Formalizing Metarouting in PVS

Anduo Wang
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, USA
anduo@seas.upenn.edu

Boon Thau Loo
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, USA
boonloo@seas.upenn.edu

ABSTRACT

In this paper, we extend PVS specification logic with abstract *metarouting* theory to aid the development of complex routing protocol models based on *metarouting*, which is an algebraic framework for specifying routing protocols in a restricted fashion such that the protocol is guaranteed to converge. Our formalization of metarouting theory utilizes the *theory-interpretation* extensions of PVS. Our use of a general purpose theorem prover provides a structured framework for a network designer to incrementally develop and refine their algebraic routing protocol model by starting from various base routing algebras, and composing them into complex algebra models with composition operators. In addition, one can leverage PVS's type checking capability and built-in proof engine to ensure routing model consistency.

1. INTRODUCTION

The Internet today runs on a complex routing protocol called the *Border Gateway Protocol* or *BGP* for short. BGP enables Internet-service providers (ISP) world-wide to exchange reachability information to destinations over the Internet, and simultaneously, each ISP acts as an autonomous system that imposes its own import and export policies on route advertisements exchanged among neighboring ISPs.

Over the past few years, there has been a growing concern on the complexity and fragility of BGP routing. Even when the basic routing protocol converge, conflicting policy decisions among different ISPs have lead to route oscillation and slow convergence. Several empirical studies such as [7] have shown that there are prolonged periods in which the Internet cannot reliably route data packets to specific destinations due to routing errors induced by BGP. In response, the networking community has proposed several Internet architectures and policy mechanisms (e.g. [1]) aimed at addressing these challenges.

Given the proliferation of proposed techniques, there is a growing interest in formal software tools and programming frameworks that can facilitate the design, implementation, and verification of routing protocols. These proposals can be broadly classified as: (1) algebraic and logic frameworks (e.g. [3]) that enable protocol correctness checking in the design phase; (2) runtime debugging platforms that provide mechanisms for runtime verification and distributed replay, and (3) programming frameworks that enable network protocols to be specified, implemented, and in the case of the Mace toolkit, verified via model checking [6].

In this paper, we extend PVS specification logic with abstract *metarouting* theory [3] to aid the development of complex routing protocol models based on *metarouting*, which is an algebraic framework for specifying routing protocols in a restricted fashion such that the protocol is guaranteed to converge. Using the *theory-interpretation* [8] extensions of the PVS theorem prover, we formalize in PVS a variety of metarouting algebra instances and demonstrate that an interactive theorem prover is suitable for modeling the complicated BGP system using the metarouting theory developed in PVS.

The main benefits of formalizing metarouting within a mechanized theorem prover are as follows. First, the network designer can now focus on high-level protocol design and the conceptual decomposition of the BGP system, and shift the low level details of ensuring consistency of the derived protocol model with respect to metarouting theory to the PVS type checker. Second, the PVS proof engine handles most of the proof effort (via the top-level strategy *grind* and other built-in type checking capabilities), and therefore frees the network operator from the trivial and tedious proof necessary to ensure the convergence of their BGP algebra model. In the long run, we believe that our framework will also result in the support of relaxed algebra models, which allow a wider range of well-behaved convergent component protocols to be supported compared to the restrictions imposed by metarouting.

2. BACKGROUND

2.1 Internet Routing

The Internet can be viewed as a network of *Autonomous Systems* (AS) each administrated by an *Internet Server Provider* (ISP). The *routing protocol* is executed on all ASes in order to compute reachability information. Given a destination address, each packet sent by a source is forwarded by each intermediate node to the next neighboring node along the best path computed by the routing protocol.

In particular, within an AS, the ISP runs its own class of routing protocols called the *Internal Gateway Protocol* (IGP), whereas between ASes , the class of protocols used are called the *External Gateway Protocol* (EGP). EGP en-

ables routing across AS administration borders by including mechanism for *policy-based routing*. The role of policy routing is to allow ISPs to influence route decisions for economical or political concerns, and the basic mechanism used is to decide which routes to accept from neighbors (*import policies*), and which routes to advertise to other neighbors (*export policies*).

