# Detecting Unsafe BGP Policies in a Flexible World

Debbie Perouli*, Timothy G. Griffin†, Olaf Maennel‡, Sonia Fahmy*,
Cristel Pelsser§, Alexander Gurney¶ and Iain Phillips‡
* Department of Computer Science, Purdue University, USA. Email: {depe,fahmy}@cs.purdue.edu
† Computer Laboratory, University of Cambridge, UK. Email: tgg22@cam.ac.uk
‡ Department of Computer Science, Loughborough University, UK. Email: {O.M.Maennel,I.W.Phillips}@lboro.ac.uk
§ Innovation Institute, Internet Initiative Japan, Japan. Email: cristel@iij.ad.jp
¶ Department of Computer and Information Science, University of Pennsylvania, USA. Email: agurney@seas.upenn.edu

***Abstract*—Internet Service Providers (ISPs) need to balance multiple opposing objectives. On one hand, they strive to offer innovative services to obtain competitive advantages; on the other, they have to interconnect with potentially competing ISPs to achieve reachability, and coordinate with them for certain services. The complexity of balancing these objectives is reflected in the diversity of policies of the Border Gateway Protocol (BGP), the standard inter-domain routing protocol.**

**Unforeseen interactions among the BGP policies of different ISPs can cause routing anomalies. In this work, we propose a methodology to allow ISPs to check their BGP policy configurations for guaranteed convergence to a single stable state. This requires that a set of ISPs share their configurations with each other, or with a trusted third party. Compared to previous approaches to BGP safety, we (1) allow ISPs to use a richer set of policies, (2) do not modify the BGP protocol itself, and (3) detect not only instability, but also multiple stable states. Our methodology is based on the extension of current theoretical frameworks to relax their constraints and use incomplete data. We believe that this provides a rigorous foundation for the design and implementation of safety checking tools.**

## I. Introduction

The Border Gateway Protocol (BGP) [1] is critical to today's Internet. It enables routers from different Internet Service Providers (ISPs) to exchange reachability information. The highly expressive policy language of BGP allows ISPs to implement the "layer 3 details" of their complex commercial relationships with neighboring ISPs. This expressive power comes with some risks. Routing policies may appear rational to each individual ISP, yet policy *interactions* among ISPs can cause routing anomalies such as protocol oscillation [2], [3] and unintended stable states [4].

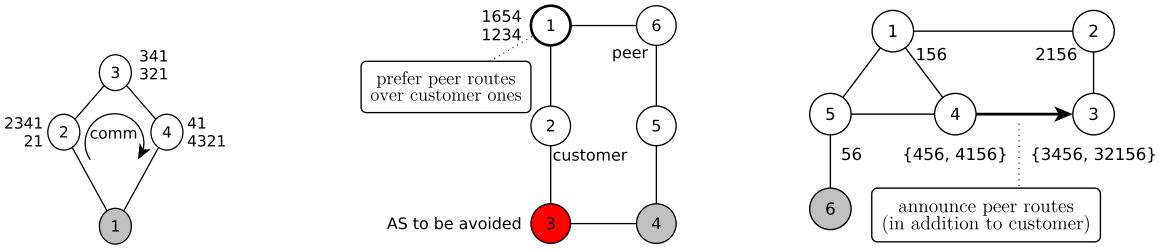
Advances in the theory of policy-based routing (for example, [5]–[7]) have yielded sufficient conditions that ensure BGP *safety* — that is, guarantees of convergence to a unique stable state. Yet *flexibility* is typically seen as the more important concern in the operation of real-world networks. We can break flexibility down into the distinct concepts of *autonomy* and *expressiveness*. Autonomy here refers to the ability of each ISP to configure its own network with little or no global coordination, while expressiveness refers to the ability to innovate in the policy domain in response to evolving and unforeseen customer needs. An open research question in inter-domain routing is whether or not it is possible to design protocols that guarantee safety while maintaining levels of flexibility acceptable for ISP operations.

978-1-4673-2447-2/12/$31.00 

In the meantime, we must live with legacy BGP and the fact that safety is sacrificed (knowingly or unknowingly) in favor of full autonomy and unconstrained policy expressiveness. In this context, one possible approach to safety is to check router configurations for possible safety violations. There are two difficulties with this approach, both of which may be overcome to a limited extent. First, it has been shown that determining safety from configuration files is in the worst case an intractable problem [8]. However, we feel it may be practically feasible to do this type of analysis when the number of ISPs involved is small (dozens, not thousands). Second, ISPs consider their routing polices to be private. For this, we imagine that ISPs may be willing to share their configuration files with a trusted third party in return for information that facilitates debugging. This approach may provide ISPs with more control over the balance between safety and flexibility. We will present modifications to existing theoretical models of BGP that are specifically aimed at supporting this approach.

Previous approaches to configuration analysis [9] have been based on the Stable Paths Problem (SPP) framework [6]. However, the SPP contains a "strictness condition" which essentially requires every step of the BGP decision procedure be captured — all the way down to the "last gasp" tie-break on router identifiers. The problem with this approach is that it requires too much information regarding the internal details of an ISP (like IGP distances, and the use of the troublesome MED attribute) and it may depend on many tricky vendor-specific details of BGP policy implementation.

In this work, we develop a variant of the SPP model, called *Extended SPP* (Section III), that does not require the strictness condition and so allows for an approximation to the full SPP model. Our Extended SPP is related to the recently introduced *partial SPP* [12] that allows for path rankings that are entirely unknown. However, we believe that the Extended SPP model is better suited for the analysis of existing BGP configuration files. Section IV develops data structures derived from the Extended SPP framework while Section V presents the theoretical results needed to apply these structures to the safety problem. Finally, Section VI describes a methodology of configuration analysis that is based on the theoretical models and structures of the preceding sections.




(a) The backup relationship requires coordination [5]. The backup paths (21, 321, 4321) need to be propagated with a "de-preference" community on *each* hop in order to ensure safety.

(b) Avoiding sending traffic through a particular AS may require preferring a peer or provider route over a customer route. AS 1 prefers the 1654 peer route over the 1234 customer route, because it does not wish its traffic to traverse AS 3.

(c) Propagating peer routes to peers may become a routine policy in a flatter (less hierarchical) Internet [10], [11]. Node 3 depends less on its provider node 2, since it has the additional route 3456.

Fig. 1. Three commonly occurring scenarios that violate the Gao-Rexford guidelines.

## II. Related Approaches

The Gao-Rexford guidelines [5] prevent problematic specifications. The guidelines define the roles (provider/customer, or peer) that an ISP is allowed to have with respect to each of its neighbors. Depending on the role, the ISP then has to follow the corresponding rules that specify (a) the preference it may assign to routes, and (b) the routes it is allowed to announce to other neighbors. *Safety* and *autonomy* are guaranteed. However, *expressiveness* is severely limited to those policies that comply with the guidelines. Recent studies [10], [11] indicate that the Internet has many more peering relationship types, and thus more policies, than the fundamental hypothesis of the Gao-Rexford model assumes.

