Yale University
Department of Computer Science

Relating Two Formal Models of Path-Vector Routing

Aaron D. Jaggard Vijay Ramachandran

YALEU/DCS/TR-1301
July 2004

This work was partially supported by the U.S. Department of Defense (DoD) University Research Initiative (URI) program administered by the Office of Naval Research (ONR). A revised version of this work will appear in *Proceedings of IEEE INFOCOM 2005*.

Relating Two Formal Models of Path-Vector Routing

Aaron D. Jaggard, Vijay Ramachandran

***Abstract*— This paper unifies two independently developed formalisms for path-vector routing protocols such as the Border Gateway Protocol (BGP), the standard inter-domain routing protocol for the Internet. The works of Griffin, Jaggard, and Ramachandran [4] and Sobrinho [8] proved conditions for guaranteed protocol convergence, but as they operate at different levels of abstraction in modeling the protocols, the relationship between them is not obvious. Here we provide a rigorous translation between the two frameworks and use it to connect the convergence results, yielding a more complete set of analysis tools than in either paper alone. We motivate our discussion by presenting an example of applying both frameworks to analyze a set of protocols; in doing so, we show how the models, in conjunction, give important guidelines for protocol design.**

***Index Terms*— Graph theory, Routing**

I. Introduction

The Border Gateway Protocol (BGP) is a path-vector protocol used for inter-domain routing on the Internet [7]. Because the domains, or autonomous systems (ASes), are independently administered, routing is calculated on a hop-by-hop basis: Each AS learns information about reachable destinations from its neighbors, determines best routes based on its routing policies, and shares its choice with its neighbors who, in turn, determine their best routes using their policies. The languages and techniques used to write policies are not part of the BGP specification, however. Routers are usually configured using vendor-developed languages that have evolved from the needs of router operators but in an environment lacking vendor-independent standards. Configurations can be quite expressive, allowing network engineers to encode complex policies that deal with various scenarios and interests.

This work is partially supported by the U.S. Department of Defense (DoD) University Research Initiative (URI) program administered by the Office of Naval Research (ONR). A. D. Jaggard is at Dept. of Mathematics, Tulane Univ., New Orleans, LA, USA, adj@math.tulane.edu; he is partially supported by ONR Grant N00014-01-1-0795. V. Ramachandran is at Dept. of Computer Science, Yale Univ., New Haven, CT, USA, vijayr@cs.yale.edu; he is partially supported by a 2001–2004 DoD NDSEG Fellowship and by ONR Grant N00014-01-1-0795.

This expressiveness comes with a cost: Interaction between locally defined routing policies can lead to global routing anomalies, such as protocol divergence [1], [6], [9]. Unfortunately, in the general case, it is *NP*-hard to check all policies to prevent such interactions before they happen [5], and the protocol is not built to detect such interactions even when they happen. This motivates the study of path-vector protocols from a design perspective. Ideally, we want a balance of expressiveness and robustness[1]; however, this is not possible with today's BGP-configuration methods. To achieve this, we must understand the effect of policy elements on protocol behavior so that appropriate policy-configuration techniques and languages can be designed.

Recently, papers by Sobrinho [8] and Griffin, Jaggard, and Ramachandran [4] presented independently developed formal models for path-vector routing. Both are frameworks for protocol design, rigorously defining desirable protocol properties and identifying the conditions needed to achieve them. Both papers establish abstract formalisms with which to discuss protocol semantics separate from specific networks or implementation details. Furthermore, both papers provide protocol-design guidelines that provably guarantee protocol convergence on any network. Although the results are similar, the relationship between these two models is unclear.

This paper establishes this relationship and provides a context for understanding the process of protocol design from abstract specification to implementation. We not only prove that many results from the two papers above are indeed equivalent, but we also show how to translate a protocol-design specification and framework-specific design properties from one model into the other. This is beneficial because, as we show, each model has its particular strengths in discussing different parts of the protocol-design puzzle. The models focus on different levels of abstraction of protocol design; being able to use the complementary machinery of these two frameworks allows a more complete analysis of a protocol. Finally, we summarize what is known about achieving robustness and other design goals, using our equivalence results.

[1]These properties are discussed in the next section.

A. Related Work

A series of papers apply theoretical methods to analyze the convergence of BGP and related protocols. Gao and Rexford in [3] showed that obeying constraints on local configuration can guarantee global BGP convergence. Griffin, Shepherd, and Wilfong [5] introduced the Stable Paths Problem (SPP) to model the static semantics of path-vector routing and introduced a rigorous model for analyzing robustness, giving hardness results for the underlying problem that inter-domain routing protocols are trying to solve. Gao, Griffin, and Rexford [2] combined the results of these two papers and showed how to use the previous results to design a safe protocol allowing back-up routes. All of these papers recognize the various routing anomalies in BGP [1], [6], [9] and attempt to constrain the protocol so that the interaction of autonomously defined policies do not result in protocol divergence.

Griffin, Jaggard, and Ramachandran [4] and Sobrinho [8] extended the work of these papers to complete frameworks that model path-vector routing protocols in general. The former proves that inherent trade-offs exist in the design space of path-vector protocols; in particular, that reaching a combination of desirable design goals (including protocol convergence) is impossible without non-trivial global constraints. The latter gives a simple model for analyzing protocol convergence. Both of these prove sufficient conditions that guarantee convergence, and here we answer the natural question of how these conditions relate.

II. Two Routing Formalisms

We begin with a discussion of two existing formal models for the design of path-vector protocols. Both frameworks model the static semantics of path-vector protocols so that properties can be analyzed without being tied to the specifics of certain networks or implementations. Before we review each framework in detail, we discuss some general concepts for path-vector routing—basic design properties and the intended routing dynamics—because they motivate the definitions of the models' components. For each model, we then examine the framework components, the relationship of those components to routing dynamics, and any relevant, framework-specific properties.

A. Path-Vector Routing

Design Properties The work in [4] identifies several dimensions of the protocol design space and the trade-offs inherent in this space. The framework in [8] is more abstract, thus it does not model all of these dimensions. In this paper we focus on properties regarding protocol convergence, which is modeled by both frameworks and is arguably the most important protocol-design goal. Below we give definitions for these properties of interest; they are compatible with the formal definitions in [4] and [8].

Expressiveness The *expressiveness* of a protocol is essentially the number of routing configurations that can be captured by the protocol, *i.e.*, how can route information be set by a router, and how many ways can paths be ranked at a node?

Robustness A protocol *converges robustly* on a network if a unique routing solution is found, even in the presence of link or node failures.

Optimality A protocol *converges optimally* if every router is assigned its most preferred path out of all possible paths (not just those made available to it while running the protocol).

Routing Dynamics Routes to a network destination d are established using the following iterative process: the router most directly associated with d, *e.g.*, the BGP speaker for d, *originates* routes to d by notifying its neighbors that it can reach d; its neighbors then *advertise* to their neighbors that they can reach d, *etc.* A router may receive multiple advertisements for paths to d; it must then decide which path is *best*. Because each AS is administered autonomously, this choice depends only on the *policies* configured locally at that router. What information about paths is available to the router is part of the protocol specification: the routing frameworks both have some *path-data-structure* component that models this. Policies, then, are essentially transformations to the path data structure during an advertisement exchange between neighbors that affect the choice of best path. Both frameworks model constraints on these policies as part of the protocol specification. The model in [4] differentiates policies applied when receiving an advertisment, an *import policy*, and those used when sending a route advertisement, an *export policy*. Changes in routes can occur when links or nodes are added or removed from the network and when changes are made to router configurations; routes are then withdrawn or added by sending the appropriate reachability advertisements. Because all choices of best path depend on the composition of policies along advertised routes, changes in best route are advertised using the above-mentioned iterative process.