2.2 Metarouting

The Internet uses the *Border Gateway Protocol* (BGP) as its de facto routing protocol. This protocol is a combination of the IGP/EGP protocol described above. Metarouting [3] is first proposed to extend the use of routing algebra to BGP design and specification. Metarouting enables the construction of a complicated BGP system model from a set of pre-defined base routing algebras and composition operators. Prior to metarouting, Griffin *et al.*first proposed combinatorial models for BGP [2, 4] to aid the static analysis of convergence of routing protocols. Later a *Routing Algebra Framework* was proposed by Sobrinho [9, 10] to provide the rigorous semantics for the design and specification of routing protocols. Sobrinho uses various algebra instances to represent possible routing protocols and policy guidelines. Sobrinho further identifies and proves monotonicity as a sufficient condition for protocol convergence. Meta-routing builds upon these two earlier pieces of work. In the rest of the section, we provide a short overview of metarouting.

First, metarouting adopts the use of routing algebra as the mathematical model for routing. An abstract routing algebra is a tuple A: $A = \langle \Sigma, \preceq, \mathcal{L}, \oplus, \mathcal{O}, \phi \rangle$. Here Σ is the set of *signatures* used to describe paths in the network totally ordered by preference relation $\preceq$. Intuitively, the preference relation is used by a routing protocol to optimize path cost; $\mathcal{L}$ is a set of $labels$ describing links between immediate neighbors. Note that labels may denote complicated policies associated with the corresponding link; $\oplus$ is a mapping from $\mathcal{L} \times \Sigma$ to Σ, which is the *label application operation* that generates new paths by combining existing paths and adjacent links; And $\mathcal{O}$ is a subset of Σ called *origination* that represents the initial routes stored at network nodes; Finally ϕ is a special element in Σ denoting the prohibited path. The semantics of routing algebra is given by the following axioms:

Maximality	$\forall_{\alpha \in \Sigma - \{\phi\}} \quad \alpha \preceq \phi$
Absorption	$\forall_{l \in \mathcal{L}} \quad l \oplus \phi = \phi$
Monotonicity	$\forall_{l \in \mathcal{L}} \forall_{\alpha \in \Sigma} \quad \alpha \preceq l \oplus \alpha$
Isotonicity	$\forall_{l \in \mathcal{L}} \forall_{\alpha, \beta \in \Sigma} \quad \alpha \preceq \beta \implies l \oplus \alpha \preceq l \oplus \beta$

Maximality and **Absorption** are straightforward properties of the prohibited path ϕ, stating that any other paths are always preferred over ϕ, and that extending the un-usable path ϕ with any usable link would still result in prohibited path. On the other hand, **Monotonicity** and **Isotonicity** are two non-trivial properties that ensure network *convergence*[1] of a routing protocol modeled by the routing algebra.

Furthermore, based on the abstract routing algebra, metarouting identifies a set of atomic (base) algebras such as *ADD(n,m)* and *LP(n)*, and composition operators such as *Lexical Product* $\otimes$ and *Scaled Product* $\odot$ as the building blocks for more complicated routing algebras. This paper presents the incremental development of metarouting abstract algebras and the use of such abstract theory to build concrete BGP systems.

Unlike previous combinatorial models [2, 4], metarouting identifies and proves that the properties of *monotonicity* and *isotonicity* are sufficient conditions for network convergence. Convergence verification of BGP systems are then reduced to proofs of monotonicity and isotonicity of the related routing algebra, whereas in the analysis of BGP systems using previous combinatorial models, the proof requires genuine insights into the models themselves.

Despite its advantages, metarouting is fairly restricted in two ways. First, it cannot represent all protocols that converge. Second, it places the burden on network designers to write algebras and composition operators correctly. Our work aims to address these two limitations by using PVS to provide a framework for expressing routing algebras and their operators correctly, and then flexibly reason about the convergence properties of these protocols even when the sufficient conditions are violated. One should view our paper as providing the initial building blocks and methodology for interesting explorations elaborated in Section 5.

3. BASIC APPROACH

This section describes the basic technique of embedding metarouting in PVS. In particular, this paper presents the development of metarouting in PVS using its extensions on *theory interpretation* [8].

The basic approach is to encode metarouting algebraic objects in PVS's type system. It involves formalization of abstract routing algebra theory A, the set of atomic algebra instances of A, and composition operator $\otimes$.