Fig. 1 illustrates some simple scenarios that violate the Gao-Rexford guidelines, yet we believe these to be quite common in today's Internet. Fig. 1a depicts ISP 2 providing a "backup route" service to ISP 1, which wants the rest of the nodes to access it through its primary provider, node 4, unless the 4-1 link is down. In that case only, nodes should access 1 through the backup link 2-1. Fig. 1b presents a situation where ISP 1 selects the peer path 1654 over the customer path 1234 to avoid sending traffic through a specific Autonomous System (AS). If the rest of the ASes follow traditional policies, the specification is still safe. In Fig. 1c, ISP 4 announces the route it learns from its peer ISP 5 to another of its peers, ISP 3. Although announcing peer routes to peers is prohibited by the Gao-Rexford guidelines, this policy may be desirable to ISPs that wish to strengthen their peer relationships and depend less on their providers [13], [14]. ISP 3 now has two choices, paths 3456 and 32156, instead of only its provider path through 2.

It is important to note that this relaxation to the guidelines does not always create an unsafe system. However, when multiple ISPs choose such "non-standard" policies, the interactions can lead to unexpected results. For example, RFC 4264 [4] describes how scenarios like that depicted in Fig. 1a can lead to multiple stable states, some of which violate the intent of the policy writers. It is exactly this kind of unintended policy interaction that motivates our work.

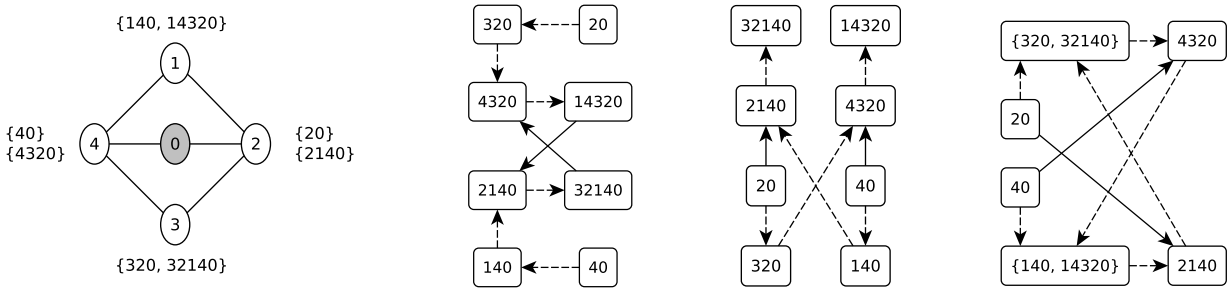
In addition to guidelines that are targeted at routing safety, there has been a lot of work on the theoretical foundations of policy-based routing. The SPP model has been used in many related studies. Necessary conditions have been proposed for more specific safety related problems [15]–[18]. A dynamic model of SPP that verifies the compliance of a network with the Gao-Rexford guidelines [5] is proposed by Epstein *et. al.* [19]. Arye *et al.* [20] follow a different formal approach. They develop a tool to generate specifications with no stable solution or multiple solutions. The tool can be used to test the validity of sufficient conditions. Cittadini *et. al.* [9] propose a heuristic and tool to statically check BGP configurations for guaranteed routing convergence.

From the empirical side, Wetherall *et. al.* [21] study BGP misconfigurations. They observe on the order of 600 misconfigurations in originating prefixes and 100 in exporting prefixes in RouteViews data. The RCC tool [22] includes a set of syntactic, and a limited number of semantic, policy checks, for detecting misconfigurations. Nemecis [23] is another tool for analyzing policies, expressed in RPSL.

A different approach that allows more policy flexibility to ISPs has been taken by [24]–[26]. Their distributed algorithms detect and resolve policy interactions which lead to BGP oscillations. The methodology is based on the paths that routers have seen in the past, in their "route history." The inherent limitation of this approach is that it is meant to detect routing oscillations, not any kind of problematic policy interaction. Keeping track of the route history does not guarantee the detection of problematic interactions that lead to multiple stable solutions. The implementation of these algorithms also requires modifications to BGP itself.

## III. Extended SPP

The *Stable Paths Problem* (SPP) [6], [27] is a graph-theoretic model that provides sufficient conditions for safe policy-based routing. To verify SPP conditions in operational networks, SPP requires detailed information about the configuration of a network—the specifics of the BGP best path selection process [1]. In this section, we relax the strictness property




(a) Strict SPP Specification $\mathcal{S}$ (b) $\mathcal{DD}(\mathcal{S})$ (c) $\mathcal{PD}(\mathcal{S})$ (d) $\mathcal{MD}(\mathcal{S})$

Fig. 2. This example illustrates: (i) a multipath case that can be directly modeled in *strict* SPP, (ii) a specification whose $\mathcal{PD}$ does not have a cycle, while $\mathcal{DD}$ (or $\mathcal{MD}$) does, and (iii) the smallest possible cycle of only *inflation* arcs in $\mathcal{MD}$ that can appear in *strict* SPP.

of SPP and set the ground for a theory that significantly reduces the complexity of the safety checking algorithm.

We model a routing problem as an SPP by defining three parameters: $(\mathcal{G}, \mathcal{P}, \Lambda)$. A simple, undirected, connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ captures the topology. Each node of $\mathcal{V}$ is labeled $u_i$, $0 \leq i < |\mathcal{V}|$ where $u_0$ is the *origin* of all routing paths (*i.e.,* the destination of data paths).

Set $\mathcal{P}$ consists of all the simple, permitted, routing paths of $\mathcal{G}$. A routing path of length $k \geq 1$ is a sequence of nodes $p_k = u_k, u_{k-1}, \ldots, u_1, u_0$ such that $u_i \in \mathcal{V}$, $(u_i, u_{i-1}) \in \mathcal{E}$ and $1 \leq i \leq k$. If $p_k \in \mathcal{P}$, it is implied that $p_k$ belongs to the set of permitted paths of node $u_k$, that is $p_k \in \mathcal{P}^{u_k}$.

Every node $u_i \neq u_0$ has a ranking function $\lambda^{u_i}$ which specifies which paths are permitted at $u_i$. The function $\lambda^{u_i}$ also sorts $u_i$'s permitted paths to $u_0$ from most to least preferred. The most preferred path of $u_i$ is the path for which $\lambda^{u_i}$ is maximum. $\Lambda$ is the set that contains all functions $\lambda^{u_i}, \forall i \neq 0$.

The definition of SPP also specifies the *strictness* property: two paths are allowed to be equally preferred by a node $u_i$, if they have the same next hop. Two paths are equally preferred, trivially, when the paths themselves are equal (they consist of the same sequence of nodes).

**Example III.1.** The specification shown in Fig. 2a is an example of an SPP. Paths 40 and 20 are more preferred than paths 4320 and 2140, respectively. Paths 140 and 14320 are equally preferred by node 1, and since they have the same next-hop (node 4), the preference of node 1 satisfies the strictness property. Similarly, node 3 satisfies strictness.

**Definition III.1 (*Strict* SPP).** We refer to the SPP model that was first introduced in [27] as *strict* SPP. Under the strictness property of this model, two paths can be equivalent iff they have the same next hop or they are the same path.