B. Path-Vector Algebras

Framework Components A *path-vector algebra* [8] describes some basic semantics of a path-vector protocol. It is a seven-tuple $(\mathcal{W},\ \preceq,\ \mathcal{L},\ \Sigma,\ \phi,\ \oplus,\ f)$ comprising:

- $\mathcal{W}$ a set of *weights* totally ordered by $\preceq$;
- $\mathcal{L}$ a set of *labels*;
- Σ a set of *signatures* containing the special signature ϕ;
- $\oplus$ a binary operation $\oplus : \mathcal{L} \times \Sigma \to \Sigma$; and
- f a "weighing function" $f : \Sigma \to \mathcal{W}$.

Signatures model the path data structure; they contain enough information to determine a path's weight using the function f. Weights influence choice of best route; heavier paths are less preferred. Each directed signaling edge in a network is associated with a label, which models the transformation made to a path's data structure when advertised along the edge, *i.e.*, labels correspond to import and export policies along the edge. The operation $\oplus$ computes the signature for a path advertised to a neighbor given the label on the signaling edge and the path's original signature; this amounts to *applying* the edge policy to the path data structure on extension.

Note that instantiating an algebra with particular labels, signatures, *etc.*, produces a protocol; this protocol may then be instantiated with a specific network and assignments of weights to edges, *etc.*.

Dynamics Given an algebra, a path-vector protocol consistent with it would run in accordance with the following dynamics: a node v knows of path P to d when it has a signature for P, either $s(d)$ for the empty path to d (when d is in v's own AS), or $s(P)$ for a path extending a neighbor's path to d; the best path P' to d is the path with lowest weight; to advertise a path P to u, $s(P)$ is sent along the signaling edge (v, u) with some associated label l, and $s(uP) = l \oplus s(P)$ is the signature of the imported, extended path at u.

Property Definitions It is assumed that the following two properties hold for any path-vector algebra.

MAXIMALITY $\forall_{\alpha \in \Sigma - \{\phi\}}\ f(\alpha) \prec f(\phi)$

ABSORPTION $\forall_{l \in \mathcal{L}}\ l \oplus \phi = \phi$

The special signature ϕ represents an unusable path, and so these properties mean that an unusable path is always least preferred and is never extended to a usable path.

The following three properties are relevant to studying protocol convergence and optimality.

ISOTONICITY $\forall_{l \in \mathcal{L}} \forall_{\alpha,\beta \in \Sigma}\ (f(\alpha) \preceq f(\beta))$
$\Rightarrow (f(l \oplus \alpha) \preceq f(l \oplus \beta))$

MONOTONICITY $\forall_{l \in \mathcal{L}} \forall_{\alpha \in \Sigma}\ f(\alpha) \preceq f(l \oplus \alpha)$

STRICT MONOTONICITY $\forall_{l \in \mathcal{L}} \forall_{\alpha \in \Sigma - \{\phi\}}$
$f(\alpha) \prec f(l \oplus \alpha)$

Isotonicity implies that the relative weights of paths are preserved when extended along the same edge, *i.e.*, two paths meeting at a node cannot flip-flop in rank as they are extended. Monotonicity means that the actual weights of paths do not decrease as they are extended. Strict monotonicity is enough to guarantee robust protocol convergence on any network; monotonicity alone only guarantees protocol convergence on networks with the following property (*free* networks).

FREENESS $\forall_{\text{cycles } u_n \cdots u_1 u_0} \forall_{w \in \mathcal{W} - \{f(\phi)\}} \exists_{0 \le i \le n} \forall_{\alpha \in \Sigma}$
$(f(\alpha) = w) \Rightarrow (f(l(u_i, u_{i-1}) \oplus \alpha) \neq w)$

This will be discussed in more detail in Section V.

C. Path-Vector Policy Systems

Framework Components A *globally constrained path-vector policy system* (PVPS) [4] is a triple $(PV,\ PL,\ \kappa)$ that models the specification of a path-vector protocol. The latter two components are not modeled by the algebraic formalism, so we discuss their roles here only briefly.

The *policy language* PL is intended to be a high-level language for policy writers in which to express their routing configuration, *i.e.*, how to process path information on import, export, and origination. A *policy configuration* $p \in PL$ will essentially be "compiled down" to a set of transformations on path data structures that conform to the protocol's specification of legal policies. Multiple policy languages can be used for the same path-vector system PV. The *global constraint* κ is a predicate on specific networks running the protocol. It is intended to be an assumption about the network that cannot be checked by examining nodes' policy configurations individually. The constraint can be enforced in a number of ways—we will not discuss the details in this paper—including supplemental protocols or economic incentives. A non-trivial global constraint is often necessary to achieve a combination of desirable protocol design goals [4].

The path-vector system PV describes the underlying mechanism to store and exchange route information. It, like the algebra, describes the semantics of the protocol, but with more attention to some implementation details. A complete description of path-vector-system components can be found in Section 2.2 of [4], but we provide a brief explanation below.

- $\mathcal{R}$ is the set of *path descriptors*, the path data structure.
- ω is the *rank map* that is used to rank path descriptors. $\omega : \mathcal{R} \to \mathcal{U}$ takes a path descriptor to a rank in a totally ordered set $\mathcal{U}$.
- λ^{in}, λ^{out} are the *local constraints* on policies.
- t^{in}, t^{out} are the *protocol transformation functions* that describe how local import and export policies are applied to path descriptors by the protocol.
- o is the constraint on originated path descriptors.

The components of a path descriptor $r \in \mathcal{R}$ are the *attributes* of r. Example attributes are the destination IP-address block of the path, a preference value, and the AS-path itself. A descriptor contains enough information about a path to determine its rank; lower rank is more preferred. The local constraints are the primary limitations on the expressiveness of the protocol. They are the protocol's specification of legal routing policies—intuitively, the less strict the constraints, the more opportunity for policy-generated protocol divergence. Although router operators write policies, it is important to realize that the router performs the low-level *application of the policies* to the path data structures. The specific way in which this is done is captured by the protocol transformation functions; *e.g.*, the router might automatically filter out loops or conditionally apply policies. O is essentially a data-structure integrity check on newly originated path descriptors, but could be used to enforce some additional constraints.

Instances I of the path-vector policy system are pairs (G, P) of an undirected network $G = (V, E)$ and a configuration function $P : V \to PL$ that maps nodes to policy configurations expressed in the policy language PL. A policy configuration determines the import and export policies at node v for each neighbor u, written $F^{in}_{(v,u)}$ and $F^{out}_{(v,u)}$, respectively.

Dynamics A protocol modeled by a path-vector policy system would run in accordance with the following dynamics: nodes originate destinations by creating path descriptors that satisfy the constraint O; these are then exported to and imported by neighbors; a best path to a given destination has lowest rank; once this is done, the descriptor for the best path is exported to and imported by neighboring routers to advertise the choice.