First of all, the abstract routing algebra structure $A = \langle \Sigma, \preceq, \mathcal{L}, \oplus, \mathcal{O}, \phi \rangle$ is formalized as an uninterpreted abstract (source) theory in PVS, as described in [11, 5, 8]. The basic idea is to use *types* to denote the sets of objects $\Sigma, \mathcal{L}, \mathcal{O}$. Accordingly, the special element ϕ denoting prohibited path is expressed as an uninterpreted constant of type Σ. And the preference relation $\preceq$ and the label application operation are denoted by functions. The PVS theory for abstract route algebra A is given as follows:

```
routeAlgebra: THEORY
  BEGIN
  sig: TYPE+
  prefRel: [sig, sig -> bool]
  label: TYPE+
  labelApply: [label, sig -> sig]
  prohibitPath: sig
  initialS: [sig -> bool]
  org: TYPE = s: sig | initialS (s)
  END routeAlgebra
```

Here uninterpreted types `sig` and `label` denote sets Σ and $\mathcal{L}$, and `org` denoting initial route set $\mathcal{O}$ is made subtype

[1] A network routing protocol converges when all routing tables can be computed to a distributed fixpoint given a stable network, and when any links are updated, these routing tables can be incrementally recomputed similarly to a fixpoint.

of Σ via auxiliary predicate `initialS` which decides if a path falls into the initial routes. Finally ϕ is made constant `prohibitPath` of type `sig`.

Semantics of abstract routing algebra A are given by the following axiomatic specification:

```
monotonicity: AXIOM
  FORALL (l: label, s: sig): mono(l, s)

isotonicity: AXIOM
  FORALL (l: label, s1, sig, s2: sig):
    prefRel(s1,s2) =>
      prefRel(labelApply (l,s1),labelApply(l,s2))

maximality: AXIOM
  FORALL (s: sig): prefRel (s, prohibitPath)

absorption: AXIOM
  FORALL (l: label):
    labelApply (l, prohibitPath) = prohibitPath
```

Where `eqRel` (`eqRel` will be used later) and `mono` are defined by the following auxiliary predicates:

```
eqRel (s1, s2: sig): bool =
  prefRel (s1, s2) and prefRel (s2, s1)
mono(l: label, s: sig): bool =
  prefRel (s, labelApply (l, s))
```

Note that this abstract routing theory A then stands for all possible routing algebra instances. We also observed that PVS parametric theories offer an alternative to define abstract algebra, sketched as follows:

```
routeAlgebra[sig: TYPE+,
             prefRel: [sig, sig -> bool],
             label:TYPE+,
             labelApply: [label, sig -> sig]]
 BEGIN
  prohibitPath: sig
  initialL: [label -> bool]
  org: TYPE+ = l: label | initialL (l)
  ...
 END routeAlgebra
```

In the rest of this paper, to exploit PVS's theory interpretation mechanism, we will use of the first uninterpreted theory representation. To encode the basic building blocks of metarouting: atomic routing algebras and composition operators, we utilize two features provided by the PVS theory interpretation [8] extensions: *mapping* and *declaration*. With the *mapping* mechanism, a general (source) theory is instantiated to an interpretation (target) theory. On the other hand, the *theory declaration* mechanism takes PVS theories as parameters, and therefore, unlike mapping, can build a theory from multiple source structures.

Figure 1 shows our basic two-step approach to formalize metarouting building blocks. First, we utilize abstract routing algebra theory `A` developed above as a source theory, applying PVS's *mapping* mechanism to this source theory to yield the set of interpretation theories `Ii` for atomic routing algebras I_i. Second, by applying PVS's theory *declaration* mechanism, we encode composition operator O_i as PVS theories taking routing algebra as parameters, which can be further instantiated to yield the resulting compositional routing algebra `Oi`.

The main benefit of this approach is that the semantics (axioms) of source routing theory `A` are enforced automatically in all target theories `Ii` and `Oi`. This ensures that all atomic routing algebra instances are valid routing algebras and that all composition operators are closed under abstract routing algebra (i.e. any compositional routing algebras that can be derived using operators `Oi` are guaranteed to be routing algebras as defined by the abstract routing algebra theory). The detailed formalization of metarouting building blocks using PVS theory interpretation is presented in the next section.

