Asynchronous Convergence of Policy-Rich Distributed Bellman-Ford Routing Protocols

Matthew L. Daggitt
University of Cambridge
mld46@cam.ac.uk

Alexander J.T. Gurney
Comcast
alexander_gurney@cable.comcast.com

Timothy G. Griffin
University of Cambridge
tgg22@cam.ac.uk

ABSTRACT

We present new results in the theory of asynchronous convergence for the Distributed Bellman-Ford (DBF) family of routing protocols which includes distance-vector protocols (e.g. RIP) and path-vector protocols (e.g. BGP). We take the *strictly increasing* conditions of Sobrinho and make three main new contributions.

First, we show that the conditions are sufficient to guarantee that the protocols will converge to a *unique* solution, preventing the possibility of BGP wedgies. Second, we decouple the computation from the asynchronous context in which it occurs, allowing us to reason about a more relaxed model of asynchronous computation in which routing messages can be lost, reordered, and duplicated. Third, our theory and results have been fully formalised in the Agda theorem prover and the resulting library is publicly available for others to use and extend. This is in line with the increasing emphasis on formal verification of software for critical infrastructure.

ACM Reference Format:
Matthew L. Daggitt, Alexander J.T. Gurney, and Timothy G. Griffin. 2018. Asynchronous Convergence of Policy-Rich Distributed Bellman-Ford Routing Protocols. In *SIGCOMM '18: SIGCOMM 2018, August 20–25, 2018, Budapest, Hungary.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3230543.3230561

1 WHAT IS POLICY-RICH ROUTING?

This paper investigates the theory of asynchronous convergence for routing protocols in the Distributed Bellman-Ford (DBF) family. This includes distance-vector (RIP-like) and path-vector (BGP-like) protocols. In particular our models include what we call *policy-rich* protocols, and so we begin by informally explaining this terminology.

Suppose that a router participating in a DBF computation has a best route r to a destination d. If it then receives a route r' for d from an immediate neighbour it will first apply some policy f associated with that neighbour and produce a candidate route $f(r')$. It will then compare r with $f(r')$ to determine which is best. Let us denote the outcome of this selection process as $\text{best}(f(r'),\ r)$. A very simple example is the shortest-paths problem where r is just an integer representing distance, $f_w(r) = w + r$ for some weight w, and $\text{best}(r,\ r') = \min(r,\ r')$.

If we dig deeply into the classical theory underlying best-path algorithms — both synchronous [1, 9], and asynchronous [3] — we find that it always assumes the following equation, or something equivalent, must hold:

$$f(\text{best}(r_1,\ r_2)) = \text{best}(f(r_1),\ f(r_2)) \tag{1}$$

This property is referred to as *distributivity*. The left-hand side of the equation can be interpreted as a decision made by a router sending routes r_1 and r_2 while the right-hand side is a decision made by the neighbour receiving those routes. Assuming the equality holds, the classical theory proves that routing protocols arrive at *globally optimal routes* — the best routes over all possible paths.

By a *policy-rich language* we mean one in which distributivity does not hold. A clear example of how distributivity violations might arise in routing can be seen in the use of *route maps* which are functions (scripts) that take routes as input and return routes as output. For example, if g and h are route maps, then we might define another route map f as:

$$f(r) = \text{if } P(r) \text{ then } g(r) \text{ else } h(r), \tag{2}$$

where P is a predicate on routes (such as "does this route contain the BGP community 17?"). To see how easily distributivity can be violated, suppose that:

$$\begin{array}{lcl} P(a) & = & \text{true}, \\ P(b) & = & \text{false}, \\ \text{best}(a,\ b) & = & a, \\ \text{best}(g(a),\ h(b)) & = & h(b). \end{array}$$

Then the left-hand side of Eq 1 is:

$$f(\text{best}(a,\ b)) = f(a) = g(a),$$

while the right-hand side becomes:

$$\text{best}(f(a),\ f(b)) = \text{best}(g(a),\ h(b)) = h(b)$$

For Eq 1 to hold we need $g(a) = h(b)$, which may not be the case (indeed, if $g(a) = h(b)$ were always true, then there would be no point in defining f!). Perhaps the most common example of such conditional policies is *route filtering*, where $h(r)$ is equal to the invalid route.

A specialist schooled in the classical theory of path finding might be tempted to forbid the use of such "broken" policies when using the Bellman-Ford or Dijkstra algorithms. Yet, once again, practice has outstripped theory. The Border Gateway Protocol (BGP) [24] — a key component of the Internet's infrastructure — is a policy-rich routing protocol.

We argue that today's BGP is in fact broken since it is possible to write policies that can result in anomalous behaviour such as non-convergence [22, 28] and multiple stable states [11]. The latter are problematic as the extra stable states are nearly always unintended and often violate the intent of policy writers. Leaving an unintended stable state requires manual intervention and, in the worst case, a high degree of coordination between competing networks.

Hence a pertinent question is how to *tame* the policy language of BGP-like protocols to ensure good behaviour? Of course we could mandate that all protocols must conform to Eq 1. However, we would then not be able to implement typical inter-domain policies that are based on commercial relationships [16, 17]. In addition, something as simple as shortest-path routing with route filtering would be banned.

Gao & Rexford [8] showed that the simple commercial relationships described in [16, 17], if universally followed, are enough to guarantee convergence. However their model is BGP-specific and gives us no guidance on how policy-rich protocols should be constrained in general. Furthermore their conditions impose constraints on the topology of the network and therefore they require on-going verification.

Instead a different middle ground for generic policy-rich protocols has been achieved for both Dijkstra's algorithm [26] and the DBF family [25]. Rather than insisting on distributivity, we require that for all routes r and policies f we have:

$$r = \text{best}(r,\ f(r)). \tag{3}$$

In other words, applying policy to a route cannot produce a route that is more preferred. Although Eq 3 is sufficient for Dijkstra's Algorithm [26], it must be strengthened for DBF algorithms [25] to:

$$r = \text{best}(r,\ f(r)) \neq f(r). \tag{4}$$

That is, applying policy to a route cannot produce a route that is more preferred *and* it cannot leave a route unchanged. We call such policy languages *strictly increasing*. Note that if policies g and h are strictly increasing, then the conditional policy f defined in Eq 2 is also strictly increasing. In other words, a strictly increasing policy language remains strictly increasing when route maps are introduced.

However, without distributivity we can no longer achieve *globally optimal* routes and so we must be content with *locally optimal* routes [26] – each router obtains the best routes possible given the best routes revealed by its neighbours.

In the case of BGP it is natural to ask if the strictly increasing condition is too strong and prohibits desirable policies? We feel that it may be the best available topologically-independent requirement. Sobrinho [25] has shown that the strictly increasing condition is more general than the Gao-Rexford conditions, by incorporating them into a strictly increasing framework. The same paper also implies that generalising the strictly increasing condition further causes the time required to verify the conditions to become exponential in the size of the network and hence infeasible to check.

1.1 Related work

We will discuss three main prior works: Griffin, Shepherd & Wilfong [12], Gao & Rexford [8] and Sobrinho [25]. Each of these prove theorems about the conditions needed for some form of convergence for path-vector protocols.

To place these papers in context and highlight the gaps that exist we now list what we think are the main desirable properties of such theorems:

(1) They should guarantee convergence from any starting state. This ensures re-convergence even after arbitrary changes to the network topology.
(2) The same final state should always be reached, no matter what state the network is in initially and what order the asynchronous events happen.
(3) The proofs should be as general as possible. Ideally they should apply not just to BGP, but to a broad range of current and future vector-based protocols.
(4) The conditions should be efficiently verifiable. By this we mean that given access to the network configuration, the time needed to verify if the conditions hold should be polynomial time in the size of the network.

The work of Griffin, Shepherd & Wilfong [12] takes general routing problem instances, and compiles them down to *stable path problems*. Using these, they then demonstrate necessary and sufficient conditions for convergence. However the results do not guarantee convergence from arbitrary states, only from a clean state, and verification takes exponential time. They do however prove that the final state is unique. Therefore they achieve points 2 & 3 but not 1 & 4.

Gao & Rexford [8] give conditions that guarantee convergence for a model of BGP. They prove that these are sufficient to guarantee convergence from an arbitrary state, but not necessarily to the same state each time. The conditions are efficiently verifiable. Therefore they achieve points 1 & 4 but not 2. With respect to point 3, we argue that there is considerable scope for generalisation. Their conditions are very strong, and impose constraints not just on the protocol, but on the topology of the network itself which must be partially ordered with respect to a customer-provider relationship. This means that it is necessary to re-verify the conditions each time the network topology changes. Their proof is specific to BGP and does not apply to other DBF protocols. Even within BGP adding additional features, such as back-up routes, requires re-proving key parts of the theorems in their entirety. As part of BGP, they also assume in-order, reliable delivery of messages.