If node v advertises descriptors $X \subseteq \mathcal{R}$ to node u, then the descriptors for the extended paths at u will be

$$f_{(u,v)}(X) = t^{in}\left(v, u, F^{in}_{(v,u)}, t^{out}(u, v, F^{out}_{(u,v)}, X)\right).$$

The function $f_{(u,v)}$ is called the *arc-policy function* for signaling edge (u, v). For convenience, we also define the path descriptor associated with a putative route $P = v_n v_{n-1} \cdots v_0$ from node v_n to destination v_0 as

$$r(P) = f_{(v_{n-1},v_n)} \circ f_{(v_{n-1},v_{n-2})} \circ \cdots \circ f_{(v_0,v_1)}(\{r_0\}),$$

where r_0 is the path descriptor originated by v_0 for the destination of P. If P is not permitted at a node then assume $r(P) = \emptyset$; in particular, for any path P that contains a filtered subpath, then $r(P) = \emptyset$.

Property Definitions We always assume that

$$\forall_{u,v \in V} \ \left|f_{(u,v)}(R)\right| \leq |R|, \tag{1}$$

and, in particular, $f_{(u,v)}(\emptyset) = \emptyset$, *i.e.*, new path descriptors cannot be originated by import and export policies. It is possible that a path descriptor is filtered on import or export; in this case, one of the nodes or the protocol has removed the route from consideration, which is why equality does not always hold in (1). Because of filtering, a node v might not receive an advertisement for a destination reachable through one of its neighbors; the set of path descriptors to the destination at v is then simply $\emptyset$.

The following PVPS property has direct relevance to protocol convergence.

INCREASING $\forall_{(u,v)\in E} \forall_{\text{paths } P=u\cdots} \ \omega(r(P)) < \omega(r(vP))$ when $r(P),\ r(vP) \neq \emptyset$.

A path-vector system that is increasing describes a protocol that converges robustly for every instance.

III. Three Levels of Abstraction

Having summarized the algebra and PVPS frameworks, we now examine the relation between them. We begin by examining how the two frameworks are applied to a set of simple protocols. Although these protocols have a simple route-selection procedure so that the example is easy to diagram below (see Tables I–III), the frameworks can just as easily isolate the important factors for convergence of protocols having the complexity of BGP itself.

From this example, it becomes clear that modeling occurs at three levels of abstraction: moving from (1) properties that describe a set of protocols; to (2) a specification for one particular protocol with some added implementation details; finally to (3) the properties of a given protocol on particular networks. These three levels of abstraction naturally correspond to: (1) the algebra framework; (2) the PVPS framework; and (3) instances of the Stable Paths Problem (SPP).[2] We then ask the natural question of how the two frameworks, and these levels of abstraction, fit together. In this section, we give the intuition behind using both frameworks to analyze protocols at all three levels. Then, in Section IV, we give a rigorous translation between the two frameworks; this translation relates the language and notation originally suited to each framework's context. Finally, in Section V, we examine the relationship between the frameworks' protocol-design guidelines using our translation.

Example Protocols Table I gives a summary of six example path-vector protocols. In each of these protocols, the path data structure includes a path cost in $\mathbb{Z}$

[2]SPP was introduced in earlier work by Griffin, Shepherd, and Wilfong [5]. It is used to describe a metric for expressiveness in [4] and will be discussed in Section IV, where we incorporate it as a metric for the algebra framework as well.

TABLE I

EXAMPLE PROTOCOLS USING THE PATH DATA STRUCTURE $\{\text{cost}, \text{path length}\}$.

PROTOCOL	PRIMARY RANK CRITERION	SEC. RANK CRITERION	COST CONSTRAINT	RANK PROPERTY
(1)	cost, prefer lower	path length, prefer shorter	nondecreasing	strict monotone
(2)	cost, prefer lower	path length, prefer shorter	none	none
(3)	path length, prefer shorter	cost, prefer lower	none	strict monotone
(4)	cost, prefer lower	none	nondecreasing	monotone
(5)	cost, prefer lower	none	none	none
(6)	path length, prefer shorter	none	none	strict monotone

TABLE II

ALGEBRAS FOR PROTOCOLS IN TABLE I.

ALGEBRA	WEIGHT	LABEL	SIGNATURE	CNV. PROPERTY
(A)	$(\text{cost}, \text{path length})$ lexically ordered	edge cost $\in \mathbb{N}$, $+1$	path cost and path length	strict monotone
(B)	$(\text{cost}, \text{path length})$ lexically ordered	edge cost $\in \mathbb{Z}$, $+1$	path cost and path length	cost constraint
(C)	$(\text{path length}, \text{cost})$ lexically ordered	$+1$, edge cost $\in \mathbb{Z}$	path length and path cost	strict monotone
(D)	cost only, prefer lower	edge cost $\in \mathbb{N}$	total path cost	monotone
(E)	cost only, prefer lower	edge cost $\in \mathbb{Z}$	total path cost	cost constraint
(F)	path length only, prefer shorter	$+1$	path length	strict monotone

and a path length in $\mathbb{N}$; we assume that nodes may modify path cost on import and export, and that the protocol automatically updates the path length data as paths are shared. The protocols differ in which of these two components they use as the primary determinant of path rank, whether they use the other component as a secondary factor, and whether they place additional restrictions on how the path cost is modified as paths are extended. As we are primarily interested in relating these protocols to the algebra and PVPS frameworks, we do not assume any protocol details at the level of, *e.g.*, [7].

Protocols (1)–(3) use both cost and path length to determine the rank of a route while Protocols (4)–(6) only use one of these components. In Protocols (1)–(2), the cost of a path is the primary criterion and path length the secondary criterion for determining rank: paths with lower cost are always preferred, and paths with shorter length are preferred among paths with equal cost. In Protocol (3), the reverse is true, *i.e.*, shorter paths are preferred most and ties are broken by choosing the path with lower cost. For Protocols (1) and (4), we require that path cost does not decrease when a path is extended (this is indicated in the COST CONSTRAINT column), while the other protocols allow negative costs to be associated with edges.

The column RANK PROPERTY abuses notation slightly; it uses some of the algebra properties discussed above to describe what happens to path rank in a protocol (rather than path weight in an algebra) when a path is extended. In Protocols (1), (3), and (6), the rank of a path must increase because path length is included in the calculation of rank and paths only increase in length as they are extended—thus we say that rank is strict monotone. Note that in Protocol (2), even though path length is a component of rank calculation, rank is not strict monotone or even monotone because the primary criterion for rank is path cost, which could possibly decrease (this protocol does not enforce a constraint on cost values); the lack of a constraint also tells us nothing about rank in Protocol (5). In Protocol (4), path cost is constrained to be nondecreasing and is the sole criterion for determining rank, so we can say the rank is monotone.

Algebras for Protocols Now consider the algebras in Table II. Recall that weight essentially describes some part of rank calculation. Lighter paths are always preferred to heavier ones; so, to be consistent with an algebra, a protocol must evaluate a path's weight first in determining rank. By their definitions of weight, Algebras (D) and (E) can, in general, describe protocols whose primary rank criterion is path cost, Algebra (F) can describe protocols whose primary rank criterion is path length, and Algebras (A), (B), and (C) can describe protocols where rank is computed using some combination of the two in the correct order.

