduo Wang and Boon Thau Loo

Department of Computer and Information Science, University of Pennsylvania



Introduction

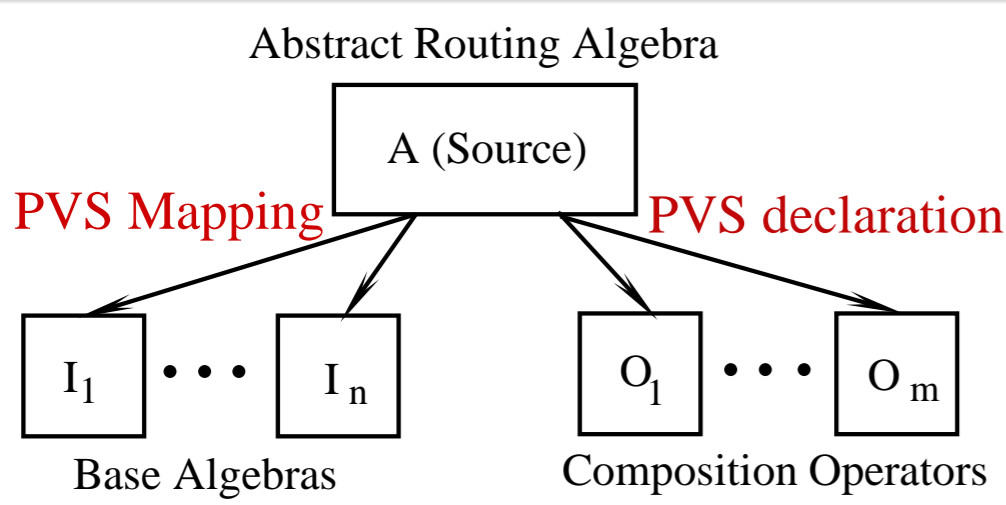
- ▶ Metarouting, algebraic framework for routing protocol
 - ▶ Models BGP systems (today's de facto Internet routing) with convergence guarantee
- ▶ Our contribution: Formalize fragment of metarouting theory in PVS
 - ▶ Heavy and interesting use of PVS theory interpretation: *mapping* and *declaration*
- ▶ Our goal: extend PVS specification logic with metarouting theory
 - ▶ Enable network operator to design BGP in PVS
 - ▶ Free network operator from the tedious low-level and trivial theory consistency checking

Metarouting

Algebraic framework for modeling BGP systems

- ▶ **Abstract routing algebra**, mathematical model:
 - sorts Σ (paths), \mathcal{L} (links)
 - opns $\preceq: \Sigma \times \Sigma \rightarrow \text{bool}$ (preference relation)
 - $\oplus: \mathcal{L} \times \Sigma \rightarrow \Sigma$ (label application function)
 - $\mathcal{O}: \text{subset of } \mathcal{L}$ (origination set)
 - $\phi: \Sigma$ (prohibited path)
 - axioms $\forall \alpha \in \Sigma - \{\phi\} \quad \alpha \preceq \phi$ (*Maximality*)
 - $\forall l \in \mathcal{L} \quad l \oplus \phi = \phi$ (*Absorption*)
 - $\forall l \in \mathcal{L} \forall \alpha \in \Sigma \quad \alpha \preceq l \oplus \alpha$ (*Monotonicity*)
 - $\forall l \in \mathcal{L} \forall \alpha, \beta \in \Sigma \quad \alpha \preceq \beta \Rightarrow l \oplus \alpha \preceq l \oplus \beta$ (*Isotonicity*)
- $A = \langle \Sigma, \preceq, \mathcal{L}, \oplus, \mathcal{O}, \phi \rangle$
- ▶ **Base algebras**, atomic building blocks for shortest-path routing, customer-provider relationship etc
- ▶ **Lexical product** for route selection, composition operator
- ▶ **Isotonicity and Monotonicity**: sufficient conditions for protocol convergence

Overview of PVS theories



- ▶ **A**: uninterpreted source theory `routeAlgebra`
- ▶ **I_i**: interpreted theory instantiated from **A**
- ▶ **O_i**: PVS theory taking routing algebra theories as parameters

Abstract Routing Algebra in PVS

```

routeAlgebra: THEORY
BEGIN
  sig: TYPE+
  label: TYPE+

  injected: [label → bool]
  org: TYPE = {l: label | injected(l)}
  prohibitPath: sig
  labelApply: [label, sig → sig]
  prefRel: [sig, sig → bool]
  eqRel(s1, s2: sig): bool = pre-
  fRel(s1, s2) ∧ prefRel(s2, s1)
  mono(l: label, s: sig): bool = pre-
  fRel(s, labelApply(l, s))
  pref_complete: AXIOM
  ∀ (x, y: sig): prefRel(x, y) ∨ pre-
  fRel(y, x)
  absorption: AXIOM
  ∀ (l: label): labelApply(l, prohibit-
  Path) = prohibitPath
  maximality: AXIOM ∀ (s: sig): pre-
  fRel(s, prohibitPath)
  monotonicity: AXIOM ∀ (l: la-
  bel, s: sig): mono(l, s)
  isotonicity: AXIOM
  ∀ (s1, s2: sig)(l: label):
  prefRel(s1, s2) ⇒
  prefRel(labelApply(l, s1), labelAp-
  ply(l, s2))
END routeAlgebra
    
```