Sobrinho [25] provided an important turning point in the theory by introducing algebraic models that are at a higher-level of abstraction than previous work. This new approach allows the paper to reason about general path-vector protocols, rather than one specific protocol. Sobrinho shows that if the algebra is *strictly increasing* [1] then convergence is guaranteed from arbitrary starting states to some final stable solution. The increasing property is efficiently verifiable as it is built-in to the algebra. Therefore like Gao & Rexford, the work achieves points 1 & 4 but not 2. With respect to 3, the work is far more general as it uses a generic model for path-vector protocols. However further generalisation is still possible as the paper assumes in-order, reliable delivery.

1.2 Our contributions

Our primary contribution is taking the *strictly increasing* condition of Sobrinho and proving a new convergence theorem that satisfies points 1, 2, 3 & 4 concurrently. Beyond this, our work has two other novel contributions: 1) a factorisation of the asynchronous environment from the synchronous algorithm and 2) a machine-checked library of proofs that can be reused by other researchers.

It is well known that reasoning about asynchronous processes is far harder than their synchronous counterparts. The related works in Section 1.1 each develop their own model of asynchronous computation that interacts in complicated ways with the essential actions of the policy-based protocol.

An improved approach can be found in Üresin & Dubois' work on asynchronous iterative algorithms [27]. They prove that if the *synchronous* algorithm obeys their ACO conditions then this is sufficient to guarantee the convergence of the asynchronous version of the algorithm. In the context of routing protocols, this cleanly separates the underlying routing problem from the distributed environment in which we are solving it. Furthermore their asynchronous model allows messages to be delayed, lost, reordered and duplicated, allowing us to prove for the first time that in-order reliable delivery is not required for convergence.

[1]Sobrinho uses the term "monotonic" to refer to the property we call "increasing". When combined with his assumptions about paths, his model can be shown to be "strictly increasing". See Section 5.1 for a discussion.

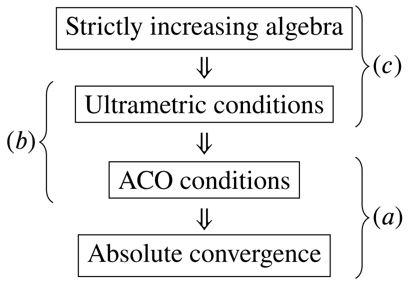

Figure 1: Putting it all together. Arrows indicate logical implications. (a) **is from Üresin & Dubois [27].** (b) **is from Gurney [15].** (c) **is presented in this paper. Our Agda formalisation [6] covers** (a)**,** (b) **and** (c)**. See [30] for a discussion of our formalisation of** (a) **and** (b)**.**

The unpublished work of Gurney [15] builds on Üresin & Dubois' work by showing that the existence of a particular type of ultrametric implies the ACO conditions. Using this, we need only define an ultrametric that correctly interacts with the synchronous routing computation to guarantee the convergence of the asynchronous computation. Figure 1 illustrates how all of the pieces fit together.

We have fully formalised our work in Agda [23], a theorem-proving language which captures much of constructive mathematics and in which both proofs and programs can be written. The formalisation includes the asynchronous theory of Üresin & Dubois and Gurney, and we discuss this preparatory work in [30]. This formalisation is in line with recent trends in the verification of infrastructure-related software which is becoming increasingly urgent in areas as diverse as operating systems [18], compilers [19, 21] and networking [14, 29].

Additionally the generality of our proofs means that the formalisation can also form the basis of future work. The proofs are freely available [6], and we hope that it will prove useful to the community. In particular, by constructing a model of a particular path-vector routing protocol in Agda and showing that it satisfies the increasing property, it is possible to show that the protocol is well-behaved. In Section 7 we outline how to do this for a routing algebra with an expressive policy language that is safe-by-design.

Although our results are fully formalised, this paper presents them in the traditional style: informal but rigorous.

1.3 Generality vs implementation details

In any mathematical model of a real-world system, there is a fundamental tension between capturing the most general form of the system and capturing every detail of the implementation. More implementation details can always be added to better model a particular system, at the cost of losing the ability to apply the work more generally.

In this paper we err on the side of generality. While we acknowledge that BGP is the dominant policy-rich path-vector protocol, the aim of our work is not only to inform how BGP policies may be constrained to provide better guarantees, but also to guide the design of future policy-rich protocols.

For example we do not explicitly model the hard-state nature of BGP. Instead we use a very general model of asynchronous computation which we believe can easily capture a hard-state protocol (this intuition is not fully formalised). Had we explicitly modelled a hard-state protocol, it would prevent us from applying our work to soft-state protocols as well.

1.4 Road map

In Section 2.1 we formalise routing preferences and policy using an algebraic approach inspired by that of Sobrinho [25]. In Sections 2.2 & 2.3 we model the synchronous DBF computation as the repeated iteration of matrix operations. This approach provides a clear, implementation independent specification of the problem we are solving: finding a fixed point of this iteration. In Section 3 we review the model of asynchronous computation from Üresin & Dubois [27] and Gurney [15]. Section 4 applies the theory to distance-vector protocols, while Section 5 applies the theory to path-vector protocols. We briefly describe our Agda formalisation in Section 6. In Section 7 we give an example of a safe-by-design algebra.

2 ALGEBRAIC MODEL

We now present our model of distance-vector and path-vector routing protocols.

2.1 Routing algebras

Definition 1. A *routing algebra* is a tuple $(S, \oplus, F, \overline{0}, \overline{\infty})$ where:

- S is the set of routes
- $\oplus : S{\times}S \to S$ is the choice operator, which given two routes returns the preferred route (this was informally referred to as *best* in the introduction).
- F is the set of edge weights. Each edge weight is a function $f : S \to S$, which when given a route returns that route extended by the edge.
- $\overline{0} \in S$ is the trivial route from any node to itself.
- $\overline{\infty} \in S$ is the invalid route.

We assume that these structures have the following minimal properties (see Table 1 for rigorous algebraic definitions):

- $\oplus$ is associative and commutative (*the order in which routes are chosen between is irrelevant*).
- $\oplus$ is selective (*choosing between two routes always returns one of the two routes*).
- $\overline{0}$ is an annihilator for $\oplus$ (*the trivial route is always preferred to any other route*).
- $\overline{\infty}$ is an identity for $\oplus$ (*all routes are preferred to the invalid route*).
- $\overline{\infty}$ is a fixed point for all $f \in F$ (*the extension of the invalid route is also the invalid route*).

Property	Definition
$\oplus$ is associative	$a \oplus (b \oplus c) = (a \oplus b) \oplus c$
$\oplus$ is commutative	$a \oplus b = b \oplus a$
$\oplus$ is selective	$a \oplus b \in \{a,\ b\}$
$\overline{0}$ is an annihilator for $\oplus$	$a \oplus \overline{0} = \overline{0} = \overline{0} \oplus a$
$\overline{\infty}$ is an identity for $\oplus$	$a \oplus \overline{\infty} = a = \overline{\infty} \oplus a$
$\overline{\infty}$ is a fixed point for $f \in F$	$f(\overline{\infty}) = \overline{\infty}$
F is increasing over $\oplus$	$a \leq f(a)$
F is strictly increasing over $\oplus$	$a \neq \overline{\infty} \ \Rightarrow\ a < f(a)$
F distributes over $\oplus$	$f(a \oplus b) = f(a) \oplus f(b)$

Table 1: Definitions of routing algebra properties. The first set are required, while the second set are optional.

S	$\oplus$	F	$\overline{\infty}$	$\overline{0}$	Use
$\mathbb{N}_\infty$	min	F_+	∞	0	shortest paths
$\mathbb{N}_\infty$	max	F_+	0	∞	longest paths
$\mathbb{N}_\infty$	max	$F_{\min}$	0	∞	widest paths
[0, 1]	max	$F_\times$	0	1	most reliable paths

Table 2: A few examples of some simple routing algebras. $F_\otimes = \{f_s(a) = s \otimes a \mid s \in S\}$ for arbitrary operator $\otimes$ e.g. $F_+ = \{f_s(a) = s + a \mid s \in \mathbb{N}_\infty\}$.

Table 2 provides examples of some routing algebras that solve some simple best-path problems.

Using the choice operator $\oplus$ we can define an ordering over routes as follows:

$$a \leq b \ \triangleq\ a \oplus b = a$$
$$a < b \ \triangleq\ a \leq b \wedge a \neq b.$$

As $\oplus$ is associative, commutative and selective, we have that $\leq$ is a total/linear order. For all routes a we have $\overline{0} \leq a \leq \overline{\infty}$.

Definition 2. A routing algebra is *increasing* if for all f and a we have that $a \leq f(a)$.

Definition 3. A routing algebra is *strictly increasing* if for all f and a (with the exception of $\overline{\infty}$) we have that $a < f(a)$.

Definition 4. A routing algebra is *finite* if the set S is finite.