Of the protocols from Table I, Algebra (F) can describe Protocols (3) and (6), while Algebra (C) can describe Protocol (3) only (Protocol (6) does not consider cost at all; thus, it may be the case that Protocol (6) breaks ties in path length inconsistent with the smaller-cost preference of Algebra (C)). Both of these algebras

TABLE III
EXAMPLE PATH-VECTOR SYSTEMS USING THE PATH DESCRIPTOR $\{\text{cost}, \text{path length}\}$.

PV	RANK MAP	POLICY CONSTRAINT
(1)	$\omega : (c, n) \mapsto (c, n)$, lexically ordered	$\text{L}^{in}(f), \text{L}^{out}(f) : ((c', n') = f(c, n)) \Rightarrow ((n' = n) \wedge (c \leq c'))$
(2)	$\omega : (c, n) \mapsto (c, n)$, lexically ordered	$\text{L}^{in}(f), \text{L}^{out}(f) : ((c', n') = f(c, n)) \Rightarrow (n' = n)$
(3)	$\omega : (c, n) \mapsto (n, c)$, lexically ordered	$\text{L}^{in}(f), \text{L}^{out}(f) : ((c', n') = f(c, n)) \Rightarrow (n' = n)$
(4)	$\omega : (c, n) \mapsto c \in \mathbb{N}$	$\text{L}^{in}(f), \text{L}^{out}(f) : ((c', n') = f(c, n)) \Rightarrow ((n' = n) \wedge (c \leq c'))$
(5)	$\omega : (c, n) \mapsto c \in \mathbb{Z}$	$\text{L}^{in}(f), \text{L}^{out}(f) : ((c', n') = f(c, n)) \Rightarrow (n' = n)$
(6)	$\omega : (c, n) \mapsto n \in \mathbb{N}$	$\text{L}^{in}(f), \text{L}^{out}(f) : ((c', n') = f(c, n)) \Rightarrow (n' = n)$

are strict monotone because path length must increase when a path is extended, and both these protocols have strict monotone rank (as discussed above). Thus, they are indeed consistent with the algebras' prescribed behavior. Note that both protocols implement the semantics of Algebra (F), but Protocol (3) in its specification breaks ties by cost while Protocol (6) does not. However, any protocol implementing Algebra (C) or (F) is strict monotone and thus converges robustly for any network (see the CONVERGENCE PROPERTY column of Table II); essentially, the detail of how the protocol then looks at cost is irrelevant to convergence. So, the algebra can be used to isolate the semantics that are most useful for understanding protocol convergence. (In particular, if Protocol (6) preferred higher path cost as its secondary rank criterion, it would still converge robustly on any network.)

Similarly, Algebra (E) can describe Protocols (1), (2), (4), and (5); Algebra (B) can describe Protocols (1) and (2); Algebra (D) can describe Protocols (1) and (4); and Algebra (A) can describe Protocol (1) only. Because path cost is most important in calculating weight for these algebras, the convergence of protocols consistent with these algebras depends on whether the permitted edge costs give monotonicity or not; this is seen in the CONVERGENCE PROPERTY column in Table II. Protocols consistent with Algebra (B) or (E) could permit negative edge costs, *e.g.*, Protocols (2) and (5), which do not necessarily converge robustly on any network. Indeed, this is specifically why we cannot make a general robustness claim about Algebras (B) and (E), and why any consistent protocol's convergence claim depends on the protocol's cost constraint.

Protocols (1) and (4) do have the additional constraint that cost is nondecreasing; they are not only consistent with Algebra (E) but are also consistent with Algebra (D). Because edge costs in $\mathbb{N}$ give monotonicity, we can say that Algebra (D) is monotone, and any protocol consistent with it will at least have monotone rank.

Note that Protocol (1), while consistent with a monotone algebra, has the additional property of breaking ties in path cost by using a strictly monotone data component (path length); thus, we can say that Protocol (1) has strict montone rank—this may not be the case for any protocol described by Algebra (D), even if the cost/label set is nonnegative, *e.g.*, Protocol (4). However, the combination of nondecreasing cost and path length in Algebra (A) ensures strict monotone weights. Of the example protocols, only Protocol (1) is consistent with Algebra (A); Protocol (2) is inconsistent because of its lack of cost constraint, even though both cost and path length are used in computation of rank, and Protocol (4) could break ties in cost arbitrarily, as it ignores path length.

PVPSes for Protocols In Table III, we show six path-vector systems that correspond by number to the protocols in Table I. In these systems, the path descriptor is the same data structure $\{\text{cost}, \text{path length}\} \subset \mathbb{Z} \times \mathbb{N}$ used by the protocols; the predicate O requires that originated path descriptors have the form $(0, 0)$; $t^{in}(u, v, f, X) = \{f(r) \mid r \in X : r \text{ describes an acyclic path}\}$, *i.e.*, the import transform applies import policies but filters routing loops; and $t^{out}(u, v, f, X) = \{(c, n + 1) \mid (c, n) \in f(X)\}$, *i.e.*, the export transform applies export policies but increments the path length automatically. The other components—the rank map ω and the local policy constraints L^{in} and L^{out}—are shown in Table III.

Each PVPS describes the corresponding Table I protocol because its rank criteria are captured by the definition of ω and its cost constraint is captured by the definition of L^{in} and L^{out}. Therefore, the consistency relationship between the Table II algebras and these PVPSes is exactly the same as that between the algebras and the Table I protocols. It can be seen that the PVPS directly models one protocol, and any protocol that satisfies the PVPS-framework property 'increasing' will converge on any network. The PVPS framework can also be used to analyze design properties of a given protocol other than convergence, but this is beyond the scope of this paper.

Levels of Abstraction Using the two frameworks, we can discuss protocol design in three distinct levels of

TABLE IV
INFORMAL TRANSLATION BETWEEN PATH-VECTOR SYSTEMS AND ALGEBRAS.

Algebra	Class-Based PVPS	General PVPS
$\mathcal{W}$	$\{g\} \subset \mathbb{N}$	$\mathcal{U}$
$\preceq$	$\leq$	$\leq_{\mathcal{U}}$
Σ	$\mathcal{R}$	$\mathcal{R}$
$\mathcal{L}$	$\{f_{(u,v)}\}$	$\{f_{(u,v)}\}$
$l \oplus \sigma$	$f_{(u,v)}(r)$	$f_{(u,v)}(r)$
$\phi \in \Sigma$	$\emptyset$	$\emptyset$
$f : \Sigma \to \mathcal{W}$	$\pi_g\omega : \mathcal{R} \to \{g\}$	$\omega : \mathcal{R} \to \mathcal{U}$
monotonicity	$\pi_g\omega(r) \leq \pi_g\omega(f(r))$	$\omega(r) \leq_{\mathcal{U}} \omega(f(r))$
isotonicity	essentially, policies act by edge, not by path	
iso. & mono.	essentially, policies are consistent with a non-negative per-edge cost function	
maximality	prefer any route to $\emptyset$	
absorption	$f_{(u,v)}(\emptyset) = \emptyset$	
optimal in-tree	a solution which gives all nodes most-preferred path	
local-optimal in-tree	a path-vector solution	

abstraction: the algebra level, the protocol level, and the network level. A move from one level of abstraction to another is not uniquely determined, as we explore below.