**Definition III.2 (*Extended* SPP).** *Extended* SPP is an SPP model without the strictness property. A specification $\mathcal{S}$ is still described by the triple $(\mathcal{G}, \mathcal{P}, \Lambda)$. However, there is no restriction on the preference functions of the set $\Lambda$ regarding which paths can be considered equivalent.

In other words, under *extended* SPP, two distinct paths $p_1$ and $p_2$ which do not share the next-hop node can still be equally preferred by a node $u_i$: $\lambda^{u_i}(p_1) = \lambda^{u_i}(p_2)$. Clearly, a specification that can be expressed as a *strict* SPP can also be expressed as an *extended* SPP.

**Definition III.3 (Equivalence Class).** A set of paths $E$ of a node $u_i$ in which for any two paths $p_x, p_y \in E$ $\lambda^{u_i}(p_x) = \lambda^{u_i}(p_y)$ is an *equivalence class*, i.e., all paths in the same equivalence class are equally preferred by node $u_i$.

**Definition III.4 (Refinement).** Let $\mathcal{S}$ be a specification modeled as a *strict* or *extended* SPP. Specification $\mathcal{R}$ is a *refinement* of $\mathcal{S}$, written as $\mathcal{R}(\mathcal{S})$, if each node $u_i$ in $\mathcal{R}$ has a strict order of path rankings which follows the rules:

- if $\lambda^{u_i}(p_1) > \lambda^{u_i}(p_2)$ in $\mathcal{S}$, then the same is true in $\mathcal{R}$
- if $\lambda^{u_i}(p_1) = \lambda^{u_i}(p_2)$ in $\mathcal{S}$, then either $\lambda^{u_i}(p_1) > \lambda^{u_i}(p_2)$ or $\lambda^{u_i}(p_1) < \lambda^{u_i}(p_2)$ in $\mathcal{R}$

**Lemma III.1.** *Every path of a refinement $\mathcal{R}(\mathcal{S})$ belongs to an equivalence class which contains no other path but itself.*

Note that Lemma III.1 does not necessarily hold for a strict SPP specification $\mathcal{S}$. Two distinct paths are allowed to belong to the same equivalence class as long as they have the same next hop.

**Lemma III.2.** *Any refinement $\mathcal{R}(\mathcal{S})$ can be modeled as a* strict *SPP.*

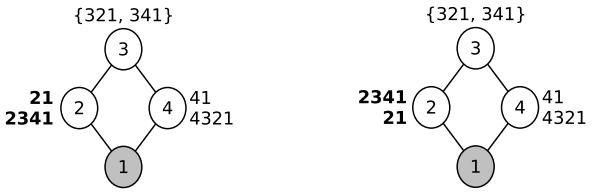
**Example III.2.** Consider again the specification shown in Fig. 2a. Paths 140 and 14320 belong to the same equivalence class of node 1, while paths 320 and 32140 belong to the same equivalence class of node 3. Ranking the paths of the equivalence classes results in the following four refinements:

1) $\mathcal{R}_1$: $\lambda^1(140) > \lambda^1(14320)$ and $\lambda^3(320) > \lambda^3(32140)$, that is path 140 is preferred over 14320 and path 320 is preferred over 32140
2) $\mathcal{R}_2$: $\lambda^1(140) > \lambda^1(14320)$ and $\lambda^3(320) < \lambda^3(32140)$
3) $\mathcal{R}_3$: $\lambda^1(140) < \lambda^1(14320)$ and $\lambda^3(320) > \lambda^3(32140)$
4) $\mathcal{R}_4$: $\lambda^1(140) < \lambda^1(14320)$ and $\lambda^3(320) < \lambda^3(32140)$

Routers execute the BGP decision process to select a *single* best path. Every step of the decision process splits equivalence classes of paths into smaller classes until a single path remains.

The routing system can be modeled as a *strict* SPP by executing all comparisons of the BGP decision process. However, if we stop at an intermediate step, the routing system needs the *extended* SPP model, since there is no guarantee that paths in the same equivalence class will have the same next-hop.

Our goal is to partly execute the decision process and still be able to decide on the satisfaction of sufficient conditions for safety of the routing system. In other words, we are interested in answering the following question: *given an* extended *SPP specification, are any of its refinements safe?* We also wish to provide the network operators with information on how to configure their networks in order to reach a safe refinement.




(a) Multiple safe refinements

(b) One refinement has multiple stable states.

Fig. 3. The example in Fig. 3a has two refinements both of which are safe, since each refinement has a unique stable state to which it always converges. Only one of the refinements of Fig. 3b is safe; the refinement where $\lambda^3(321) > \lambda^3(341)$ is a system with two stable states, first described in RFC 4264 [4].

**Example III.3.** Fig. 3a shows an *extended* SPP specification where all nodes follow shortest paths, and node 3 has two equal-cost choices. The specification has two refinements, $\mathcal{R}$ and $\mathcal{R}'$, depending on which of the two paths node 3 finally prefers in its (unipath) routing. Each of the refinements has a single stable solution, in which the selected paths are: for $\mathcal{R}$: (21, 321, 41), and for $\mathcal{R}'$: (21, 341, 41).

The specification in Fig. 3b is an example of a potentially problematic specification that our methodology aims to recognize. The refinement $\mathcal{W}$ where $\lambda^3(321) > \lambda^3(341)$ is not safe, as it has two stable states: $\mathcal{T}_1 =$ (2341, 341, 41) and $\mathcal{T}_2 =$ (21, 321, 41). $\mathcal{W}$ is referred to as a "wedgie" in RFC 4264 [4], where it is also explained why convergence to multiple stable states is undesirable.

The specification in Fig. 3a can also be seen as an equal-cost multipath routing (ECMP) [28] example. In this paper, we do not model multipath routing, although the extension to SPP and the data structures presented in Section IV can prove useful in such a study.

## IV. Data Structures

Given a *strict* or *extended* SPP specification $\mathcal{S}$, we can construct a dispute digraph $\mathcal{DD}(\mathcal{S})$ [27], a paths digraph $\mathcal{PD}(\mathcal{S})$ [7], [29], or a multipath digraph $\mathcal{MD}(\mathcal{S})$. We introduce $\mathcal{MD}(\mathcal{S})$ in this paper, but we also use $\mathcal{PD}(\mathcal{S})$ in our analysis. The relations that $\mathcal{PD}(\mathcal{S})$ and $\mathcal{MD}(\mathcal{S})$ have to $\mathcal{DD}(\mathcal{S})$ are important, because of the connection of $\mathcal{DD}(\mathcal{S})$ to the sufficient condition for safety in *strict* SPP.

| Symbol | $\mathcal{DD}$ | $\mathcal{PD}$ | $\mathcal{MD}$ |
|---|---|---|---|
| $\longrightarrow$ | Dispute Transmission | Preference Transmission | Ranking Inflationary |
| $\dashrightarrow$ | | | |

TABLE I
NAME AND SYMBOL FOR THE ARCS OF EACH DIGRAPH.