2.2 What problem are we solving?

Given a network of n nodes with weighted edges, we represent the topology by an $n \times n$ adjacency matrix $\mathbf{A}$ where $\mathbf{A}_{ij} \in F$ is the weight of the edge from i to j. Missing edges are represented by the constant function $f(a) = \overline{\infty}$.

Let $\mathbb{M}_n(S)$ be the set of $n \times n$ matrices over S. The global routing state is represented as a matrix $\mathbf{X} \in \mathbb{M}_n(S)$. The row $\mathbf{X}_i$ is therefore node i's routing table and so the element $\mathbf{X}_{ij}$ is i's best current route to node j. We define the *sum* of two matrices $\mathbf{X}$ and $\mathbf{Y}$ as:

$$(\mathbf{X} \oplus \mathbf{Y})_{ij} \triangleq \mathbf{X}_{ij} \oplus \mathbf{Y}_{ij}$$

and the *application* of $\mathbf{A}$ to $\mathbf{X}$ as:

$$\mathbf{A}(\mathbf{X})_{ij} \triangleq \bigoplus_k \mathbf{A}_{ik}(\mathbf{X}_{kj})$$

In one *synchronous* round of a distributed Bellman-Ford computation every node in the network propagates its routing table its neighbours who then update their own tables accordingly. We model this operation as σ:

$$\sigma(\mathbf{X}) \triangleq \mathbf{A}(\mathbf{X}) \oplus \mathbf{I}$$

where $\mathbf{I}$ is the *identity matrix*:

$$\mathbf{I}_{ij} = \begin{cases} \overline{0} & \text{if } i = j \\ \overline{\infty} & \text{otherwise} \end{cases}$$

The nature of the underlying computation becomes clearer when looking at a single element of $\sigma(\mathbf{X})$:

$$\begin{aligned} \sigma(\mathbf{X})_{ij} &= \left(\bigoplus_k \mathbf{A}_{ik}(\mathbf{X}_{kj}) \right) \oplus \mathbf{I}_{ij} \\ &= \begin{cases} \overline{0} & \text{if } i = j \\ \bigoplus_k \mathbf{A}_{ik}(\mathbf{X}_{kj}) & \text{otherwise} \end{cases} \end{aligned} \tag{5}$$

Node i's new route to j is the best choice out of the extensions of the routes offered to it by each of its neighbours k.

Lemma 1. After an iteration, a node's route to itself is always the trivial route (i.e. $\forall i : \sigma(\mathbf{X})_{ii} = \overline{0}$).

Proof. See Eq 5 above. □

2.3 Synchronous computation

We model the synchronous Distributed Bellman-Ford computation as repeated applications of σ:

$$\begin{aligned} \sigma^0(\mathbf{Y}) &\triangleq \mathbf{Y} \\ \sigma^{k+1}(\mathbf{Y}) &\triangleq \sigma(\sigma^k(\mathbf{Y})) \end{aligned}$$

Definition 5. A state, $\mathbf{X}$, is *stable* if it is a fixed point for σ:

$$\sigma(\mathbf{X}) = \mathbf{X}$$

This is equivalent to saying that no node can improve its selected routes by unilaterally choosing to switch. Therefore such a state is a *local* but not necessarily a *global* optimum.

The computation σ converges synchronously from an arbitrary state $\mathbf{X}$ if there exists a time t such that:

$$\sigma^{t+1}(\mathbf{X}) = \sigma^t(\mathbf{X})$$

where $\sigma^t(\mathbf{X})$ is necessarily a stable state.

3 ASYNCHRONICITY

So far we have defined a synchronous model where update messages between nodes are exchanged instantaneously and in parallel. However in reality nodes dispatch update messages asynchronously and they may be delayed, reordered, duplicated or even lost along the way. Reasoning about the outcomes of such unpredictable events is known to be extremely challenging. To help tame this complexity, we now describe an established mathematical model of such behaviour.

3.1 A model of asynchronicity

We use the model from Üresin & Dubois [27] which assumes a discrete and linear notion of time $\mathbb{T}$ denoting the times of events of interest in the network.

Definition 6. A *schedule* consists of a pair of functions:

- $\alpha : \mathbb{T} \to 2^V$ is the *activation function*, where $\alpha(t)$ is the set of nodes which update their routing table at time t.
- $\beta : \mathbb{T} \times V \times V \to \mathbb{T}$ is the *data flow function*, where $\beta(t, i, j)$ is the time at which the information used by node i at time t was sent by node j.

where V is the set of nodes in the network, such that:

S1 : every node continues to activate indefinitely

$$\forall it. \exists k.\ i \in \alpha(t + k)$$

S2 : information only travels forward in time

$$\forall ijt.\ \beta(t, i, j) < t$$

S3 : no link has a loss rate of 100%

$$\forall ijt. \exists t'. \forall k.\ \beta(t' + k, i, j) \neq t$$

Note that at first glance S1 seems to preclude node failure in the network and S3 seems to preclude link failure. We discuss why this is not the case in Section 3.2.

This is a very weak model of asynchronous communication. Nothing in S2 or S3 forbids the data flow function β from delaying, losing, reordering or duplicating messages.

For a given schedule (α, β) and starting state $\mathbf{X}$ we define δ, the asynchronous version of σ, as follows:

$$\delta^0(\mathbf{X})_{ij} \triangleq \mathbf{X}_{ij}$$

$$\delta^t(\mathbf{X})_{ij} \triangleq \begin{cases} \bigoplus_k \mathbf{A}_{ik}(\delta^{\beta(t,i,k)}(\mathbf{X})_{kj}) \oplus \mathbf{I}_{ij} & \text{if } i \in \alpha(t) \\ \delta^{t-1}(\mathbf{X})_{ij} & \text{otherwise} \end{cases}$$

We can recover σ by setting $\alpha(t) = \{1, ..., n\}$ and $\beta(t, i, j) = t - 1$, i.e. at each time step every node activates and all messages only take a single time step to propagate.

3.2 Dynamic networks and convergence

In the definition of δ, network topology and individual policies are hard-coded into the adjacency matrix $\mathbf{A}$. However real networks are in constant states of flux and we need to be able to capture this in our model.

When the network undergoes a change at time t, we view the continuing computation as a new instance of the problem, using a new adjacency matrix and taking $\delta^t(\mathbf{X})$, the current network state, to be the new starting state $\mathbf{X}'$. If an edge weight is changed or an edge is added or removed then the relevant entry in $\mathbf{A}$ is adjusted. If a node is added or removed we simply add or a remove the corresponding row and column into the adjacency matrix, $\mathbf{A}$, and the starting state, $\mathbf{X}'$.

After a change to the network the new starting state $\mathbf{X}'$ may contain stale routes that no longer exist in the new network. It is therefore vital that the theorems in this paper guarantee that δ converges from *all* states, rather than just the states consistent with the current topology.

There exist routing algebras that may converge from some states but not others. For instance the classic shortest paths algebra converges from the initial state, but if started from arbitrary states it suffers from the count-to-infinity problem.

Definition 7. *δ converges from state* $\mathbf{X}$ if there exists a stable state $\mathbf{X}^*$ such that for all schedules there exists a convergence time t such that for all k then $\delta^{t+k}(\mathbf{X}) = \mathbf{X}^*$.

Definition 8. *δ converges* if it converges from all possible starting states.

Definition 9. *δ converges absolutely* if it always converges to the same stable state from all possible starting states.

Absolute convergence therefore guarantees that δ will always converge to a single, predictable final state, no matter what state started in. Obviously in practice convergence will only occur during suitably long periods of network stability.

3.3 A convergence theorem

Using the well-established model of asynchronous computation described in Section 3.1 has two main advantages:

(1) there already exist published sufficient conditions for the absolute convergence of δ.
(2) these conditions only require properties of the synchronous iteration σ, rather than δ itself.

One of the most widely used sufficient conditions is Üresin & Dubois' [27]. Their conditions are then used as a basis for many others (see [7] for an overview), including the result we use from Gurney [15].

In our opinion is one of the more intuitive as it requires the construction of a suitable notion of distance between states such that for any state $\mathbf{X}$ the distance between $\mathbf{X}$ and $\sigma(\mathbf{X})$ is strictly greater than the distance between $\sigma(\mathbf{X})$ and $\sigma^2(\mathbf{X})$. At a high level this makes sense, as if every application of σ moves the state a smaller and smaller distance, then eventually the computation must reach a stable state where applying σ no longer moves the state at all.

Only having to reason about the synchronous behaviour greatly simplifies our proof. In fact, this paper goes on to prove absolute convergence of δ without ever mentioning α and β again. In contrast the papers discussed in Section 1.1 directly reason about the asynchronous nature of the protocols and, perhaps consequently, all make the simplifying assumptions of in-order, reliable delivery of messages.