Base Algebra for Shortest Path Routing

```

Abstract Algebra routeAlgebra →
Base Algebra addA
▶ PVS mapping makes instantiations of
uninterpreted types
sig ← upto(m+1)
label ← upto(n)
prohibitPath ← m+1
labelApply ← APPLY
prefRel ← PREF
▶ PVS mapping generates instances of
routeAlgebra axioms as Type Correctness
Conditions (TCCs)
    
```

Shortest Path Routing in PVS

```

▶ routeAlgebra Interpreted Theory: Base
Algebra addA
▶ addA: THEORY
BEGIN
  n: posnat
  m: posnat
  redundant: posnat
  N_M: AXIOM n < m
  LABEL: TYPE = upto(n)
  SIG: TYPE = upto(m+1)
  PREF(s1, s2: SIG): bool = (s1 ≤ s2)
  APPLY(l: LABEL, s: SIG): SIG =
  IF (l + s < m+1)
  THEN (l + s)
  ELSE (m+1)
  ENDIF
  IMPORTING routeAlgebra
  {{sig := SIG, label := LABEL,
  prohibitPath := m+1,
  labelApply(l: LABEL, s: SIG) :=
  APPLY(l, s),
  prefRel(s1, s2: SIG) :=
  (s1 ≤ s2)}}
END addA
    
```

Base Algebra for Provider-Customer, Peer-Peer Guideline

- ▶ For **economical reasons**, ISP reduces use of provider routes, and maximizes availability of customer routes
- ▶ $\Sigma(\text{path}): C/R/P$ (customer/peer/provider path)
- ▶ $\mathcal{L}(\text{link}): c/r/p$ (customer/peer/provider link)
- ▶ \oplus (label application):

\oplus	C	R	P
c	C	C	C
r	R	R	R
p	P	P	P
- ▶ \preceq (preference relation):
 $C \preceq R, R \preceq P, C \preceq P$

Provider-Customer, Peer-Peer Guideline in PVS

```

▶ For simplicity, rename labels and signatures:
c ← 1, r ← 2, p ← 3 and
C ← 1, R ← 2, P ← 3
▶ lpA: THEORY
BEGIN
  SIG: TYPE = upto(3)
  LABEL: TYPE = upto(3)
  IMPORTING routeAlgebra
  {{sig := SIG, label := LABEL,
  labelApply(l: LABEL,
  s: SIG) := l,
  prefRel(s1, s2: SIG) :=
  (s1 ≤ s2),}}
END lpA
    
```

Lexical Product \otimes and Route Selection

- ▶ Lexicographic comparison models route selection
 - ▶ Most important attribute of each route is compared first, if no decision is reached, the next attribute is considered
- ▶ Lexical Product $A \otimes B$ built from existing algebras: A, B
 - ▶ Models a routing protocol with multiple attributes
 - ▶ More important attributes are handled by A , and the less important by B

Lexical Product $A \otimes B$ in PVS

```

lexProduct[A: THEORY routeAlgebra,
B: THEORY routeAlgebra]:
THEORY
BEGIN
  SIG: TYPE = [A.sig, B.sig]
  LABEL: TYPE = [A.label, B.label]
  APPLY(l: LABEL, s: SIG): SIG =
  (A.labelApply(l`1, s`1),
  B.labelApply(l`2, s`2))
  PREF(s1, s2: SIG): bool =
  A.prefRel(s1`1, s2`1) ∨
  (A.eqRel(s1`1, s2`1) ∧
  B.prefRel(s1`2, s2`2))
  IMPORTING routeAlgebra
  {{sig := SIG, label := LABEL,
  labelApply(l: LABEL, s: SIG) :=
  APPLY(l, s),
  prefRel(s1, s2: SIG) :=
  PREF(s1, s2)}}
END lexProduct
    
```

- ▶ PVS **declaration** and **mapping** ensures resulting algebra $A \otimes B$ is a valid routing algebra

A Concrete BGP System

- ▶ Route paths are measured in terms of customer-provider relationship and distance cost
 - ▶ Customer-Provider Peer-Peer guideline must be enforced
 - ▶ Once customer-provider policy is satisfied, ISP wants least-cost (shortest) paths
- ▶ Decompose this BGP system into two sub-components
- ▶ **BGPsystem**: `THEORY=lexProduct[A2, B2]`

Sub-components Instantiated from Base Algebras

```

A2: THEORY =
routeAlgebra
{{sig = lpA.SIG, label = lpA.LABEL,
labelApply(l: lpA.LABEL, s: lpA.SIG)
= l + s, prohibitPath = 4,
prefRel(s1, s2: int) = (s1 ≤ s2)}}
B2: THEORY = routeAlgebra
{{sig = addA.SIG,
label = addA.LABEL,
labelApply(l: addA.LABEL,
s: addA.SIG) = mod(l + s, 16),
prohibitPath = 17,
prefRel(s1, s2: addA.SIG) =
(s1 ≤ s2)}}
    
```

Future Work

- ▶ Provide full support for modeling complex BGP systems via metarouting
 - ▶ Encode more base algebras and composition operators presented in recent metarouting development
- ▶ Relaxed algebra for BGP systems with non-monotonic attributes
 - ▶ MULTI-EXIT-DISCRIMINATOR (MED) expresses router's preference regarding which neighbor to use
 - ▶ NON monotonic attribute: $a \preceq b, b \preceq c, c \preceq a$
 - ▶ Routers in an AS cannot express monotonic ranking