### A. Dispute Digraph ($\mathcal{DD}$)

The dispute digraph, $\mathcal{DD}(\mathcal{S})$, was introduced in [27] to describe relationships among the policies of a specification $\mathcal{S} = (\mathcal{G}, \mathcal{P}, \Lambda)$ expressed in *strict* SPP. The nodes of $\mathcal{DD}$ are all paths of the specification as shown in Fig. 2b. The digraph has two types of arcs: transmission and dispute. A transmission arc connects any path $p$ to an extension $(i, j)p$, where $i$, $j$ are neighboring nodes in $\mathcal{G}$. The definition of a dispute arc is more subtle and is omitted here, since it is not of critical importance to this paper. However, we do draw the dispute arcs in Fig. 2b for completeness. Griffin *et. al.* [27] prove the following theorem.

**Theorem IV.1.** *For any specification $\mathcal{S}$ that can be modeled as a* strict *SPP: if $\mathcal{DD}(\mathcal{S})$ is acyclic, then $\mathcal{S}$ is safe [27].*

### B. Paths Digraph ($\mathcal{PD}$)

The paths digraph encapsulates the computational dependencies that exist among possible paths [7], [12], [29]. Formally, given an extended SPP $\mathcal{S}$, the paths digraph $\mathcal{PD}(\mathcal{S})$ is a directed graph where nodes represent the permitted paths in $\mathcal{S}$. Its arcs are of two types: transmission, defined as in $\mathcal{DD}(\mathcal{S})$, and preference. There is a preference arc from path $p$ to path $q$ if $p$ is in the immediately preferred equivalence class to $q$ (for paths with the same origin). Fig. 4 shows the paths digraphs for all the refinements of the specification in Fig. 2 that were described in Example III.2.

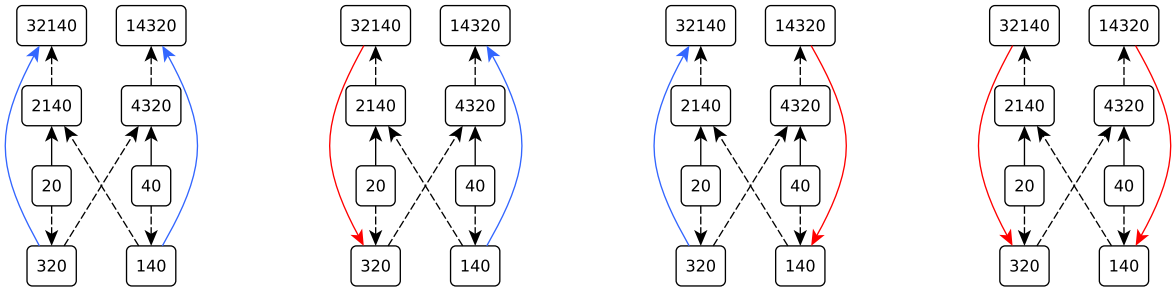
Note that in previous work [7], [29], the definition of a preference arc does not require $p$ to be in the *immediately* preferred class to $q$, it can be in *any* class that is preferred to $q$. This means that additional preference arcs were present. Such additional $(p, q)$ arcs do not affect the presence or absence of a cycle, since $p$ and $q$ are in any case connected by a chain of other preference arcs. Consequently, Theorem IV.2 holds for the $\mathcal{PD}$ definition in this paper.

**Theorem IV.2.** *For any specification $\mathcal{S}$ that can be modeled as a* strict *SPP and has a single refinement $\mathcal{R}(\mathcal{S})$ (*i.e.*, $\mathcal{R}(\mathcal{S}) \equiv \mathcal{S}$), $\mathcal{PD}(\mathcal{S})$ has a cycle iff $\mathcal{DD}(\mathcal{S})$ has a cycle [12], [29].*

### C. Multipath Digraph ($\mathcal{MD}$)

Let $\mathcal{S} = (\mathcal{G}, \mathcal{P}, \Lambda)$ be a specification expressed in *extended* SPP. Each node $u_i$ of $\mathcal{G}$ has equivalence classes of paths. We denote the $j^{th}$ best equivalence class of node $u_i$ as $M_j^{u_i}$, *i.e.*, the paths in $M_j^{u_i}$ are more preferred than the paths in $M_h^{u_i}$, $h > j$. In Fig. 5a, $u_1$ has $k$ classes and node $u_2$ has a single class. $M_1^{u_1}$ contains the most preferred paths of $u_1$.

**Definition IV.1 (Multipath Digraph).** The *Multipath Digraph* of a specification $\mathcal{S}$ uses a node $m_j^{u_i}$ to represent each

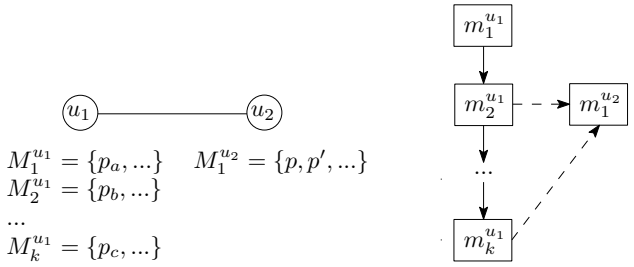



(a) $\mathcal{R}_1$ (b) $\mathcal{R}_2$ (c) $\mathcal{R}_3$ (d) $\mathcal{R}_4$

Fig. 4. Paths digraphs for the four possible refinements of the specification of Fig. 2. The differences among the graphs are the direction of the edges between paths 140, 14320 and 320, 32140. From Lemma IV.3, these are also the multipath digraphs of the refinements.

equivalence class $M_j^{u_i}$, and two types of arcs. A *ranking* arc connects a node $m_j^{u_i}$ to node $m_{j+1}^{u_i}$. An *inflationary* arc connects node $m_j^{u_i}$ to node $m_{j'}^{u_i'}$, if there is at least one path $(u_i', u_i)p$ in class $M_{j'}^{u_i'}$ which is extending a path $p$ of class $M_j^{u_i}$.

We draw ranking arcs with solid lines and inflationary arcs with dotted lines as in Table I. In Fig. 5a, both paths in $M_1^{u_2}$ happen to come from the same node ($u_1$), so this specification satisfies the strictness property. However, $\mathcal{MD}$ does not depend on strictness and can be used to model *extended* SPP.




(a) Part of an SPP specification $\mathcal{S}$ (b) $\mathcal{MD}(\mathcal{S})$

Fig. 5. In this example, path $p \in M_1^{u_2}$ extends a path that belongs in $M_2^{u_1}$ and path $p' \in M_1^{u_2}$ is extending a path that belongs in $M_k^{u_1}$.

**Definition IV.2 (Multi-node).** A node of a multipath digraph is a *multi-node* iff the equivalence class it represents contains more than one path.

**Example IV.1.** The multipath digraph of Fig. 2d has two multi-nodes: a node for the equivalence class of paths {320, 32140} and a node for the class {140, 14320}.