Definition 10. An *ultrametric* over a set S is a distance function $d : S \times S \to \mathbb{N}$ that satisfies the following conditions:

M1 : $d(x, y) = 0 \Leftrightarrow x = y$
M2 : $d(x, y) = d(y, x)$
M3 : $d(x, z) \leq \max(d(x, y), d(y, z))$

An ultrametric is just a stronger version of a standard metric, where the triangle inequality:

$$d(x, z) \leq d(x, y) + d(y, z)$$

has been strengthened to:

$$d(x, z) \leq \max(d(x, y), d(y, z))$$

Definition 11. A function $f : S \to S$ is *strictly contracting* over an ultrametric d if for all $x \neq y$ then:

$$d(x, y) > d(f(x), f(y))$$

Definition 12. A function $f : S \to S$ is *strictly contracting on orbits* over an ultrametric d if for all $x \neq f(x)$ then:

$$d(x, f(x)) > d(f(x), f^2(x))$$

Lemma 2. If f is strictly contracting on orbits over d then there exists a fixed point for f.

Proof. While $f^k(x) \neq f^{k+1}(x)$, we can use the strictly contracting on orbits property to construct the following chain:

$$d(x, f(x)) > d(f(x), f^2(x)) > d(f^2(x), f^3(x)) > ...$$

This is a decreasing chain in $\mathbb{N}$ and so must have finite length. Therefore there must eventually be a time t such that $f^t(x) = f^{t+1}(x)$ and so $f^t(x)$ is the required fixed point. □

Definition 13. A function $f : S \to S$ is *strictly contracting on its fixed point* over an ultrametric d if for all $x \neq x^*$ then:

$$d(x^*, x) > d(x^*, f(x))$$

Definition 14. An ultrametric is *bounded* if there exists a d_{max} such that for all x and y then $d(x, y) \leq d_{max}$.

Lemma 3. If d is an ultrametric over routes S then

$$D(\mathbf{X}, \mathbf{Y}) = \max_{ij} d(\mathbf{X}_{ij}, \mathbf{Y}_{ij})$$

is an ultrametric over routing states $\mathbb{M}_n(S)$.

Proof. See [15] and our Agda [6]. □

Theorem 1. Given a routing algebra $(S, \oplus, F, \overline{0}, \overline{\infty})$ then δ converges absolutely if there exists an ultrametric d over routes, such that:

(1) D is bounded
(2) σ is strictly contracting on orbits over D
(3) σ is contracting on its fixed point over D

where $D(\mathbf{X}, \mathbf{Y}) = \max_{ij} d(\mathbf{X}_{ij}, \mathbf{Y}_{ij})$.

Proof. See Theorem 5 in [15] and our Agda [6]. □

4 DISTANCE VECTOR PROTOCOLS

In this section we prove that distance-vector protocols with finite, strictly increasing routing algebras converge absolutely. This models RIP-like protocols that support conditional policies. To do so we construct an ultrametric, d, over routes and then prove that σ is strictly contracting with respect to D, hence fulfilling the pre-conditions of Theorem 1.

4.1 An ultrametric over routes

We begin by assuming that S is finite.

Height of routes. As S is finite, all downwards closed subsets under the relation $\leq$ must also be finite. The height of a route can therefore be defined as follows:

$$h(x) \triangleq |\{y \in S \mid x \leq y\}|$$

The trivial route, $\overline{0}$, is the most desirable route and so has the maximum height which we name H. The invalid route, $\overline{\infty}$, is the least desirable route and therefore is the route with the minimal height of 1. Therefore for all routes x we have:

$$1 = h(\overline{\infty}) \leq h(x) \leq h(\overline{0}) = H$$

Distance between routes. Using h, we next define the distance function $d : S \times S \to \mathbb{N}$ as follows:

$$d(x, y) \triangleq \begin{cases} 0 & \text{if } x = y \\ \max(h(x), h(y)) & \text{otherwise} \end{cases}$$

Consequently the distance between two routes is proportional to how desirable the best of the two routes is. Intuitively this is a reasonable measure of distance. Better routes are more likely to be adopted by other nodes and get propagated throughout the network. Hence if two routing states disagree on the best route from i to j, then, from a convergence perspective, the seriousness of the disagreement is directly proportional to the desirability of the two conflicting routes.

Lemma 4. d is an ultrametric.

Proof. We show that all three ultrametric axioms hold:

M1 : immediate, $1 \leq h(x)$ and so $d(x, y) = 0$ iff $x = y$.

M2 : immediate, max is commutative and so $d(x, y) = d(y, x)$.

M3 : we need to prove that for all x, y and z:

$$d(x, z) \leq \max(d(x, y), d(y, z))$$

Case 1: if $x = z$ then $d(x, z) = 0$ and so the inequality holds.
Case 2: if $x = y$ then

$$\begin{aligned} d(x, z) &\leq \max(d(x, y), d(x, z)) \\ &= \max(d(x, y), d(y, z)) \end{aligned}$$

Case 3: if $y = z$ then similar to Case 2.
Case 4: if $x \neq y$, $y \neq z$ and $x \neq z$ then

$$\begin{aligned} d(x, z) &= \max(h(x), h(z)) \\ &\leq \max(h(x), h(y), h(z)) \\ &= \max(\max(h(x), h(y)), \max(h(y), h(z))) \\ &= \max(d(x, y), d(y, z)) \end{aligned}$$

Hence d obeys all three ultrametric axioms. □

Distance between routing states. As proved by Lemma 3 in Section 3.3, we can define the ultrametric D over routing states as follows:

$$D(\mathbf{X}, \mathbf{Y}) \triangleq \max_{i,j} d(\mathbf{X}_{ij}, \mathbf{Y}_{ij})$$

Again D measures the distance between $\mathbf{X}$ and $\mathbf{Y}$. If all the elements of $\mathbf{X}$ and $\mathbf{Y}$ are equal then they occupy the same point in the space, otherwise the distance between them grows in proportion to the most desirable route they disagree on.

Lemma 5. If $(S, \oplus, F, \overline{0}, \overline{\infty})$ is strictly increasing then σ is strictly contracting over D.

Proof. We need to show that for all $\mathbf{X} \neq \mathbf{Y}$:

$$D(\mathbf{X}, \mathbf{Y}) > D(\sigma(\mathbf{X}), \sigma(\mathbf{Y})) = \max_{i,j} d(\sigma(\mathbf{X})_{ij}, \sigma(\mathbf{Y})_{ij})$$

It therefore suffices to prove that for all nodes i and j

$$D(\mathbf{X}, \mathbf{Y}) > d(\sigma(\mathbf{X})_{ij}, \sigma(\mathbf{Y})_{ij}) \quad (6)$$

Case 1: $\sigma(\mathbf{X})_{ij} = \sigma(\mathbf{Y})_{ij}$

Then we immediately have (6) as:

$$\begin{aligned} D(\mathbf{X}, \mathbf{Y}) &> 0 && \text{(by M1 \& } \mathbf{X} \neq \mathbf{Y}) \\ &= d(\sigma(\mathbf{X})_{ij}, \sigma(\mathbf{Y})_{ij}) && \text{(by M1 \& case 1)} \end{aligned}$$

Case 2: $\sigma(\mathbf{X})_{ij} \neq \sigma(\mathbf{Y})_{ij}$

Without loss of generality we assume that $\sigma(\mathbf{X})_{ij}$ is a more desirable route than $\sigma(\mathbf{Y})_{ij}$:

$$\sigma(\mathbf{X})_{ij} < \sigma(\mathbf{Y})_{ij} \tag{7}$$

A node's route to itself is always the trivial route (Lemma 1) and so if $i = j$ then $\sigma(\mathbf{X})_{ij} = \overline{0} = \sigma(\mathbf{Y})_{ij}$ which contradicts Case 2's assumption. Hence going forwards we assume that $i \neq j$. Consequently by Eq 5 (in Section 2.2) we have that:

$$\sigma(\mathbf{X})_{ij} = \bigoplus_k \mathbf{A}_{ik}(\mathbf{X}_{kj})$$

and so, as $\oplus$ is selective, there exists a node k such that:

$$\sigma(\mathbf{X})_{ij} = \mathbf{A}_{ik}(\mathbf{X}_{kj}) \tag{8}$$

If $\mathbf{X}_{kj} = \overline{\infty}$ then we have that $\sigma(\mathbf{X})_{ij} = \overline{\infty}$ which contradicts (7) and therefore we have that:

$$\mathbf{X}_{kj} \neq \overline{\infty} \tag{9}$$

If $\mathbf{X}_{kj} = \mathbf{Y}_{kj}$ then we have that:

$$\begin{aligned}
\sigma(\mathbf{X})_{ij} &= \mathbf{A}_{ik}(\mathbf{X}_{kj}) && \text{(by 8)}\\
&= \mathbf{A}_{ik}(\mathbf{Y}_{kj}) && \text{(by assumption)}\\
&\geq \sigma(\mathbf{Y})_{ij} && \text{(by defn. of } \sigma)
\end{aligned}$$

which contradicts (7) and so:

$$\mathbf{X}_{kj} \neq \mathbf{Y}_{kj} \tag{10}$$

We can now prove (6) as follows:

$$\begin{aligned}
D(\mathbf{X},\mathbf{Y}) &\geq d(\mathbf{X}_{kj},\mathbf{Y}_{kj}) && \text{(by defn. of } D)\\
&= \max(h(\mathbf{X}_{kj}), h(\mathbf{Y}_{kj})) && \text{(by 10)}\\
&\geq h(\mathbf{X}_{kj}) && \text{(by defn. of max)}\\
&> h(\mathbf{A}_{ik}(\mathbf{X}_{kj})) && \text{(by str. incr. \& 9)}\\
&= h(\sigma(\mathbf{X})_{ij}) && \text{(by 8)}\\
&= \max(h(\sigma(\mathbf{X})_{ij}), h(\sigma(\mathbf{Y})_{ij})) && \text{(by 7)}\\
&= d(\sigma(\mathbf{X})_{ij}, \sigma(\mathbf{Y})_{ij})) && \text{(by Case 2)}
\end{aligned}$$

which is the required result. □

Theorem 2. If $(S, \oplus, F, \overline{0}, \overline{\infty})$ is a strictly increasing and finite routing algebra then δ converges absolutely.