Because algebras do not specify all the implementation details for a protocol, an algebra can describe a set of path-vector protocols. The detail most relevant to protocol convergence is how to decide between equal-weight routes. Because weight influences path rank but does not totally determine it, some additional method must be used to rank paths of the same weight. Any strict-monotone criterion, such as path length, guarantees robustness for that protocol. The different methods for tie-breaking correspond to different protocol instantiations of the algebra.

Just as an algebra describes a set of protocols, a given protocol might have multiple algebras that describe it. The definition of weight, labels, and signatures in an algebra correspond to only some part of the path data structure; different subsets correspond to different algebras. Thus, in translating protocols to algebras, a select part of the protocol behavior can be isolated and studied; this is a useful tool for analyzing convergence constraints for a complex protocol an an important role for the algebra framework.

Although a PVPS does not include the bit-level details of a full protocol specification (*e.g.*, [7]), it does include all of the essential details of a protocol; we therefore identify a PVPS with a single protocol. At this level of abstraction, protocol designers can study the balance between enforcing constraints, convergence, and other properties from [4] using results from the PVPS framework.

Any protocol, whether a PVPS or a protocol instance of an algebra, may be instantiated further to a particular network with specific node policies or edge weights and signatures. Each instance is a permitted routing configuration by that protocol; the sets of routing configurations is used later in this paper as a metric of expressiveness to rigorously match algebras and PVPSes. Some protocols conditionally converge and require specific constraints on the network; one of these constraints is the freeness property defined in Section II-B, which is an example of a *network-level* design property.

IV. MAPPING BETWEEN FORMALISMS

A. Intuition

It is easy to see that both frameworks model similar protocol components although different notation is used. So, before showing the rigorous translation between formalisms, we outline the intuitive relationship between framework components. Table IV summarizes this correspondence (in addition to the relationship with 'class-based' PVPSes [4]) and the properties defined for each.

The rows of the table list components as follows. Paths are preferred based upon their weight in the set $\mathcal{W}$ ordered by $\preceq$ (algebra) or rank in the set $\mathcal{U}$ ordered by $\leq_{\mathcal{U}}$ (PVPS). Each framework has a function—f and ω, respectively—that assigns values from the ranking set to other objects (the set Σ of signatures in an algebra or the set $\mathcal{R}$ of path descriptors in a PVPS).[3] Each path in the network is assigned one of these objects, so we want to view the sets Σ and $\mathcal{R}$ as containing the information about paths that is exchanged between nodes. With this in mind, it is natural to identify a

[3]Path descriptors in class-based systems, discussed again in Sec. IV-E, have a *level* attribute g which is the primary factor in computing the rank of a path. It is natural to associate this to the primary determinant of path preference, *i.e.*, weight, in an algebra; this accounts for the differences between the second and third columns of Table IV.

function $l \oplus \_ : \Sigma \to \Sigma$, which modifies signatures passed over an edge labeled with l, with an arc policy function $f_{(u,v)}$, which modifies path descriptors exchanged over the edge (u, v); in particular, we should view the labels of edges as corresponding to the arc policies on edges. The final component of an algebra, the special signature $\phi \in \Sigma$ for unusable paths, is most naturally identified with the empty set, the result of filtering paths.

The informal translation of properties between the two frameworks shows some of their differences. Convergence properties studied in [8] may be translated to the language of PVPSes as shown in Table IV. Many of the properties studied in [4] involved policies and implementation details not explicitly included in algebras; we thus do not translate these to the language of algebras, although they may of course be investigated for particular protocols derived from algebras.

Note that it is the *arc policies* (combinations of nodes' import and export policies and the policy application functions) of a PVPS that correspond to the edge labels in an algebra; the abstraction of the algebra framework does not explicitly include the separate import and export policies of individual nodes.

B. Algebra-Protocol Consistency

Here we make rigorous the relationship between the three levels of protocol-design abstraction that naturally follow from our example; we do this by extending the notion of expressiveness, defined in [4] for path-vector systems, to algebras. The semantic domain used for expressiveness is the Stable Paths Problem (SPP) [5]; we give a condensed definition below. An SPP instance corresponds to one routing configuration (at the network level of abstraction).

Definition 4.1: $S = (G,\ v_0,\ \mathcal{P},\ \Lambda)$ is an instance of the *Stable Paths Problem (SPP)* if $G = (V,\ E)$ is a finite undirected graph, $v_0 \in V$ (called the *origin*), $\mathcal{P}$ is a set of simple paths in G terminating at v_0, and the mapping Λ takes nodes $v \in V$ to a path ranking function $\lambda^v = \Lambda(v)$. Each λ^v is a function that takes a path in $\mathcal{P}^v = \{P \in \mathcal{P} \mid P \text{ is a path starting at } v\}$ to its *rank* in $\mathbb{N}$. If $W \subseteq \mathcal{P}^v$, then define the subset of "best paths" in W $\min(\lambda^v,\ W) = \{P \in W \mid \text{for every } P' \in W,\ \lambda^v(P) \leq \lambda^v(P')\}$. A *path assignment* for an SPP-instance S is any mapping π from V to subsets of $\mathcal{P}$ such that $\pi(v) \subseteq \mathcal{P}^v$. The set candidates$(u, \pi)$ consists of all permitted paths at u that can be formed by extending the paths assigned to neighbors of u. For $u = v_0$, candidates$(u, \pi) = \{(u)\}$, and for $u \neq v_0$, candidates$(u, \pi) = \{uQ \in \mathcal{P}^u \mid \{v,\ u\} \in E \text{ and } Q = \pi(v)\}$. A path assignment π is a *solution* for an SPP if for every node u we have $\pi(u) = \min(\lambda^u,\ \text{candidates}(u, \pi))$.

Two SPP instances are equivalent iff the set of permitted paths and the ordering of those paths by rank at each node are the same. Let $\mathcal{E}(S)$ be the set of all SPPs equivalent to the SPP instance S.

The constraints of a protocol specification determine the set of network configurations possible when running a protocol. This idea motivates the definitions below for the protocol and algebra levels of abstraction, in which expressiveness is defined as the set of permitted routing configurations.

Definition 4.2: Given a path-vector system PV, let $I(w)$ be a *restriction* of a PV-instance I in which the only destination originated is w. Each restriction maps to an SPP instance $S_{(I,w)}$ in which the set of permitted paths is $\{P \mid r(P) \neq \emptyset\}$, and at each node $v \in V$, the ranking function is consistent with path-descriptor rank: $\lambda^v(P_1) < \lambda^v(P_2)$ iff $\omega(r(P_1)) < \omega(r(P_2))$ and $\lambda^v(P_1) = \lambda^v(P_2)$ iff $\omega(r(P_1)) = \omega(r(P_2))$. Define the *expressiveness of* PV as the set $\mathcal{M}(PV) = \{\mathcal{E}(S_{(I,w)}) \mid w \in V,\ r_w \in F^{orig}(w)$ in some PV-instance $I\}$.

Definition 4.3: Given an algebra A, let G^A be the set of all networks whose edges are assigned labels from A. For each $G \in G^A$, let $T(G, w)$ be the restriction of G where only one destination w is originated with the signature $s(w)$. Each $T(G, w)$ maps to an SPP instance $\mathcal{S}^A(T(G, w))$ in which the set of permitted paths is $\{P \mid s(P) \neq \phi\}$ and the ranking functions are consistent with weights in the following manner:

(i) if $\lambda(P) < \lambda(P')$ then $f(s(P)) \preceq f(s(P'))$,
(ii) if $\lambda(P) = \lambda(P')$ then $f(s(P)) = f(s(P'))$, and
(iii) if $f(s(P)) \prec f(s(P'))$ then $\lambda(P) < \lambda(P')$.