**Lemma IV.3.** *Let $\mathcal{S} = (\mathcal{G}(\mathcal{V}, \mathcal{E}), \mathcal{P}, \Lambda)$. If $\mathcal{R}(\mathcal{S})$ is a refinement of $\mathcal{S}$, then $\mathcal{PD}(\mathcal{R}(\mathcal{S}))$ and $\mathcal{MD}(\mathcal{R}(\mathcal{S}))$ are isomorphic as graphs.*

*Proof:* Based on Lemma III.1, there is no multi-node in $\mathcal{MD}(\mathcal{R}(\mathcal{S}))$: all equivalence classes are singletons, and so each node contains exactly one path. Moreover, $p$ is a node in $\mathcal{PD}(\mathcal{R}(\mathcal{S}))$ if and only if $\{p\}$ is a node in $\mathcal{MD}(\mathcal{R}(\mathcal{S}))$, since both statements are equivalent to the path $p$ being one of the permitted paths in $\mathcal{P}$.

We now show that preference and transmission arcs in $\mathcal{PD}(\mathcal{R}(\mathcal{S}))$ correspond to ranking and inflationary arcs, respectively, in $\mathcal{MD}(\mathcal{R}(\mathcal{S}))$. First, $(p, q)$ is a preference arc in $\mathcal{PD}(\mathcal{R}(\mathcal{S}))$ if and only if $p$ and $q$ are paths with the same origin node, and $p$ is ranked higher than $q$; this condition is equivalent to the existence of a ranking arc. Finally, $(p, (i, j)p)$ is a transmission arc in $\mathcal{PD}(\mathcal{R}(\mathcal{S}))$ whenever both $p$ and $(i, j)p$ are permitted paths, $p$ has source node $j$, and $(i, j)$ is an arc in $\mathcal{E}$. Again, this is equivalent to $(\{p\}, \{(i, j)p\})$ being an inflationary arc in $\mathcal{MD}(\mathcal{R}(\mathcal{S}))$. ■

## V. Sufficient Conditions for Safety of Refinements

In this section, we prove the results on which our methodology is based. Specifically, we show how the graphs of section IV can be used to verify the safety of a refinement $\mathcal{R}(\mathcal{S})$. Note that we make no assumptions regarding the strictness property. Specification $\mathcal{S}$ is modeled as an *extended* SPP, which makes our algorithm less sensitive to the vendor-specific implementation details of the BGP best path selection process. Fig. 6 summarizes our key results.

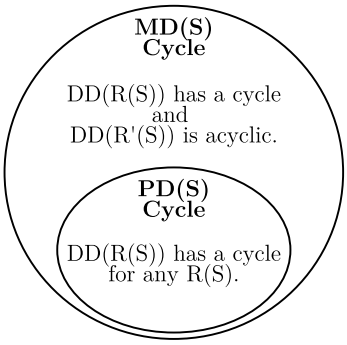



Fig. 6. The relationship between a cycle in $\mathcal{MD}(\mathcal{S})$ or $\mathcal{PD}(\mathcal{S})$ and a cycle in $\mathcal{DD}(\mathcal{R}(\mathcal{S}))$. If $\mathcal{MD}(\mathcal{S})$ is acyclic, $\mathcal{DD}(\mathcal{R}(\mathcal{S}))$ is acyclic too, therefore any $\mathcal{R}(\mathcal{S})$ is safe (Lemma V.4).

**Lemma V.1.** *For a refinement $\mathcal{R}(\mathcal{S})$, if $\mathcal{DD}(\mathcal{R}(\mathcal{S}))$ is acyclic, then $\mathcal{R}(\mathcal{S})$ is safe.*

*Proof:* Follows from Lemma III.2 and Theorem IV.1. ■

**Theorem V.2.** *If $\mathcal{PD}(\mathcal{S})$ is acyclic, there exists a refinement $\mathcal{R}(\mathcal{S})$ that is safe.*

*Proof:* If $\mathcal{PD}(\mathcal{S})$ is acyclic, then there is at least one refinement $\mathcal{R}(\mathcal{S})$ of $\mathcal{S}$ for which $\mathcal{PD}(\mathcal{R}(\mathcal{S}))$ is acyclic [29]. From Theorem IV.2, if $\mathcal{PD}(\mathcal{R}(\mathcal{S}))$ is acyclic then $\mathcal{DD}(\mathcal{R}(\mathcal{S}))$ is acyclic; from Lemma V.1, it follows that $\mathcal{R}(\mathcal{S})$ is safe. ■

**Theorem V.3.** *If $\mathcal{MD}(\mathcal{S})$ is acyclic, $\mathcal{MD}(\mathcal{R}(\mathcal{S}))$ is acyclic for any refinement $\mathcal{R}(\mathcal{S})$.*

*Proof:* We prove this by contradiction. Suppose that there is a cycle in $\mathcal{MD}(\mathcal{R}(\mathcal{S}))$ visiting the nodes $p_1, p_2, \ldots, p_k, p_1$. Based on Lemma III.1, every node in $\mathcal{MD}(\mathcal{R}(\mathcal{S}))$ contains a single path. Let $m_i$ be the node in $\mathcal{MD}(\mathcal{S})$ which contains path $p_i$. If paths $p_i$ and $p_{i+1}$ are connected through an inflationary arc in $\mathcal{MD}(\mathcal{R}(\mathcal{S}))$, then there is an inflationary arc from $m_i$ to $m_{i+1}$ in $\mathcal{MD}(\mathcal{S})$ as well. If paths $p_i$ and $p_{i+1}$ are connected through a ranking arc in $\mathcal{MD}(\mathcal{R}(\mathcal{S}))$, there are two possibilities:

- Paths $p_i$, $p_{i+1}$ belong to the same node in $\mathcal{MD}(\mathcal{S})$ ($m_i \equiv m_{i+1}$). In this case, the paths are equally preferred in $\mathcal{S}$, while $p_i$ is more preferred than $p_{i+1}$ in $\mathcal{R}(\mathcal{S})$.
- Paths $p_i$, $p_{i+1}$ belong to different nodes in $\mathcal{MD}(\mathcal{S})$ ($m_i \neq m_{i+1}$). Since any refinement respects the preferences in $\mathcal{S}$ by definition, there is a ranking arc from $m_i$ to $m_{i+1}$ in $\mathcal{MD}(\mathcal{S})$.