Proof. We can apply Theorem 1 directly as:

- D is clearly bounded above by H.
- Lemma 5 applied to $\mathbf{X}$ and $\sigma(\mathbf{X})$ gives us that σ is strictly contracting on orbits.
- Lemma 2 therefore provides the existence of the fixed point $\mathbf{X}^*$, and Lemma 5 applied to $\mathbf{X}^*$ and $\mathbf{X}$ gives us that σ is contracting on this fixed point.

Hence δ converges absolutely over $(S, \oplus, F, \overline{0}, \overline{\infty})$. □

4.2 Practical implications

Theorem 2 guarantees that distance-vector routing protocols with finite, strictly increasing algebras are guaranteed to converge from any starting state to the same final state, even in the most unfavourable of asynchronous conditions. In particular, it implies that convergence would still be guaranteed if complex conditional policies were added to distance-vector protocols like RIP.

5 PATH VECTOR PROTOCOLS

In practice the finiteness condition proves restrictive as many routing algebras of interest have an infinite set of routes. For example even the shortest-path algebra, which solves perhaps the most basic routing problem of all, uses the carrier set $\mathbb{N}$. However remember that this theorem guarantees convergence from *any* state. Shortest-path distance-vector protocols on the other hand may experience *count-to-infinity* problems when the starting state contains routes generated along paths that do not exist in the current topology.

How do real routing protocols get around this? RIP artificially limits the maximum hop count to 16, hence ensuring that the set S is finite. However the most common approach is that of path-vector protocols which track the paths along which the routes are generated. Routes are then removed if they contain a looping path. We show that for strictly increasing algebras this is sufficient to guarantee that eventually the protocol will always reach a finite subset of *consistent routes* from which it can then converge. We will now define some additional theory for talking about path-vector protocols.

5.1 Paths

A *path* is a defined to be a sequence of contiguous edges and a path is *simple* if it never visits a node more than once. For our purposes, we also consider an additional simple path $\bot$, which will represent the path of the invalid route. Note that in order to deal with arbitrary starting states, we do not restrict the set of paths to those in the current network topology.

Let $\mathcal{P}$ be the set of simple paths. The $weight : \mathcal{P} \to S$ of a path p is defined as follows:

$$weight(p) = \begin{cases} \overline{\infty} & \text{if } p = \bot \\ \overline{0} & \text{if } p = [\,] \\ \mathbf{A}_{ij}(weight(q)) & \text{if } p = (i,j) :: q \end{cases}$$

There is no standardised way that path-vector protocols keep track of paths, and so we need an abstraction to hide the implementation details. Therefore we assume that there exists a projection function $path : S \to \mathcal{P}$ which takes a route and returns the path that the route was generated along.

Definition 15. A *path algebra* is a routing algebra equipped with a *path* function that obeys the following properties:

P1 : $x = \overline{\infty} \Leftrightarrow path(x) = \bot$
P2 : $x = \overline{0} \Rightarrow path(x) = [\,]$

$$\textbf{P3} : path(\mathbf{A}_{ij}(r)) = \begin{cases} \bot & \text{if } i \in path(r) \\ \bot & \text{if } j \neq src(path(r)) \\ (i,j) :: path(r) & \text{otherwise} \end{cases}$$

Note that $P3$ implies that any increasing path algebra is automatically strictly increasing as it cannot be the case that $\mathbf{A}_{ij}(r) = r$ as the paths are not equal. Going forwards we therefore only require increasingness for path algebras.

We feel that the *path* function abstraction has some significant advantages over the approach of Sobrinho [25], who models the paths separately from the routing algebra. Our abstraction can model operations such as AS prepending by simply adjusting the *path* function to strip out padded ASs, whereas to do so in [25] would require adjusting the proof itself. Similarly it is more difficult to describe route filtering in [25] as by default the algebra does not have access to the path information.

Definition 16. A route r is *consistent* if it is equal to the weight of the path along which it was generated, i.e.

$$weight(path(r)) = r$$

Every consistent route is uniquely specified by a simple path. Therefore the set of consistent routes, S^c, can be defined as:

$$S^c \triangleq \{weight(p) \mid p \in \mathcal{P}\}$$

If p and q are consistent then, by the selectivity of $\oplus$, so is $p \oplus q$. Likewise if route x is consistent then P1 – P3 guarantee that $\mathbf{A}_{ij}(x)$ is consistent as well. Therefore if every route in $\mathbf{X}$ is consistent so is every route in $\sigma(\mathbf{X})$. Consequently the only way inconsistent routes can be introduced into the routing state is by a change to the network topology, for example if a node along the route's path changes its policy or a link along the path is removed.

As S^c is finite and is closed under σ then Theorem 2 immediately implies that δ converges from any consistent state for increasing path algebras. However if S is infinite then are an infinite number of inconsistent states and so Theorem 2 does not guarantee absolute convergence for increasing path algebras. In particular the proof falls down as the height function h is ill-defined when S is infinite.

5.2 An ultrametric over routes

When constructing an ultrametric over routes for path algebras the key insight is that, as S^c is finite, we can reuse the ultrametric from Section 4.1 for consistent routes.

The remaining problem is to find a quantity that decreases when applying σ to an inconsistent routing state. As S^c is closed under σ, any inconsistent route in $\sigma(\mathbf{X})$ must be an extension of some inconsistent route in $\mathbf{X}$. Therefore after each application of σ the length of the shortest inconsistent path must strictly increase. As all the paths are simple their length cannot be greater than n and so, in the absence of further topology changes, all routes must become consistent.

$$\begin{tikzcd}
\text{Distance vector} & & \text{Path vector} & & \\
h(x) \arrow[d] & = & h^c(x) \arrow[d] & & h^i(x) \arrow[d] \\
d(x,y) \arrow[ddd] & = & d^c(x,y) \arrow[dr] & & d^i(x,y) \arrow[dl] \\
 & & & d(x,y) \arrow[dd] & \\
 & & & & \\
D(\mathbf{X},\mathbf{Y}) & & & D(\mathbf{X},\mathbf{Y}) &
\end{tikzcd}$$

Figure 2: The structure of the ultrametrics in the paper.

The two separate ultrametrics can then be combined to form a unified ultrametric over both inconsistent and consistent routes (see Figure 2).

Inconsistent height of routes. With this in mind, the inconsistent height of a route can therefore be defined as follows:

$$h^i(x) \triangleq \begin{cases} 1 & \text{if } x \in S^c \\ (n+1) - length(path(x)) & \text{otherwise} \end{cases}$$

where n is the number of nodes in the network.

All consistent routes have the minimum height 1, and the maximum height is $n+1$ as we only consider simple paths. To explicitly highlight the parallels with Section 4.1, we will call this maximum height:

$$H^i \triangleq n + 1$$

Inconsistent distance between routes. Define a new distance function d^i over S as:

$$d^i(x,y) \triangleq \max(h^i(x), h^i(y))$$

Note that this is not a true ultrametric as it does not obey assumption M1 (i.e. $x = y \Leftrightarrow d(x,y) = 0$). However as we will see, this requirement is unnecessary as it will never be used to compare two equal elements.

Distance between routes. As S^c is finite then H and d from Section 4.1 are defined over consistent routes. Let us rename them H^c and d^c respectively.

A distance function over all routes can now be defined as:

$$d(x,y) = \begin{cases} 0 & \text{if } x = y \\ d^c(x,y) & \text{if } x \neq y \text{ and } \{x,y\} \subseteq S^c \\ H^c + d^i(x,y) & \text{if } x \neq y \text{ and } \{x,y\} \nsubseteq S^c \end{cases}$$

If the two routes are both consistent then we use d^c to compute the distance between them. However if either one is inconsistent then we use d^i instead. It is necessary to add H^c to $d^i(x, y)$ in order to get the contraction properties required later. In particular it ensures that the distance between inconsistent routes is always greater than the distance between consistent routes. This is necessary as the distance must decrease at the point in time at which the last inconsistent route is flushed from the routing state.