4. COMPOSITIONAL ROUTING ALGEBRA

This section presents the formalization of metarouting building blocks by stepping through atomic routing algebras *addA(n,m)* and *cpA(n)*; as well as composition operator lexical product $\otimes$.

4.1 Atomic Routing Algebra Instance

Shortest Path Routing.

The first simple routing algebra *addA(n,m)* describes shortest path routing. The labels/signatures can be thought of as the distance costs associated with the corresponding links/paths. Note that in practice, costs of valid links/paths have an upper bound, and links/paths with higher cost are considered prohibited. We use PVS theory `addA` to capture algebra *addA(n,m)* as follows:

```
addA: THEORY
 BEGIN
  n, m: posnat
  N_M: AXIOM n < m
  LABEL: TYPE = upto(n)
  SIG: TYPE = upto(m + 1)
  ...
 END addA
```

Here `n, m` are the uninterpreted constants denoting link/path cost bounds. The preference relation $\preceq$ over signatures `SIG` are then simply interpreted as the normal $\leq$ relation over natural numbers, indicating that a low-cost path is preferred over high-cost path. The label application operation $\oplus$ can be interpreted as a function that computes the cost of the new path obtained from a sub-path and the adjacent link, where the cost of the new path is simply the normal addition of that of the sub-path and link. In PVS, we write as follows:

```
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
```

Note that in the definition of label application function `APPLY`, `m+1` is used as the value of prohibited path. This is directly defined using PVS mapping of abstract algebra `routeAlgebra` in the following `IMPORTING` clause:

```
IMPORTING
  routeAlgebra{{sig := SIG,
                label := LABEL,
                prohibitPath := m + 1,
                labelApply(l:LABEL, s:SIG)
                           := APPLY(l,s),
                prefRel(s1, s2: SIG)
                           := PREF (s1, s2)}}
```

Recall that a routing algebra consists of a set of signatures `sig`, labels `label`, preference relations `prefRel` over signatures, and label application functions `labelApply`. Here,

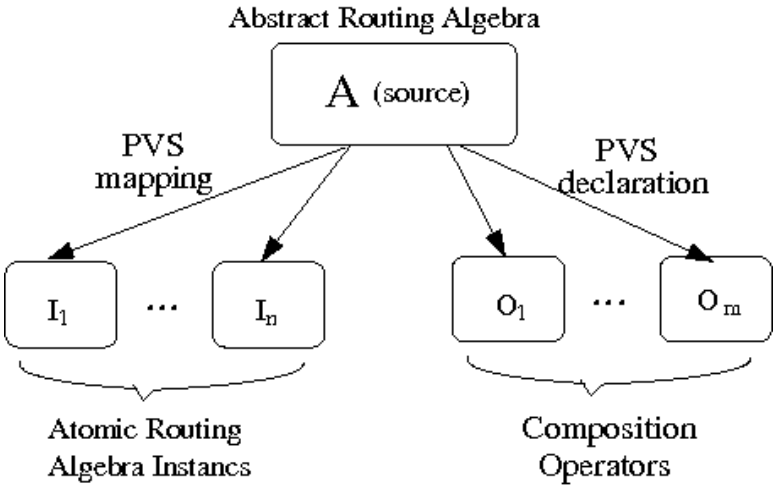

Figure 1: Overview of PVS Theories

theory `addA` imports the uninterpreted abstract algebra theory `routeAlgebra`, and makes the following instantiations:

$$\begin{aligned} \texttt{sig} &\leftarrow \texttt{upto(m+1)} \\ \texttt{label} &\leftarrow \texttt{upto(n)} \\ \texttt{prohibitPath} &\leftarrow \texttt{m+1} \\ \texttt{labelApply} &\leftarrow \texttt{APPLY} \\ \texttt{prefRel} &\leftarrow \texttt{PREF} \end{aligned}$$

The corresponding instances of `routeAlgebra` axioms defining the semantics of routing algebra are proof obligations called type correctness conditions (*TCCs*). For example, monotonicity axiom is instantiated and denoted by the following automatically generated TCC:

```
IMP_A_monotonicity_TCC1: OBLIGATION
  FORALL (l: LABEL, s: SIG): mono(l, s)
```

All of the TCCs are automatically discharged by either the default TCC proof strategy or high-level strategy `grind`.