In either case, if there is an arc from $p_i$ to $p_{i+1}$ in $\mathcal{MD}(\mathcal{R}(\mathcal{S}))$, $m_i \equiv m_{i+1}$ or nodes $m_i$ and $m_{i+1}$ are also connected in $\mathcal{MD}(\mathcal{S})$. Therefore, the existence of a cycle in $\mathcal{MD}(\mathcal{R}(\mathcal{S}))$ implies the existence of a cycle in $\mathcal{MD}(\mathcal{S})$. Since this conclusion contradicts the hypothesis, there can be no cycle in $\mathcal{MD}(\mathcal{R}(\mathcal{S}))$ if there is no cycle in $\mathcal{MD}(\mathcal{S})$. ■

**Lemma V.4.** *If $\mathcal{MD}(\mathcal{S})$ is acyclic,* any *refinement $\mathcal{R}(\mathcal{S})$ is safe.*

*Proof:* It follows from Theorem V.3 that $\mathcal{MD}(\mathcal{R}(\mathcal{S}))$ is acyclic for any refinement $\mathcal{R}(\mathcal{S})$. From Lemma IV.3 $\mathcal{PD}(\mathcal{R}(\mathcal{S}))$ is also acyclic, and based on Theorem IV.2 $\mathcal{DD}(\mathcal{R}(\mathcal{S}))$ will have no cycle either. From Lemma V.1, $\mathcal{R}(\mathcal{S})$ is safe. ■

**Theorem V.5.** *If $\mathcal{PD}(\mathcal{S})$ has a cycle, $\mathcal{PD}(\mathcal{R}(\mathcal{S}))$ has a cycle for any refinement $\mathcal{R}(\mathcal{S})$.*

*Proof:* $\mathcal{PD}(\mathcal{S})$ and $\mathcal{PD}(\mathcal{R}(\mathcal{S}))$ have exactly the same nodes. The transmission arcs of the two graphs are also the same. We examine whether two paths that are connected through a preference arc in $\mathcal{PD}(\mathcal{S})$ are also connected in $\mathcal{PD}(\mathcal{R}(\mathcal{S}))$.

Let ($p_i$, $p_j$) be a preference arc in $\mathcal{PD}(\mathcal{S})$. Then, $p_i$ and $p_j$ belong to different equivalence classes $P_i$ and $P_j$ in $\mathcal{S}$, respectively. Each of the paths in $P_i$ has a preference arc towards each path of $P_j$ in $\mathcal{PD}(\mathcal{S})$. For any refinement $\mathcal{R}(\mathcal{S})$, the least preferred path of $P_i$ has a preference arc towards the most preferred path of $P_j$ in $\mathcal{PD}(\mathcal{R}(\mathcal{S}))$. Every refinement will also specify strict preferences among the paths in $P_i$ and $P_j$. Therefore, there will be a sequence of zero or more preference arcs from $p_i$ to the least preferred path of $P_i$ in $\mathcal{PD}(\mathcal{R}(\mathcal{S}))$. Similarly, there will be a sequence of zero or more preference arcs from the most preferred path of $P_j$ to $p_j$.

In any case, if there is a preference arc from $p_i$ to $p_j$ in $\mathcal{PD}(\mathcal{S})$, there is a sequence of one or more preference arcs from $p_i$ to $p_j$ in $\mathcal{PD}(\mathcal{R}(\mathcal{S}))$ for any $\mathcal{R}(\mathcal{S})$. We conclude, then, that a cycle in $\mathcal{PD}(\mathcal{S})$ implies a cycle in $\mathcal{PD}(\mathcal{R}(\mathcal{S}))$. ■

**Lemma V.6.** *Let $\mathcal{R}(\mathcal{S})$ be a refinement of $\mathcal{S}$. If $\mathcal{PD}(\mathcal{S})$ has a cycle, $\mathcal{DD}(\mathcal{R}(\mathcal{S}))$ also has a cycle.*

*Proof:* It follows from Theorem V.5 that $\mathcal{PD}(\mathcal{R}(\mathcal{S}))$ has a cycle, where $\mathcal{R}(\mathcal{S})$ is any refinement of $\mathcal{S}$. From Theorem IV.2, $\mathcal{DD}(\mathcal{R}(\mathcal{S}))$ will also have a cycle. ■

**Theorem V.7.** *If $\mathcal{PD}(\mathcal{S})$ has a cycle, $\mathcal{MD}(\mathcal{S})$ also has a cycle.*

*Proof:* Suppose there exists a sequence of arcs in $\mathcal{PD}(\mathcal{S})$ visiting the nodes $p_1, p_2, \ldots, p_k, p_1$. Without loss of generality, suppose that this sequence of arcs forms a simple cycle, *i.e.*, a circuit. Each path $p_i$ that is represented by a node in $\mathcal{PD}(\mathcal{S})$ is contained in a node $m_i$ in $\mathcal{MD}(\mathcal{S})$. By definition, it is possible that more than two paths are contained in the same node in $\mathcal{MD}(\mathcal{S})$. If there is a transmission arc from $p_i$ to $p_j$ in $\mathcal{PD}(\mathcal{S})$, then there is an inflationary arc from $m_i$ to $m_j$ in $\mathcal{MD}(\mathcal{S})$. Similarly, the existence of a preference arc from $p_i$ to $p_j$ in $\mathcal{PD}(\mathcal{S})$ implies the existence of a ranking arc from $m_i$ to $m_j$ in $\mathcal{MD}(\mathcal{S})$. Therefore, nodes $m_1, m_2, \ldots, m_k, m_1$ form a cycle in $\mathcal{MD}(\mathcal{S})$, which is not necessarily a circuit. ■

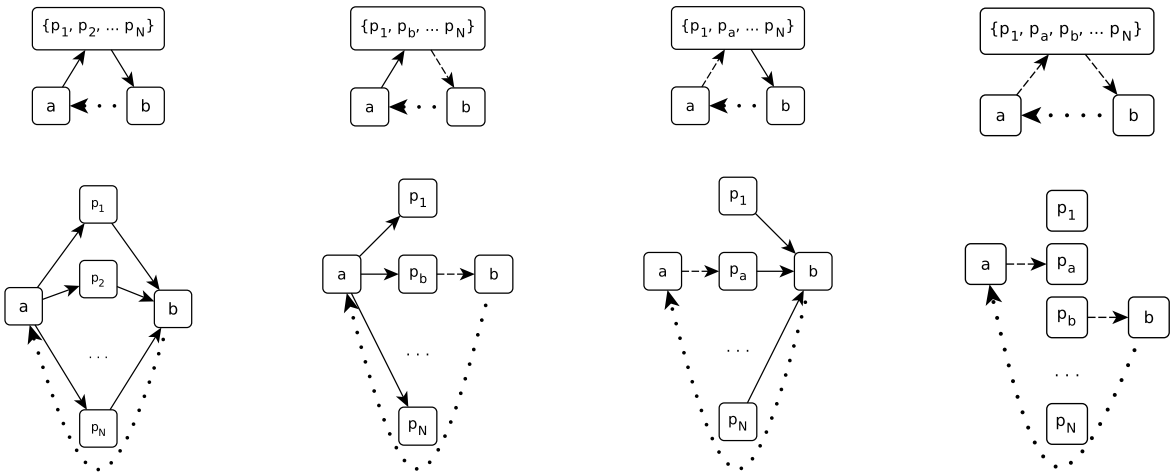
*Note.* The reverse of Theorem V.7 does not hold. Fig. 2 shows an example where $\mathcal{MD}(\mathcal{S})$ has a cycle but $\mathcal{PD}(\mathcal{S})$ is acyclic.