Clearly M1 and M2 are satisfied by d but M3 is not immediately obvious. In fact the triangle inequality does hold but the proof does not provide any insights. Interested readers can find the details in our Agda formalisation [6].

Distance function over states. As before we can now define the distance function over routing states as:

$$D(\mathbf{X}, \mathbf{Y}) = \max_{ij} d(\mathbf{X}_{ij}, \mathbf{Y}_{ij})$$

Unlike in Section 4.1, we cannot prove that σ is a strict contraction with respect to D, thereby concurrently proving that σ is both strictly contracting on orbits and contracting on its fixed point. Hence it is necessary to prove both of the contraction properties required by Theorem 1 separately.

Lemma 6. If $\sigma(\mathbf{X})_{ij}$ is inconsistent then there exists a node k such that $\mathbf{X}_{kj}$ is inconsistent and $\mathbf{X}_{kj} \neq \sigma(\mathbf{X})_{kj}$.

Proof. As $\sigma(\mathbf{X})_{ij}$ is inconsistent it must be an extension of some inconsistent route in $\mathbf{X}$ and therefore there exists a node l such that $\sigma(\mathbf{X})_{ij} = \mathbf{A}_{il}(\mathbf{X}_{lj})$ and $\mathbf{X}_{lj}$ is inconsistent.

If $\mathbf{X}_{lj} \neq \sigma(\mathbf{X})_{lj}$ then l is our required node. Otherwise if $\mathbf{X}_{lj} = \sigma(\mathbf{X})_{lj}$ then $\sigma(\mathbf{X})_{lj}$ is inconsistent. We can therefore repeat the entire argument with $\sigma(\mathbf{X})_{lj}$. However the length of the path of $\sigma(\mathbf{X})_{lj}$ is strictly less than that of the path of $\sigma(\mathbf{X})_{ij}$ and so each time the length of the path strictly decreases and therefore the argument must eventually terminate. □

Lemma 7. If the path algebra is increasing then σ is strictly contracting on orbits over D.

Proof. We need to prove that for all $\mathbf{X} \neq \sigma(\mathbf{X})$:

$$D(\mathbf{X}, \sigma(\mathbf{X})) > D(\sigma(\mathbf{X}), \sigma^2(\mathbf{X}))$$

Assume $\mathbf{X} \neq \sigma(\mathbf{X})$. Let i and j be the nodes such that entries $\mathbf{X}_{ij}$ and $\sigma(\mathbf{X})_{ij}$ have the maximum d distance between them, then:

$$D(\mathbf{X}, \sigma(\mathbf{X})) = d(\mathbf{X}_{ij}, \sigma(\mathbf{X})_{ij})$$

We then need to show that for all nodes p and q:

$$d(\mathbf{X}_{ij}, \sigma(\mathbf{X})_{ij}) > d(\sigma(\mathbf{X})_{pq}, \sigma^2(\mathbf{X})_{pq}) \tag{11}$$

<u>Case 1</u>: $\mathbf{X}_{ij} = \sigma(\mathbf{X})_{ij}$

Then by M1 we have that

$$D(\mathbf{X}, \sigma(\mathbf{X})) = d(\mathbf{X}_{ij}, \sigma(\mathbf{X})_{ij}) = 0$$

and therefore by M1 again that $\mathbf{X} = \sigma(\mathbf{X})$ which contradicts the assumption $\mathbf{X} \neq \sigma(\mathbf{X})$.

<u>Case 2</u>: $\mathbf{X}_{ij}$ and $\sigma(\mathbf{X})_{ij}$ are consistent.

<u>Case 2.1</u>: $\sigma(\mathbf{X})_{pq} = \sigma^2(\mathbf{X})_{pq}$

Then (11) holds immediately as:

$$\begin{aligned} d(X_{ij}, \sigma(\mathbf{X})_{ij}) &= d^c(\mathbf{X}_{ij}, \sigma(\mathbf{X})_{ij}) \\ &> 0 \\ &= d(\sigma(\mathbf{X})_{pq}, \sigma^2(\mathbf{X})_{pq}) \end{aligned}$$

<u>Case 2.2</u>: $\sigma(\mathbf{X})_{pq}$ and $\sigma^2(\mathbf{X})_{pq}$ are consistent.

We can prove that (11) by using Case 2 from Lemma 5 to show that d^c is strictly contracting, and hence:

$$\begin{aligned} d(X_{ij}, \sigma(\mathbf{X})_{ij}) &= d^c(\mathbf{X}_{ij}, \sigma(\mathbf{X})_{ij}) \\ &> d^c(\sigma(\mathbf{X})_{pq}, \sigma^2(\mathbf{X})_{pq}) \\ &= d(\sigma(\mathbf{X})_{pq}, \sigma^2(\mathbf{X})_{pq}) \end{aligned}$$

<u>Case 2.3</u>: $\sigma(\mathbf{X})_{pq}$ or $\sigma^2(\mathbf{X})_{pq}$ is inconsistent.

In either case by Lemma 6 there exists a node k such that $\mathbf{X}_{kq}$ is inconsistent and $\mathbf{X}_{kq} \neq \sigma(\mathbf{X})_{kq}$. This contradicts the assumptions that $D(\mathbf{X}, \sigma(\mathbf{X})) = d(\mathbf{X}_{ij}, \sigma(\mathbf{X})_{ij})$ where $\mathbf{X}_{ij}$ and $\sigma(\mathbf{X})_{ij}$ are both consistent, as our definition of d ensures that the distance between consistent routes is always strictly less than the distance between inconsistent routes.

<u>Case 3</u>: $\mathbf{X}_{ij}$ or $\sigma(\mathbf{X})_{ij}$ is inconsistent.

<u>Case 3.1</u>: $\sigma(\mathbf{X})_{pq} = \sigma^2(\mathbf{X})_{pq}$

Then (11) holds immediately as:

$$\begin{aligned} d(X_{ij}, \sigma(\mathbf{X})_{ij}) &= H^c + d^i(\mathbf{X}_{ij}, \sigma(\mathbf{X})_{ij}) \\ &> 0 \\ &= d(\sigma(\mathbf{X})_{pq}, \sigma^2(\mathbf{X})_{pq}) \end{aligned}$$

<u>Case 3.2</u>: $\sigma(\mathbf{X})_{pq}$ and $\sigma^2(\mathbf{X})_{pq}$ are consistent.

Then (11) holds immediately as:

$$\begin{aligned} d(X_{ij}, \sigma(\mathbf{X})_{ij}) &= H^c + d^i(\mathbf{X}_{ij}, \sigma(\mathbf{X})_{ij}) \\ &> d^c(\sigma(\mathbf{X})_{pq}, \sigma^2(\mathbf{X})_{pq}) \\ &= d(\sigma(\mathbf{X})_{pq}, \sigma^2(\mathbf{X})_{pq}) \end{aligned}$$

and d^c is bounded above by H^c

Case 3.3: $\sigma(\mathbf{X})_{pq}$ or $\sigma^2(\mathbf{X})_{pq}$ are inconsistent.

In this final case we have that

$$\begin{aligned} D(\mathbf{X}, \sigma(\mathbf{X})) &= d(\mathbf{X}_{ij}, \sigma(\mathbf{X})_{ij}) \\ &= d^i(\mathbf{X}_{ij}, \sigma(\mathbf{X})_{ij}) \\ &= \max(h^i(\mathbf{X}_{ij}), h^i(\sigma(\mathbf{X})_{ij})) \end{aligned}$$

and therefore $D(\mathbf{X}, \sigma(\mathbf{X}))$ is the height of the shortest inconsistent path in $\mathbf{X}$ and $\sigma(\mathbf{X})$. Likewise we have that:

$$\begin{aligned} d(\sigma(\mathbf{X})_{pq}, \sigma^2(\mathbf{X})_{pq}) &= d^i(\sigma(\mathbf{X})_{pq}, \sigma^2(\mathbf{X})_{pq}) \\ &= \max(h^i(\sigma(\mathbf{X})_{pq}), h^i(\sigma^2(\mathbf{X})_{pq})) \end{aligned}$$

and so $d(\sigma(\mathbf{X})_{pq}, \sigma^2(\mathbf{X})_{pq})$ is the height of some inconsistent route in $\sigma(\mathbf{X})$ or $\sigma^2(\mathbf{X})$. However that route must be an extension of some inconsistent route in $\mathbf{X}$ or $\sigma(\mathbf{X})$ and therefore its path is strictly longer than the shortest inconsistent path in $\mathbf{X}$ and $\sigma(\mathbf{X})$. Hence (11) holds by the definition of d^i . □

Hence by the above result and Lemma 2 we have that there exists a stable state $\mathbf{X}^*$.