So far, we have established the encoding of shortest path algebra in PVS by providing mappings for uninterpreted types in the source theory `routeAlgebra` into the target theory (interpreting) `addA`. We observe that even in this simple example, PVS significantly reduces manual effort ensuring consistency, generating proof obligations and enabling the user to focus on high-level mapping for shortest path routing.

Customer-Provider and Peer-Peer Relationship.

We provide another example of base/atomic algebra *cpA(n)* that captures the policy guideline regarding the economic relationship between ASes. Customer-Provider and Peer-Peer relationships between ASes are prevalent in today's Internet. A common policy guideline to help BGP convergence is to always prefer customer-routes to peers or providers routes.

More specifically, in the algebra $cpA = \langle \Sigma, \preceq, \mathcal{L}, \oplus, \mathcal{O}, \phi \rangle$, the signature set can take three values $C/R/P$, representing customer/peer/provider routes respectively (i.e. routes advertised by a node's customer, peer, or provider). Accordingly, labels can take values $c/r/p$, representing customer/peer/provider link (i.e. links to customer/peer/provider).

The preference relation over signatures is given by: $C \preceq R, \quad R \preceq P, \quad C \preceq P$. Intuitively this relation means, a customer route is always preferred over a peer and provider route, and a peer route is preferred over a provider route. The intuition is that each ISP enforces the policy to reduce the use of provider routes, while maximizing availability and use of its customer routes.

The complete definition of the label application operation $\oplus$ is given by the following table:

$\oplus$	C	R	P
c	C	C	C
r	R	R	R
p	P	P	P

For example the first line $c \oplus (C/R/P) = C$ can be read as a customer/peer/provider path extended by a customer link results in a customer path, hence has the highest priority of all available paths.

For simplicity, rename labels and signatures as follows: $c \leftarrow 1, r \leftarrow 2, p \leftarrow 3$ and $C \leftarrow 1, R \leftarrow 2, P \leftarrow 3$. This renaming enables the preference relation to be expressed as normal $\leq$ over natural number. Similar to the algebra for shortest path, `cpA` can be encoded using PVS mapping as follows:

```
cpA: THEORY
  BEGIN
  SIG: TYPE = x: posnat | x<=3
  LABEL: TYPE = x: posnat | x<=3
  APPLY (l: LABEL, s: SIG): SIG = l
  IMPORTING
    routeAlgebra{{sig := SIG,
                  label := LABEL,
                  labelApply(l:LABEL, s:SIG)
                             := APPLY (l,s),
                  prohibitPath := c+1}}
  END cpA
```

As in the case of shortest paths, all the TCCs enforcing routing algebra axioms (for example, monotonicity) are au-

tomatically discharged. This is consistent with the intuition that customer-provider policy does help BGP convergence.

4.2 Lexical Product and Route Selection

This section presents development of lexical product $\otimes$, a composition operator that enables construction of routing algebra from atomic algebra described in section 4.1. It is particularly useful in modeling route selection in BGP system where multiple attributes are involved.

Consider product algebras $A \otimes B$ constructed from two route algebra `A, B`, where the parameter theories `A` and `B` model two attributes `a` and `b` respectively. First define the signature and label of $A \otimes B$ as product of that from `A` and `B` in PVS as:

```
lexProduct[A, B: THEORY routeAlgebra]: THEORY
  BEGIN
  SIG: TYPE = [A.sig, B.sig]
  LABEL: TYPE = [A.label, B.label]
  ...
  END lexProduct
```

Here the first component of signature/label comes from `A` and the second component comes from `B`. And a natural interpretation of label application function over path and label is given by the following product in PVS:

```
APPLY(l:LABEL,s:SIG):SIG =
(A.labelApply(l`1,s`1),B.labelApply(l`2,s`2))
```

Here the two components invoke the corresponding label application functions defined in theory `A` and `B` respectively.