**Theorem V.8.** *If $\mathcal{MD}(\mathcal{S})$ has a cycle, $\mathcal{PD}(\mathcal{S})$ does not have a cycle iff the cycle in $\mathcal{MD}(\mathcal{S})$ contains at least one multi-node* m *for which both of the following hold: (1) Both the incoming and outgoing edges of* m*, which are part of the cycle, are inflationary arcs, and (2) The path that is contained in* m *and is being extended through the outgoing edge is different from the path in* m *that extends the incoming edge.*

*Proof:* Suppose that there is a cycle in $\mathcal{MD}(\mathcal{S})$. If the cycle contains no multi-node, then all nodes and arcs that participate in the $\mathcal{MD}(\mathcal{S})$ cycle exist in $\mathcal{PD}(\mathcal{S})$ as well.

Assume that there is exactly one multi-node *m* in the cycle in $\mathcal{MD}(\mathcal{S})$. Let *a* be the node in the $\mathcal{MD}(\mathcal{S})$ cycle before *m* and let *b* be the node following *m* in the cycle. The part of the cycle from *b* back to *a* contains no multi-nodes, so there must also be a sequence of arcs from *b* to *a* in $\mathcal{PD}(\mathcal{S})$. We examine whether there is a sequence of arcs from *a* to *b* in $\mathcal{PD}(\mathcal{S})$.

There are four cases for the (*a*, *m*) and (*m*, *b*) arcs as shown in Fig. 7. Let the paths contained in the multi-node *m* be $p_1, p_2, \ldots, p_N$. Each of these paths is a separate node in $\mathcal{PD}(\mathcal{S})$.




(a) Both Ranking (b) Ranking and Inflationary (c) Inflationary and Ranking (d) Both Inflationary

Fig. 7. In each of the four subfigures, the top graph is $\mathcal{MD}(\mathcal{S})$ and the bottom graph is $\mathcal{PD}(\mathcal{S})$. The subfigures differ in the type of the arcs in $\mathcal{MD}(\mathcal{S})$ from node $a$ to the multi-node and from the multi-node to node $b$. When both of the arcs are inflationary (as in 7d), there is no cycle in $\mathcal{PD}(\mathcal{S})$ unless $p_a$ and $p_b$ are the same path. The refinements that create a cycle in $\mathcal{PD}(\mathcal{R}(\mathcal{S}))$ are those where $p_a$ is more preferred than $p_b$.

- (*a*, *m*): ranking arc, (*m*, *b*): ranking arc
  In $\mathcal{PD}(\mathcal{S})$, there is a preference arc from *a* to each node $p_i$ and from each node $p_i$ to *b*. There are no preference arcs among the $p_i$ paths, since they all belong to the same equivalence class. As shown in Fig. 7a, there are multiple cycles in $\mathcal{PD}(\mathcal{S})$.
- (*a*, *m*): ranking arc, (*m*, *b*): inflationary arc
  The existence of the inflationary arc means there is a path $p_b$ in *m* which is being extended by a path in *b*. In $\mathcal{PD}(\mathcal{S})$, there is a transmission arc from $p_b$ to *b*. As in the previous case, the ranking arc in $\mathcal{MD}(\mathcal{S})$ implies preference arcs in $\mathcal{PD}(\mathcal{S})$. Fig. 7b illustrates the $\mathcal{PD}(\mathcal{S})$ cycle.
- (*a*, *m*): inflationary arc, (*m*, *b*): ranking arc
  The inflationary arc means there is a path $p_a$ in *m* which extends the path in *a*. In $\mathcal{PD}(\mathcal{S})$, there is a transmission arc from *a* to $p_a$, while the preference arcs are drawn as in the previous cases. Fig. 7c shows the cycle in $\mathcal{PD}(\mathcal{S})$.
- (*a*, *m*): inflationary arc, (*m*, *b*): inflationary arc
  In the general case, there is no cycle in $\mathcal{PD}(\mathcal{S})$ as shown in Fig. 7d. The only case that creates a cycle is $p_a \equiv p_b$.

We conclude that if there is a cycle in $\mathcal{MD}(\mathcal{S})$ that contains exactly one multi-node *m*, there is always a cycle in $\mathcal{PD}(\mathcal{S})$ unless both (*a*, *m*), (*m*, *b*) are inflationary arcs and $p_a \neq p_b$. If the cycle contains many multi-nodes, it suffices to find one multi-node that satisfies the conditions of Theorem V.8 for $\mathcal{PD}(\mathcal{S})$ to be acyclic. ■

**Lemma V.9.** *If $\mathcal{MD}(\mathcal{S})$ has a cycle, then there exists a refinement $R(S)$ for which $\mathcal{DD}(\mathcal{R}(\mathcal{S}))$ has a cycle.*

*Proof:* For all the cases specified in Theorem V.8 in which $\mathcal{PD}(\mathcal{S})$ also has a cycle, it follows from Theorem V.5 that $\mathcal{PD}(\mathcal{R}(\mathcal{S}))$ has a cycle for any $\mathcal{R}(\mathcal{S})$.

We examine the case where $\mathcal{PD}(\mathcal{S})$ is acyclic whereas $\mathcal{MD}(\mathcal{S})$ has a cycle, which is depicted in Fig. 7d with $p_a \neq p_b$. We observe that the refinements whose $\mathcal{PD}(\mathcal{R}(\mathcal{S}))$ has a sequence of preference arcs connecting node $p_a$ to node $p_b$ have a cycle in $\mathcal{PD}(\mathcal{R}(\mathcal{S}))$.

We conclude that regardless of the acyclicity of $\mathcal{PD}(\mathcal{S})$, if $\mathcal{MD}(\mathcal{S})$ has a cycle there always exists a refinement $\mathcal{R}(\mathcal{S})$ such that $\mathcal{PD}(\mathcal{R}(\mathcal{S}))$ has a cycle. From Theorem IV.2, $\mathcal{DD}(\mathcal{R}(\mathcal{S}))$ has a cycle as well. ■

**Lemma V.10.** *If $\mathcal{MD}(\mathcal{S})$ has a cycle and $\mathcal{PD}(\mathcal{S})$ does not have a cycle, there exist refinements $\mathcal{R}(\mathcal{S})$ and $\mathcal{R}'(\mathcal{S})$, $\mathcal{R}(\mathcal{S}) \neq \mathcal{R}'(\mathcal{S})$ , such that $\mathcal{DD}(\mathcal{R}(\mathcal{S}))$ is acyclic (and therefore safe) while $\mathcal{DD}(\mathcal{R}'(\mathcal{S}))$ has a cycle.*

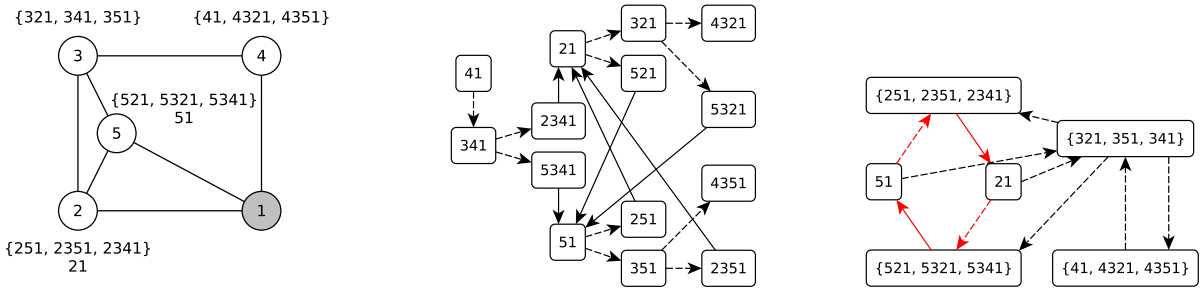
*Proof:* Follows from Theorem V.2 and Lemma V.9. ■

## VI. Configuration Analysis with Incomplete Data

In this section, we describe a configuration analysis algorithm that leverages our theoretical results from previous sections. These results enable us to reduce the complexity of verifying the BGP safety of router configuration files in several cases. They also provide tools to account for partial information. By partial information, we mean that only a few ISPs share their configurations, while the rest of the Internet has unknown policies.