Define the *expressiveness of an algebra* A to be the set $\mathcal{X}(A) = \{\mathcal{E}(S) \mid S = \mathcal{S}^A(T(G, w))$ for some $G \in G^A,\ w \in V\}$.

Definition 4.4: A path-vector system PV and an algebra A are *consistent* iff $\mathcal{M}(PV) \subseteq \mathcal{X}(A)$.

If A and PV are consistent, we may be able to relate the properties of each. Because consistency is defined in terms of equivalence classes of SPPs, we will use properties which hold for *some* SPP in each of the equivalence classes that defines the expressiveness of an algebra or PVPS. We start with the following definition.

Definition 4.5: Let P be an algebra or PV property. We say that P can be *defined in terms of SPP instances* if there is an equivalent property $\hat{P}$ that holds for network instances of the algebra or PV. Formally, the algebra algebra A has property P iff for every $E \in \mathcal{X}(A)$, there is an SPP $S \in E$ such that S has property $\hat{P}$; the PVPS PV has property P iff for every $E \in \mathcal{M}(PV)$, there is an SPP $S \in E$ such that S has property $\hat{P}$.

Example 4.6: Monotonicity is a property that can be defined in terms of SPP instances because every network

instance of a monotone algebra is equivalent to an SPP with rank functions λ nondecreasing on path extension.

Proposition 4.7: If an algebra A has a property P that can be defined in terms of SPP instances, then every PVPS PV consistent with A also has property P.

Proof: If PV is consistent with A, then $\mathcal{M}(PV) \subseteq \mathcal{X}(A)$; the result follows. ■

C. Mapping an Algebra to a General Path-Vector Policy System

Here we show how to use the path-vector-system framework to describe a protocol whose design specification is consistent with a given algebra. At its higher level of abstraction, an algebra describes the behavior of protocols on just a select part of the protocol's path data structure. This part of the data structure is modeled by the algebra's signature, which is used to determine weight; the actual rank of paths with the same weight is determined individually by nodes and is not modeled by the algebra.

To instantiate an algebra, we create a protocol whose data structure—the path descriptor—includes the components of the signature and an opaque[4] local-preference attribute that can be set at a node to differentiate paths of equal weight. The ranking function ω is defined to first weigh the signature components using the algebra function $f : \Sigma \to \mathcal{W}$ and then consider local preference.

Transformations to signatures are modeled by applying labels from $\mathcal{L}$ using the operator $\oplus$. As noted above, these transformations correspond to arc policy functions. Here we use one possible combination of node policies that yield arc policies that are consistent with the label transformations of the given algebra—we constrain export policies to be the identity function on path descriptors and constrain import policies to be those policy functions that correspond to labels. This choice does affect the implementation of the protocol and the individual expressiveness of import and export policies; in specific cases, it may be wise to change these constraints so that arc-policy functions still correspond to labels in $\mathcal{L}$ without artificially limiting the expressiveness of export of import policies (but we do not go into details here). This construction yields the following theorem.

Theorem 4.8: If $A = (\mathcal{W}, \preceq, \mathcal{L}, \Sigma, \phi, \oplus, f)$ is an algebra, the path-vector system PV with the following components is consistent with it (in particular, $\mathcal{X}(A) = \mathcal{M}(PV)$):

$$
\begin{aligned}
\mathcal{R} &= \Sigma \times \mathbb{N} \\
\mathcal{U} &= \mathcal{W} \times \mathbb{Z} \\
\leq_{\mathcal{U}} &= \preceq \times \leq \\
\omega &= (\sigma \in \Sigma,\ n \in \mathbb{N}) \mapsto (f(\sigma), -n) \\
\text{L}^{in}(F) &= [((\sigma', n') = F((\sigma, n))) \\
&\quad \Rightarrow (\exists\, l \in \mathcal{L} : \sigma' = l \oplus \sigma)] \\
\text{L}^{out}(F) &= \text{TRUE} \\
t^{in}(v, u, F, X) &= \{(\sigma', n') \mid ((\sigma', n') \in F(X)) \\
&\quad \wedge\ (\sigma' \neq \phi)\} \\
t^{out}(u, v, F, X) &= \{(\sigma, 0) \mid (\sigma, n) \in X \text{ is a loopless} \\
&\quad \text{path when extended to } v\} \\
\text{O}(X) &= [((\sigma, n) \in X) \Rightarrow (\sigma \in \Sigma)]
\end{aligned}
$$

Proof: Note that by construction, PV allows exactly the same paths as A.

Suppose $E \in \mathcal{M}(PV)$. Then for all SPP $S \in E$: (I) $\lambda(P) = \lambda(P')$ iff $\omega(r(P)) = \omega(r(P'))$ and (II) $\lambda(P) < \lambda(P')$ iff $\omega(r(P)) < \omega(r(P'))$. We use these properties and the definition of ω to show that $E(S) \in \mathcal{X}(A)$.

For one of these SPPs S, by the definition of ω, $\omega(r(P)) = \omega(r(P')) \Rightarrow f(s(P)) = f(s(P'))$. Thus, by (I), $\lambda(P) = \lambda(P') \Rightarrow \omega(r(P)) = \omega(r(P'))$ and so $f(s(P)) = f(s(P)) = f(s(P'))$. ω is such that $\omega(r(P)) < \omega(r(P)) \Rightarrow f(s(P)) \preceq f(s(P'))$. Thus, by (II), $\lambda(P) < \lambda(P') \Rightarrow \omega(r(P)) < \omega(r(P'))$ and hence $f(s(P)) \preceq f(s(P'))$. By the definition of ω, $f(s(P)) \prec f(s(P')) \Rightarrow \omega(r(P)) < \omega(r(P'))$, so by (II) we have $\lambda(P) < \lambda(P')$. Conditions (i)–(iii) of Def. 4.3 are thus satisfied and $E(S) \in \mathcal{X}(A)$. ■

D. Describing a General Path-Vector Policy System with an Algebra

The path-vector-system specification contains a definition of the path data structure, the elements involved in ranking, and constraints on policies. We can easily construct an algebra that has the same specification components.

To do this, let the sets Σ and $\mathcal{W}$ of signatures and weights be the set $\mathcal{R}$ of path descriptors and the ranking set $\mathcal{U}$, respectively. A natural way to construct labels that correspond to arc policy functions $f_{(u,v)}(r)$ is simply to list the arguments other than r—the node names u and v and the policy functions $F^{in}_{(v,u)}$ and $F^{out}_{(u,v)}$—to the policy application functions from the definition of $f_{(u,v)}(r)$. Labels then contain sufficient information so that $\oplus$ may used to recover $f_{(u,v)}(r)$ as $l \oplus r$, with ϕ used in place of filtered routes ($f_{(u,v)}(r) = \emptyset$).