Next consider the preference relation over $A \otimes B$ that in PVS as follows:

```
PREF(s1,s2:SIG):bool =
 A.prefRel(s1`1, s2`1) OR
 (A.eqRel(s1`1,s2`1) AND B.prefRel(s1`2,s2`2))
```

The above definition is particularly interesting because it models the route selection process in BGP system. This preference relation reads as: a path with two attributes `a` and `b` represented by signature `s1` is considered better than a path denoted by `s2` given one of the two following conditions: (1) first component `s1`1` of `s1` is better than the first component `s2`1` of `s2`, as defined in algebra `A`; or (2) if the first component of `s1` and `s2` are equally good, but `s1` is better than `s2` with respect to the second component, as described in algebra `B`. This lexicographic comparison captures the route selection process, which is a major part for any BGP system with multiple attributes. Intuitively, in selecting a route towards a given destination, the router compares all its possible paths towards that destination by going through a comparison list, checking one attribute at a time, selecting the best path based on attributes ordering. The router goes down the list and compares the next attribute only if the attributes seen in previous steps are equally good.

As before, we can now instantiate route algebra theories and corresponding sets of axioms as follows:

```
IMPORTING
  routeAlgebra{{sig := SIG,
                label := LABEL,
                labelApply(l:LABEL,s:SIG)
                           := APPLY(l,s),
                prefRel(s1, s2: SIG)
                       := PREF(s1, s2)}}
```

Again, PVS automatically generate and prove all the type checking conditions.

4.3 A Concrete First Example

This section presents an concrete example routing protocol algebra built from metarouting atomic algebras and composition operators developed in previous sections. We demonstrate the ease of applying abstract metarouting theory to concrete example algebra in PVS. In particular, we highlight the intuitive networking interpretation in practice.

Consider a simple BGP system where the route paths are measured in terms of customer-provider relationship and distance cost. For all possible routes reaching a given destination, a route path going through customers and peers is preferred to path going through providers; and a route go through peers is preferred to those through providers. Once this customer-provider policy is enforced, the ISP is concerned with distance cost with respect to each path. For the same types of paths, the ISP will choose the shortest path with lowest cost.

In the top level, this BGP system can be decomposed into two sub-components: customer-provider component and the shortest path component developed in section 4.1. Because the customer-provider relationship has higher-priority over the distance cost attribute, it can be naturally implemented by construction using lexical product, as shown in the following PVS code:

```
firstExample: THEORY
 BEGIN
 IMPORTING AlgebraInstance, lexProduct
 firstAlgebra: THEORY = lexProduct[A2,B2]
 END firstExample
```

Here `firstAlgebra` is defined to be the concrete algebra modeling this BGP system. It is constructed from customer-provider component algebra `A2` and shortest path algebra `B2` by applying lexical product, where `A2` and `B2` are defined in the imported theory `AlgebraInstance`. First, we show the definition of `A2` that enforces ISP customer-provider policy simply as an instance of `cpA`, where the uninterpreted constant `c` is mapped to 3.

```
AlgebraInstance: THEORY
 BEGIN
  IMPORTING cpA{{ c := 3 }}
  A2:THEORY =
    routeAlgebra{{
      sig = cpA.SIG,
      label = cpA.LABEL,
      labelApply(l:cpA.LABEL,s:cpA.SIG)
                          = mod(l+s,c),
      prohibitPath = c + 1,
      prefRel(s1,s2:cpA.SIG) = (s1<=s2)}}
  ...
 END AlgebraInstance
```

Likewise, concrete algebra `B2` for shortest path can be defined in terms of `addA` as follows:

```
IMPORTING addA{{n:= 16, m:=16}}
B2:THEORY =
  routeAlgebra{{
    sig = addA.SIG,
    label = addA.LABEL,
    labelApply(l:int,s:int) = l+s,
    prohibitPath=16,prefRel(s1,s2:int)
                          =(s1<=s2)}}
```

Where uninterpreted bounds on signature/labels `m/n` in `addA` are mapped to 16, which is the actual value used in distance

vector protocol practice. Finally, by type checking, PVS automatically figures out all type correctness conditions to ensure consistency. All of the TCCs are discharged with default/high-level proof procedure in one step. This ensures the BGP system we derived from atomic algebras `addA`, `cpA` by using composition operator $\otimes$ are indeed a valid routing algebra that is guaranteed to converge.

In summary, we observe that by incorporating metarouting abstract theory, a non-specialized standard proof assistant like PVS, can be used to specify a specific routing protocol instance with great ease. And the routing algebra semantics is enforced by proof obligations (TCCs) automatically generated in PVS, all of which can be discharged by either PVS default TCC proof strategy or high-level strategy `grind` in one step!