Lemma 8. *If the path algebra is increasing then σ is contracting on its fixed point $\mathbf{X}^*$.*

Proof. The proof's structure is the same as that of Lemma 7. The only major difference is in Case 2.3, where the contradiction takes a slightly different form which we now briefly outline below. Interested readers can find the remaining details in our Agda formalisation [6].

Case 2.3: $\mathbf{X}^*$ cannot be inconsistent as otherwise applying σ would increase the length of the shortest inconsistent path. Hence $\mathbf{X}^*$ is consistent. In Case 2.3 we have that:

$$D(\mathbf{X}^*, \mathbf{X}) = d^c(\mathbf{X}^*_{ij}, \mathbf{X}_{ij}) \tag{12}$$

and so, as $\mathbf{X}^*$ is consistent, $\mathbf{X}$ must be consistent as well. Hence $\sigma(\mathbf{X})$ is consistent, which contradicts the assumption of Case 2.3 that either $\mathbf{X}^*_{pq}$ or $\sigma(\mathbf{X})_{pq}$ is inconsistent. □

Theorem 3. *Given an increasing path algebra $(S, \oplus, F, \overline{0}, \overline{\infty})$ then δ converges absolutely.*

Proof. We can apply Theorem 1 directly as:

- D is clearly bounded above by $H^c + (n + 1)$.
- σ is strictly contracting on orbits by Lemma 7.
- σ is contracting on its fixed point by Lemma 8.

Hence δ converges absolutely over $(S, \oplus, F, \overline{0}, \overline{\infty})$. □

6 FORMALISATION IN AGDA

Every mathematical result in this paper, down to the most trivial lemma, has been formalised in the dependently-typed theorem-proving language Agda. This includes any theorems the paper uses from Üresin & Dubois [27] and Gurney [15], a discussion of which can be found in [30].

6.1 The advantages of formalisation

The proofs in our formalisation have been checked by a computer, and therefore we are far more confident in their correctness than usual. Even where we omit details or use standard informal mathematical reasoning to improve readability (e.g. Lemma 8), the proofs are backed by mathematical arguments which are guaranteed to be fully rigorous.

Furthermore the act of formalisation itself was invaluable in creating and shaping these proofs. For example the unstructured way in which we initially laid out the pen-and-paper proof of Lemma 7 led us to overlook Case 2.3. Only when formalising the result, did we notice that this case remained unproven. This in turn led us to presenting the proof in the much cleaner structure displayed in this paper.

The library of proofs is freely available [6] and is highly modular. Users only have to prove that the algebraic implementation of their protocol obeys the strictly increasing conditions in order to guarantee absolute convergence. In the Section 7 we give an outline of how to define such an algebra in Agda. We hope that the library's extensible nature means that it will be of use to the research community.

6.2 A small example

Another advantage of Agda is that it allows the use of unicode symbols, and consequently the formal proofs can follow their pen-and-paper versions much more closely.

Consider the distance function over routes $d : S \to S \to \mathbb{N}$ which we defined at the start of Section 4.1 as:

$$d(x, y) \triangleq \begin{cases} 0 & \text{if } x = y \\ \max(h(x), h(y)) & \text{otherwise} \end{cases}$$

In Agda the definition of d is almost identical:

```
d : S → S → ℕ
d x y with x ≟ y
... | yes x=y = 0
... | no  x≢y = h x ⊔ h y
```

To prove that d obeys, for example, M2, we can use exactly the same language constructs to write the proof:

```
d−sym : ∀ x y → d x y ≡ d y x
d−sym x y with x ≟ y | y ≟ x
... | yes x=y | yes y=x = refl
... | yes x=y | no  y≢x = contradiction (sym x=y) y≢x
... | no  x≢y | yes y≡x = contradiction (sym y≡x) x≢y
... | no  x≢y | no  y≢x = ⊔-comm (h x) (h y)
```

The proof first checks whether x equals y and y equals x. If both the equalities hold then we need to prove that $0 = 0$ which is trivially true (proved using refl, a proof that equality is reflexive). If one equality holds but the other does not, then we have a contradiction (using sym, a proof that equality is symmetric). If neither equality holds then we need to prove that h x ⊔ h y is equal to h y ⊔ h x. This is proved via ⊔−comm, a proof of the commutativity of the max operator which is available in the Agda standard library.

7 A SAFE-BY-DESIGN ALGEBRA

We now present an example of how one could use our Agda library to develop a safe-by-design routing algebra. Our example is a path-vector algebra that contains many of the features of BGP such as local preferences, community values [4] and conditional policies. These policies can perform operations such as path-filtering and modifying local preferences and communities. The conditions themselves are implemented using a simple language of predicates that includes the ability to inspect communities. The algebra is a superset of the Stratified Shortest Paths algebra [10].

It differs from the algebra underlying today's BGP in two crucial ways. Firstly, BGP allows ASs to hide their local preferences and set them to arbitrary values upon importing routes from other ASs. This violates the assumption that the algebra is increasing. Secondly, the implementation of the MED attribute violates the assumption that ⊕ is associative [13].

Our algebra avoids these two issues by a) ignoring MED and b) having policies that only allow local preference to increase. We present our algebra only to give a practical example of how our theory can be used, and to show that "most" of the features of BGP are inherently safe. We are not presenting it as a practical solution to these two problems in real-world BGP. We discuss the open question of whether hidden information is compatible with increasing algebras in Section 8.2.

As our algebra is increasing, our Agda implementation of Theorem 3 guarantees that any protocol based on it is safe-by-design (i.e. it is impossible to write policies that interfere with convergence).

7.1 The path algebra

A route is defined as following:

```
data Route : Set where
  invalid : Route
  valid : LPref → CommunitySet → SimplePath n → Route
```

i.e. there exists an invalid route, and all other routes have a local preference, a set of communities and a simple path.

The trivial route, $\overline{\infty}$, is the route "valid 0 ∅ []" and the invalid route, $\overline{0}$, is "invalid". The *path* function from routes to simple paths required by a path algebra can be defined as:

```
path : Route → SimplePath n
path invalid         = ⊥
path (valid _ _ p) = p
```

The Agda definition of ⊕, the operation for choosing between routes, is a little too long for this paper. Interested readers may consult the Agda code [6]. However $x \oplus y$ follows the following decision procedure:

(1) If x or y is invalid return the other.
(2) else if the level of one of x or y is strictly less than the other return that route.
(3) else if the length of the path of one of x or y is strictly less than the other return that route.
(4) else break ties by a lexicographic comparison of paths.

We will now construct the set of edge weights F. We start by defining a simple yet expressive language for conditions that can be used by our policy language to make decisions.

```
data Condition : Set where
  and      : Condition    → Condition → Condition
  or       : Condition    → Condition → Condition
  not      : Condition    → Condition
  inPath   : Node         → Condition
  inComm   : Community → Condition
  lprefEq  : LPref        → Condition
```

We next define the policy language as follows:

```
data Policy : Set₁ where
  reject     : Policy
  incrPrefBy : ℕ → Policy
  addComm    : Community → Policy
  delComm    : Community → Policy
  compose    : Policy → Policy → Policy
  condition  : Condition → Policy → Policy
```

The semantics of each type of policy are defined by the function that applies policies to routes:

```
apply : Policy → Route → Route
apply _                invalid        = invalid
apply reject           _              = invalid
apply (incrPrefBy x)  (valid l cs p) = valid (l + x) cs p
apply (addComm c)     (valid l cs p) = valid l (add c cs) p
apply (delComm c)     (valid l cs p) = valid l (remove c cs) p
apply (compose p q)   r              = apply q (apply p r)
apply (condition c p) r              =
                   if (evaluate c r) then (apply p r) else r
```

Note that since we cannot decrease a route's local preference, it is not possible to define a non-increasing policy.

We define the set of edge weight functions as:

$$F = \{ f_{i,j,pol} \mid \forall i, j, pol \}$$

where nodes i and j are the source and destination of the edge and pol is a Policy. The function $f_{i,j,pol}$ is defined as follows:

```
f : (Node × Node × Policy) → Route → Route
f _              invalid          = invalid
f (i , j , pol) (valid x cs p) with (i , j) ⇿? p | i ∉? p
... | no ¬ij⇿p | _              = invalid
... | _              | no  i∈p = invalid
... | yes ij⇿p  | yes i∉p =
                     apply pol (valid x cs ((i , j) ∷ p | ij⇿p | i∉p))
```

where $(i\ ,\ j)$ ⇿? p tests if the edge (i, j) is a valid extension of path p (i.e. if $j = src(p)$), and $i \notin? \ p$ tests whether or not i already exists in p (i.e. if the resulting path would loop).