Theorem 4.9: Given a path-vector system $PV = (\mathcal{D},\ \mathcal{R},\ \mathcal{U},\ \leq_{\mathcal{U}},\ \text{L}^{in},\ \text{L}^{out},\ \text{O},\ t^{in},\ t^{out})$, let

$$L(u, v, f_1, f_2) = X \mapsto t^{in}(v, u, f_1, t^{out}(u, v, f_2, X)),$$

[4]An opaque attribute is essentially one that is replaced by a null value when a descriptor is advertised to a neighbor so that the information in it is kept private.

i.e., $L(l)$ is the arc-policy function determined by the tuple $l = (u, v, f_1, f_2)$, where $u, v \in \mathbb{N}$ represent node identifiers defining a signaling edge (u, v) and $f_1, f_2 \in 2^{\mathcal{R}}$ are the import and export policies along that edge. (These tuples are members of the constructed label set $\mathcal{L}$ below.) Then the algebra with the following components is consistent with PV:

$$
\begin{aligned}
\mathcal{W} &= \mathcal{U} \\
\preceq &= \leq_{\mathcal{U}} \\
\Sigma &= \mathcal{R} \\
\phi &= \text{signature for filtered routes} \\
\mathcal{L} &= \mathbb{N} \times \mathbb{N} \times \{F \in 2^{\mathcal{R}} \mid \mathrm{L}^{in}(F)\} \\
&\quad \times \{F \in 2^{\mathcal{R}} \mid \mathrm{L}^{out}(F)\} \\
\oplus &= l \oplus (r \in \mathcal{R}) \mapsto \begin{cases} L(l)(r), & L(l)(r) \neq \emptyset \\ \phi, & L(l)(r) = \emptyset \end{cases} \\
f &= \omega
\end{aligned}
$$

Proof: (Sketch) By definition, this algebra allows exactly the same paths as PV. Suppose $E \in \mathcal{M}(PV)$. Then for every SPP $S \in E$ we have (I) and (II) from the proof of Thm. 4.8. Using the definitions of f, $\oplus$, and $\mathcal{L}$ we may show that $E(S) \in \mathcal{X}(A)$, so PV is consistent with the algebra A constructed this way. ■

Although we omit the details here, it is clear that an algebra constructed in this way for an increasing path-vector system will be strict monotone, because the weighing function is just the increasing rank function ω.

This construction errs on the side of being too specific. The algebra here is defined such that it models protocol components that are consistent with only minor tweaks of the original path-vector system. The algebra is meant to model a subset of path-data-structure components so that their role can be analyzed; here, we include all the components in the signature. By doing so, we can always construct an algebra that is consistent with the protocol, but not necessarily one that yields generally applicable results. Depending on the protocol, we note that it may be more useful to construct an algebra restricted to some subset of the rank criteria (rather than the entire path descriptor), because the properties of that algebra would apply to a greater number of protocol implementations than those of the construction above. The intuition used behind mapping the components, *i.e.*, the informal correspondence the first and third columns of Table IV, would still apply.

E. Algebras and Class-Based Systems

An application of the path-vector-system framework is the class-based path-vector system (Section 8 of [4]). These systems describe protocols that separate rank criteria encoded into two attributes: (1) the level attribute, a shared, nondecreasing integer, which takes precedence in ranking; and (2) the local-preference attribute, an opaque integer. The level attribute naturally corresponds to the shared component of a path data structure used as the primary rank criterion, modeled by signature and weight in an algebra, while the local preference naturally corresponds to the local settings used to break ties among paths with the same weight. The class-based systems also have another mechanism for constraining the ranking order on paths: directed edges in the network signaling graph have *class assignments* that identify the relationship between two nodes; these assignments naturally correspond to labels.

The translation from algebra to class-based systems is not straightforward, however. Because the level attribute is nondecreasing, all protcols described by class-based systems must be monotonic (thus, we cannot translate non-monotonic algebras). Furthermore, the encoding from signature components to the level-attribute integer is only clear if the weight set is countable and the weighing function f is invertible. In this case, there is a natural bijection ψ from weights to the natural numbers, and then $\sigma = f^{-1}(\psi^{-1}(g))$ is the signature σ corresponding to the level value g; the level-attribute g can be used to encode the signature σ in a path descriptor. Labels then correspond to increments in level-attribute values: If $l \oplus \sigma = \sigma'$, then l corresponds to the increment $\psi(f(\sigma')) - \psi(f(\sigma))$. However, the class structure may provide a better way of describing the constraints on the system than this encoding does, and there is no known general way to generate a set of classes for an algebra.

In the other direction, we note that a consistent algebra can be constructed to describe a class-based system where there is a total relative-preference order among the classes: The weights are (level attribute, class) pairs; the weighing function ranks paths based on level attribute first, then on the class assignment for the next-hop edge; labels are increments to level-attribute values and class assignments for the edge; and the operator $\oplus$ changes the signature of paths using the labels in accordance with the relative-preference and scoping rules of the class-based system. This mapping will not work, however, if the classes are not totally ordered in relative-preference. An example of a total order among classes is the structure motivated by [2], in which neighbors are assigned one of three business relationship labels—'customer', 'provider', or 'peer'—and routes are chosen such that customer routes are most preferred, then peer routes, then provider routes.

If the class-structure design principles are ignored, the following algebra is consistent with any path-vector system because it models only the level attribute. The

integers below correspond to level-attribute values (label integers correspond to level-attribute increments). The algebra is monotonic because the level attribute is the primary rank criterion and nondecreasing.

Theorem 4.10: Any class-based path-vector system is consistent with the following monotonic algebra:

$$\begin{array}{rcl} \mathcal{W} & = & \mathbb{N} \\ \preceq & = & \leq \\ \Sigma & = & \mathbb{N} \\ \phi & = & \text{signature for filtered routes} \\ \mathcal{L} & = & \mathbb{N} \cup \{\phi\} \\ \oplus & = & n_1 \oplus n_2 \mapsto \begin{cases} n_1 + n_2, & n_1, n_2 \neq \phi \\ \phi, & n_1 \text{ or } n_2 = \phi \end{cases} \\ f & = & \text{identity} \end{array}$$

Proof: (Sketch) Use (I) and (II) from the proof of Thm. 4.8 and other arguments as above to show that $E \in \mathcal{X}(A)$ for every $E \in \mathcal{M}(PV)$. ■

V. Equivalence of Design Guidelines

The two design properties modeled by both frameworks are robustness and optimality, and we discuss these below. As optimality was not considered in the original work on PVPSes, Sect. V-B illustrates the utility of our translations in expressing properties in different frameworks and in obtaining results in one framework using analysis in the other.

A. Monotonicity and Robustness

The main result from both frameworks is that if paths increase in absolute rank when they are extended from router to router, the protocol is guaranteed to converge on any network. Here we show that these results are compatible. We start with the following, a composition of results in [4]:

Theorem 5.1: An increasing PVPS is robust.

This result is meaningful at the protocol level of abstraction, and it is consistent with the following extension of Proposition 4 in [8]:

Theorem 5.2: Any protocol consistent with either a strict-monotone algebra or a monotone algebra with shortest-paths tie-breaking is robust.

If either one of these theorems is combined with the translation between models given in Section IV, the other theorem directly follows. (The equivalence proofs are omitted due to lack of space.) This shows that these results are equivalent; this is intuitively clear because any instance of a strict-monotone algebra can be described by an increasing path-vector system.

We now address some other results.

Theorem 5.3: Any protocol consistent with a monotone algebra will converge robustly on free networks.