5. FUTURE WORK

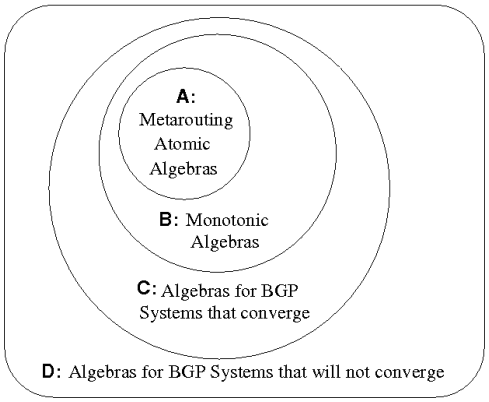

Figure 2: Classification of BGP algebras

A straightforward application of the specification technique we explored in this paper is to construct incrementally in PVS the routing algebraic model for complicated BGP systems by using the base algebra blocks and composition operators we developed in this paper, and the resulting algebraic model checked in PVS is then used as part of design document to derive the real BGP implementation. To achieve full set support for the modeling of BGP system via metarouting, we plan to encode in PVS more base routing algebras, such as `TAG` which is critical in the modeling of complicated routing policies, and more composition operators, such as scoped product, which models a BGP system running in and between administrative regions (i.e. the behavior of BGP protocol across AS boarder).

Furthermore, we conceive a more ambitious (adventurous) use of PVS to aid the verification of BGP system convergence using a relaxed algebra model. As depicted in Figure 2, we label the set of atomic metarouting algebras with type `A` and denote them with the inner ring. We then observed that all metarouting algebras that can be composed from type `A` algebras by composition satisfy monotonicity by definition and therefore fall into type `B` algebras represented by the middle ring. Sobrinho's original paper [9] showed that monotonicity is a *sufficient* (not necessary) condition for BGP system convergence. There are known BGP systems that converge but violate monotonicity, and this reveals existence of type `C` algebras modeling the set of converging BGP systems that are not monotonic. By relaxing the monotonicity property, we would like to explore the modeling and reasoning of type `C` systems that fall outside Monotonic type `B` Algebra (the middle ring) but are equally good with respect to convergence. Taking this basic approach one step further, instead of starting from the algebra model, we would like to develop in PVS an algebraic representation of a given BGP system that falls outside the scope of current metarouting algebra, and with the aid of PVS proof engine, decide if that corresponding BGP system falls into type `C` and converges or type `D` that does not converge.

6. REFERENCES

[1] EE, C. T., CHUN, B.-G., RAMACHANDRAN, V., LAKSHMINARAYANAN, K., AND SHENKER, S. Resolving Inter-Domain Policy Disputes. In *SIGCOMM* (2007).

[2] GRIFFIN, T. G., SHEPHERD, F. B., AND WILFONG, G. The stable paths problem and interdomain routing. *IEEE/ACM Trans. Netw.* (2002).

[3] GRIFFIN, T. G., AND SOBRINHO, J. L. Metarouting. In *ACM SIGCOMM* (2005).

[4] GRIFFIN, T. G., AND WILFONG, G. An Analysis of BGP Convergence Properties. *SIGCOMM Comput. Commun. Rev.* (1999), 277–288.

[5] GUNTER, E. L. Doing algebra in simple type theory.

[6] KILLIAN, C., ANDERSON, J., JHALA, R., AND VAHDAT, A. Life, death, and the critical transition: Finding liveness bugs in systems code. In *NSDI* (2007).

[7] LABOVITZ, C., MALAN, G., AND JAHANIAN, F. Internet Routing Instability. *ACM/IEEE Trans. on Networking* (1998).

[8] OWRE, S., AND SHANKAR, N. Theory interpretations in pvs. Tech. rep., 2001.

[9] SOBRINHO, J. Network routing with path vector protocols: theory and applications. In *SIGCOMM* (2003).

[10] SOBRINHO, J. An algebraic theory of dynamic network routing. Tech. rep., October 2005. (William R. Bennett Prize 2006).

[11] WINDLEY, P. J. Abstract theories in hol. In *HOL'92: Proceedings of the IFIP TC10/WG10.2 Workshop on Higher Order Logic Theorem Proving and its Applications* (1993), North-Holland/Elsevier, pp. 197–210.