The above definitions ensure that the algebra

$$(\text{Route}, \oplus, \text{F}, \text{valid } 0 \ \varnothing \ [], \text{invalid})$$

satisfies all the requirements of an increasing path algebra as defined in Sections 2.1 & 5.1. For formal proofs of these properties see the Agda formalisation [6]. Our implementation of Theorem 3 in Agda guarantees that a protocol based on this algebra converges from any state to a unique solution even in the presence of message loss, reordering and duplication.

There are many other features of BGP that are safely increasing but, for space reasons, are not included in this model. For example AS path prepending is possible to add with minor tweaks to the *path* function and the policy language.

8 OPEN QUESTIONS

8.1 Convergence time

The rate of convergence of σ for increasing path algebras is still poorly understood. Consider a network of n nodes. With distributive policies we know that in the worst case $O(n)$ synchronous iterations of σ are required to reach a fixed point [1, 9]. In our upcoming work [5] we prove an upper bound for increasing path algebras of $O(n^2)$, and show that this bound is tight by exhibiting an algebra and a family of networks that require $O(n^2)$ synchronous iterations to converge.

However it appears that not all non-distributive, increasing algebras require $O(n^2)$ iterations (e.g. the shortest-widest-paths algebra). We suspect that a careful analysis of policy language features might be able to tease apart distinct classes with respect to worst-case convergence time.

8.2 Hidden information

In the algebra described in Section 7, unlike in (external) BGP, the local preference attribute is not deleted when exporting a route. This raises a more general issue for routing protocols that allow information to be hidden. It is an open question as to whether it is possible to have increasing algebras with hidden information without requiring global coordination.

Ensuring increasing policies in today's BGP may require communicating lost information with some other mechanism such as community values. Of course only the relative ranking of local preference values assigned within an AS matter. For example one AS might use a local preference of 100 for its most preferred routes, while another could use 2000. In this context can we ensure increasing policies using only bilateral agreements between neighbouring networks or does it truly require global coordination? If the latter, then a political rather than a technical solution is required.

8.3 Verification of data-center policies

BGP is widely used today to implement (private) connectivity within and between data-centers [20]. In such an environment the network architects have total control of the global topology and therefore hidden information is not an issue. Yet even here we have witnessed the use of conditional policies, combined with filtering and the manipulation of local preference on routes. Perhaps tools such as Propane [2] could be extended to either ensure that all policies are strictly increasing, or at the very least provide warnings when they are not?

8.4 Formalising bisimulation

As discussed in Section 1.3 there are trade-offs between generality and implementation details. However some more operations which don't immediately fit into our formalism can be addressed using *bisimulation*.

An algebra A is *bisimilar* to an algebra B if the behaviour of A's σ is indistinguishable from the behaviour of B's σ. Therefore if A converges absolutely, then so does B. We can use this to prove the convergence of some algebras that don't technically fulfil the earlier definitions of path algebras, by exhibiting a bisimilar path algebra.

For example, a path algebra assumes the existence of the *path* function, that provides the router-level path a route was generated along. However in BGP, routes only store the AS level path, and perhaps the router-level path of the current AS. At first glance this would appear to preclude applying Theorem 3 to algebras with hierarchical paths.

However imagine a version of a BGP-like protocol that did not discard the router level path upon exiting an AS, but also did not let policies make decisions based on this extra information. This algebra has a *path* function and therefore satisfies Theorem 3, and is clearly bisimilar to the original algebra that did discard router-level paths. Hence the original algebra converges as well. This, and similar arguments, have not yet been formalised.

REFERENCES

[1] John S Baras and George Theodorakopoulos. 2010. Path problems in networks. *Synthesis Lectures on Communication Networks* 3, 1 (2010), 1–77.

[2] Ryan Beckett, Ratul Mahajan, Todd Millstein, Jitendra Padhye, and David Walker. [n. d.]. Don'T Mind the Gap: Bridging Network-wide Objectives and Device-level Configurations. In *Proceedings of the 2016 ACM SIGCOMM Conference*.

[3] Dimitri P Bertsekas, Robert G Gallager, and Pierre Humblet. 1992. *Data networks*. Vol. 2. Prentice-Hall International New Jersey.

[4] R. Chandra, P. Traina, and T. Li. 1996. *BGP Communities Attribute*. RFC 1997. http://www.ietf.org/rfc/rfc1997.txt

[5] Matthew L. Daggitt and Timothy G. Griffin. 2018. Rate of convergence of increasing path-vector routing protocols. (2018). Under submission.

[6] Matthew L. Daggitt, Ran Zmigrod, and Timothy G. Griffin. 2018. Agda Routing Library. (2018). https://github.com/MatthewDaggitt/agda-routing/tree/sigcomm2018

[7] Andreas Frommer and Daniel B Szyld. 2000. On asynchronous iterations. *Journal of computational and applied mathematics* 123, 1 (2000), 201–216.

[8] Lixin Gao and Jennifer Rexford. 2001. Stable Internet routing without global coordination. *IEEE/ACM Transactions on Networking (TON)* 9, 6 (2001), 681–692.

[9] Michel Gondran and Michel Minoux. 2008. *Graphs, dioids and semirings: new models and algorithms*. Vol. 41. Springer Science & Business Media.

[10] Timothy G. Griffin. 2012. Exploring the stratified shortest-paths problem. *Networking Science* 1, 1 (01 Mar 2012), 2–14. https://doi.org/10.1007/s13119-011-0003-6

[11] Timothy G Griffin and G Huston. 2005. *BGP wedgies*. RFC 4264. http://www.ietf.org/rfc/rfc4264.txt

[12] Timothy G Griffin, F Bruce Shepherd, and Gordon Wilfong. 2002. The stable paths problem and interdomain routing. *IEEE/ACM Transactions on Networking (ToN)* 10, 2 (2002), 232–243.

[13] Timothy G Griffin and Gordon Wilfong. 2002. Analysis of the MED Oscillation Problem in BGP. In *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*. IEEE, 90–99.

[14] Arjun Guha, Mark Reitblatt, and Nate Foster. 2013. Machine-verified Network Controllers. In *PLDI*.

[15] Alexander J T Gurney. 2017. *Asynchronous iterations in ultrametric spaces*. Technical Report. https://arxiv.org/abs/1701.07434

[16] Geoff Huston. 1999. Interconnection, Peering and Settlements: Part I. *Internet Protocol Journal (Cisco)* 2, 1 (June 1999).

[17] Geoff Huston. 1999. Interconnection, Peering and Settlements: Part II. *Internet Protocol Journal (Cisco)* 2, 2 (June 1999).

[18] Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch, and Simon Winwood. 2009. seL4: Formal Verification of an OS Kernel. In *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles (SOSP)*.

[19] Ramana Kumar, Magnus O. Myreen, Michael Norrish, and Scott Owens. 2014. CakeML: A Verified Implementation of ML. In *POPL*.

[20] Parantap Lahiri, George Chen, Petr Lapukhov, Edet Nkposong, Dave Maltz, Robert Toomey, and Lihua Yuan. 2012. *Building Scalable Data Centers: BGP is the Better IGP!* presentation.

[21] Xavier Leroy. 2009. Formal Verification of a Realistic Compiler. *Commun. ACM* 52, 7 (2009), 107–115.

[22] D. McPherson, V. Gill, D. Walton, and A. Retana. 2002. *Border Gateway Protocol (BGP) Persistent Route Oscillation Condition*. RFC 3345. http://www.ietf.org/rfc/rfc3345.txt

[23] Ulf Norell. 2009. Dependently Typed Programming in Agda. In *Proceedings of the 4th International Workshop on Types in Language Design and Implementation*.

[24] Y. Rekhter, T. Li, and S. Hares. 2006. *A Border Gateway Protocol 4 (BGP-4)*. RFC 4271. http://www.ietf.org/rfc/rfc4271.txt

[25] João Luís Sobrinho. 2005. An algebraic theory of dynamic network routing. *IEEE/ACM Transactions on Networking (TON)* 13, 5 (2005), 1160–1173.

[26] Joao Luıs Sobrinho and Timothy G Griffin. 2010. Routing in equilibrium. *Mathematical Theory of Networks and System* (2010).

[27] Aydin Üresin and Michel Dubois. 1990. Parallel asynchronous algorithms for discrete data. *Journal of the ACM (JACM)* 37, 3 (1990), 588–606.

[28] Kannan Varadhan, Ramesh Govindan, and Deborah Estrin. 2000. Persistent route oscillations in inter-domain routing. *Computer networks* 32, 1 (2000), 1–16.

[29] Arseniy Zaostrovnykh, Solal Pirelli, Luis Pedrosa, Katerina Argyraki, and George Candea. 2017. A Formally Verified NAT. In *SIGCOMM*.

[30] Ran Zmigrod, Matthew L. Daggitt, and Timothy G. Griffin. 2018. An Agda Formalization of Üresin & Dubois' Asynchronous Fixed-Point Theory. *9th International Conference on Interactive Theorem Proving (ITP)* (July 2018).