The constraint in Sect. 8 of [4] regarding level-equality cycles for class-based systems is a specialized version of this theorem, which is a restatement of Prop. 3 in [8]. In addition, because freeness can be checked in time proportional to a polynomial in the number of labels, signatures, and edges of a graph [8], freeness is a feasible network-level global constraint guaranteeing the robustness of any protocol consistent with a monotone algebra. The above theorem can be proven (omitted due to lack of space) in the path-vector-system framework using techniques similar to the proofs of Lem. 8.5 and 8.6 in [4]. These proofs can be used directly if the translation guidelines in Sect. IV-E are used to instantiate a monotonic algebra as a class-based system.

The theorem below captures one direction of Prop. 5 in [8]; however, it uses our translation between levels of abstraction to state and prove the result more precisely.

Theorem 5.4: If an algebra is not monotone, there exists a network instance on which some protocol consistent with the algebra does not converge; *i.e.,* there exists some path-vector system consistent with every non-monotone algebra that is not robust.

Proof: (Sketch) Sect. 6.2 in [8] essentially proves this result by constructing a permitted SPP instance (DISAGREE from [5]) on which some consistent protocol would diverge (a non-robust SPP instance). This proof method applies directly to the path-vector-system framework: Given the algebra, construct a consistent protocol PV using Thm. 4.8; the same SPP instance used in the original proof is contained in one of the equivalence classes in $\mathcal{M}(PV)$. Because the SPP instance is not robust, PV is, by definition, not robust. ■

In summary, both the algebra and PVPS models can isolate properties to analyze convergence, and the sufficient conditions are equivalent. The properties translate by Prop. 4.7.

B. Isotonicity and Optimality

Sobrinho [8] effectively shows that while convergence depends on monotonicity, optimal convergence depends on isotonicity. Formally, we have the following result that combines Prop. 1, 2, and 5 from [8].

Proposition 5.5: A protocol that converges robustly and optimally is consistent with some monotone and isotone algebra. Furthermore, there is some protocol consistent with any monotone and isotone algebra that converges robustly and optimally.

Optimality is not considered in [4], so we define the following additional properties for the path-vector-system framework to translate the results from [8].

PV-Isotonicity $\quad \forall_{(u,v)\in E} \forall_{r_1,r_2 \in \mathcal{R}}\ \omega(r_1) \leq \omega(r_2)$
$\Rightarrow \omega(f_{(u,v)}(r_1)) \leq \omega(f_{(u,v)}(r_2))$

COHERENCE $\forall_{P,\,\text{loop}\,Q}\ \omega(r(P)) \leq \omega(r(QP))$

Coherence is an instance property because it depends on the graph structure, and that PV-isotonicity is both a property of a PVPS and of a network (in this case, the condition holds for all edges in the network rather than all possible arc-policy functions). A PVPS that is PV-isotone has only isotone instances and is consistent with an isotone algebra; a PVPS that is not PV-isotone has at least one instance where arc policy functions violate the PV-isotonicity condition (instantiation by Prop. 4.7.)

The following propositions about PVPSes are then analogous to Prop. 1 and 2 in [8]. As noted above, their statements and proofs (omitted due to lack of space) follow from applying our translation between algebras and PVPSes to the corresponding results for algebras [8].

Proposition 5.6: If a PVPS is PV-isotone, then for a network with coherent policies, there exists an optimal path from any node u to a destination d such that any subpath with destination d is optimal on its own.

Proposition 5.7: If a PVPS is PV-isotone, then any coherent instance has an optimal solution.

Proposition 5.8: If a PVPS is not PV-isotone, then:

1) there exists an instance with no optimal solution;
2) for any non-isotone instance I, there exists another instance I' generated by filtering a subset of permitted routes in I such that I' has no optimal solution.

In summary, the optimality results from [8] can be stated using the PVPS framework, again allowing analysis for this property using either level of abstraction during the protocol-design process.

VI. CONCLUSIONS AND FUTURE WORK

We have established the relationship between the two recently published formal models for path-vector routing, the algebra framework [8] and the PVPS framework [4]. In doing so, we provided several theorems that constructively translate between design specifications in the algebra framework and protocol specification in the PVPS framework. Our translation preserves several important properties, *e.g.*, (strict) monotonicity (or increasing systems), isotonicity, and coherence, that are relevant to studying protocol convergence in both frameworks. This translation thus aids the design process by giving the protocol designer the ability to study protocol properties at several levels of abstraction using languages naturally suited to these different levels. The combination of results from the two frameworks, translated and summarized here, and the intuitive relationship between these frameworks that we have provided allows for a more complete analysis of path-vector protocol design and configuration.

There are remaining open questions related to applying these frameworks. We believe that our equivalence results further motivate studying routing protocols from the design perspective in order to find a good balance between local constraints and global constraints and between expressiveness and robustness. This may lead to insight in considering configuration languages for the PVPS framework.

In addition, the frameworks can be applied to more interesting examples of protocols than simplified versions of BGP. The convergence results may prove useful in analyzing path-vector protocols used for intra-domain routing (*e.g.*, IGRPs) and the interaction between intra-domain and inter-domain routing. The frameworks should be equipped to deal with the additional complexity of these scenarios, but, as we have discussed in this paper, meaningful results may depend on the selection of an appropriate subset of the path data (or protocol behavior) to analyze. We give general constructions for framework design specifications given a particular protocol, but this approach may be more general than needed; application of the theory developed here to additional specific examples would be interesting and helpful.

Finally, this work models only eBGP. While the guidelines *do* apply to iBGP signaling correctness inside one AS, the question of using these frameworks to model the interaction between eBGP and iBGP remains open.

REFERENCES

[1] Cisco Field Note. Endless BGP Convergence Problem in Cisco IOS Software Releases. October 2001. `http://www.cisco.com/warp/public/770/fn12942.html`

[2] L. Gao, T. G. Griffin, and J. Rexford. Inherently Safe Backup Routing with BGP. In *Proc. IEEE INFOCOM 2001*, 1:547–556, April 2001.

[3] L. Gao and J. Rexford. Stable Internet Routing without Global Coordination. In *Proc. ACM SIGMETRICS*, pp. 307–317, June 2000.

[4] T. G. Griffin, A. D. Jaggard, and V. Ramachandran. Design Principles of Policy Languages for Path-Vector Protocols. Originally published in *Proc. ACM SIGCOMM'03*, pp. 61–72, August 2003. Full version available as Yale Tech. Report YALEU/DCS/TR-1250. `ftp://ftp.cs.yale.edu/pub/TR/tr1250.{ps,pdf}`

[5] T. G. Griffin, F. B. Shepherd, and G. Wilfong. The Stable Paths Problem and Interdomain Routing. *IEEE/ACM Transactions on Networking*, 10(2):232–243, April 2002.

[6] D. McPherson, V. Gill, D. Walton, and A. Retana. BGP Persistent Route Oscillation Condition. Manuscript, 2002.

[7] Y. Rekhter and T. Li. A Border Gateway Protocol. RFC 1771 (BGP version 4), 1995.

[8] J. Sobrinho. Network Routing with Path Vector Protocols: Theory and Applications. In *Proc. ACM SIGCOMM'03*, pp. 49–60, August 2003.

[9] K. Varadhan, R. Govindan, and D. Estrin. Persistent Route Oscillations in Inter-domain Routing. *Computer Networks*, 32:1–16, 2000.