### A. On-Demand BGP Decision Process Exploration

Our methodology only examines steps of the BGP decision process as needed. First, we split the paths of each node into equivalence classes according to the first step of the BGP decision process. This step is the comparison of the BGP local preference attribute [1]. Every equivalence class of a node in the specification will correspond to a local preference value of that node. For router implementations that have a different first step, the procedure is similar.




(a) Double Backup Wedgie (b) $\mathcal{PD}(\mathcal{S})$ (c) $\mathcal{MD}(\mathcal{S})$

Fig. 8. An example from RFC 4264 [4] whose $\mathcal{PD}(\mathcal{S})$ has a cycle. By examining all cycles of $\mathcal{MD}(\mathcal{S})$ we can determine whether $\mathcal{PD}(\mathcal{S})$ has a cycle without creating $\mathcal{PD}(\mathcal{S})$ as proven in Theorem V.8. In this example, the red cycle $51 \rightarrow \{251, 2351, 2341\} \rightarrow 21 \rightarrow \{521, 5321, 5341\} \rightarrow 51$ in Fig. 8c guarantees the existence of the cycle $51 \rightarrow 251 \rightarrow 21 \rightarrow 521 \rightarrow 51$ in $\mathcal{PD}(\mathcal{S})$ since the pairs of arcs adjacent to the multi-nodes are (inflationary, ranking) and this violates the second condition of Theorem V.8.

Then, we create $\mathcal{MD}(\mathcal{S})$. Each equivalence class of paths will be a node in $\mathcal{MD}(\mathcal{S})$. The ranking arcs among classes that belong to the same node $u_i$ of $\mathcal{S}$ will have direction from the classes of higher local preference value of $u_i$ towards classes with lower local preferences. Since all paths of the specification are known, it is also possible to place inflationary arcs between the classes.

If $\mathcal{MD}(\mathcal{S})$ is acyclic, BGP safety is guaranteed for the specification. It is then unnecessary to consider additional steps of the BGP decision process. Note that the local preference attribute is typically used to configure policies which override shortest path routing and are therefore more prone to BGP anomalies [15]. Due to the presented extension to the SPP theory, we may be able to conclude the safety of such policies by examining this attribute alone.

If $\mathcal{MD}(\mathcal{S})$ has a cycle, we need to create $\mathcal{PD}(\mathcal{S})$ and check if it has a cycle. Theorem V.8 provides an alternative to the $\mathcal{PD}(\mathcal{S})$ construction. Checking for a cycle in $\mathcal{PD}(\mathcal{S})$ is equivalent to checking the type of cycles in $\mathcal{MD}(\mathcal{S})$. If there is a cycle in $\mathcal{MD}(\mathcal{S})$ that does not satisfy the conditions specified in Theorem V.8, then that cycle is guaranteed to appear in $\mathcal{PD}(\mathcal{S})$ as well. If $\mathcal{PD}(\mathcal{S})$ has a cycle, then sufficient conditions for safety are not met for any $\mathcal{R}(\mathcal{S})$. Again, it is unnecessary to consider other steps of the BGP decision process. The specification will not meet the sufficient conditions regardless of the decisions made by BGP in subsequent steps.

Theorem V.7 proves that it is not possible for $\mathcal{PD}(\mathcal{S})$ to have a cycle and $\mathcal{MD}(\mathcal{S})$ to be acyclic. For the case when $\mathcal{PD}(\mathcal{S})$ is acyclic but $\mathcal{MD}(\mathcal{S})$ has a cycle, we present two options.

One option is to analyze router configurations to further split equivalence classes of paths according to subsequent steps in the BGP decision process. We then repeat the previous analysis for the new $\mathcal{MD}(\mathcal{S})$. For instance, suppose that the $\mathcal{MD}(\mathcal{S})$ created by separating the paths based on local preference values has a cycle, but $\mathcal{PD}(\mathcal{S})$ does not. Then, we separate each local preference equivalence class into sub-classes based on the AS path length, which is the second step of the BGP decision process. Each sub-class is a node in the new $\mathcal{MD}(\mathcal{S})$, *i.e.*, an equivalence class in the new $\mathcal{MD}(\mathcal{S})$ has paths with the same local preference and AS path length. We repeat the procedure of checking for cycles in the new $\mathcal{MD}(\mathcal{S})$.

Instead of checking whether the current configuration leads to a safe refinement by executing additional steps, the second option is to select one of the safe refinements and configure the network accordingly. Lemma V.10 states that there are refinements which are safe and refinements that are not. In other words, depending on the decisions made in subsequent steps of the decision process, the specification can be guaranteed to be safe or not. From the proof of Theorem V.8, we know which refinements are guaranteed to be safe: the refinements which split the multinode in Fig. 7d so that path $p_b$ is more preferred than $p_a$. Thus, we know what decisions subsequent steps in the BGP decision process need to make in order to have a system that is guaranteed to be safe.

**Example VI.1.** Fig. 8a is a specification inspired by the double backup wedgie example presented in RFC 4264 [4]. The equivalence classes of paths shown can result if one separated paths/prefixes based on local preference values. Customer AS 1 receives its primary service through provider AS 4, while it is signed up for backup service from providers 2 and 5. The two backup providers should use the direct link to their customer (21 or 51 respectively) if and only if they do not have another path available. We assume that AS 2 and 5 implement the specified contract by assigning the lowest possible BGP local preference value to the direct link path. We have also assumed that these "de-preference" policies are the only policies configured through local preference.

Fig. 8b shows that $\mathcal{PD}(\mathcal{S})$ has a cycle. $\mathcal{PD}(\mathcal{S})$ has a cycle even if we configure more policies with local preference, for example the peer-peer policy on the (3, 4) link. As expected, $\mathcal{MD}(\mathcal{S})$ also has a cycle. In particular, the cycle $51 \rightarrow \{251, 2351, 2341\} \rightarrow 21 \rightarrow \{521, 5321, 5341\} \rightarrow 51$ (shown in red in Fig. 8c) guarantees the existence of a cycle in $\mathcal{PD}(\mathcal{S})$ as proven in Theorem V.8.

**Example VI.2.** In the specification of Fig. 2, $\mathcal{MD}(\mathcal{S})$ has the cycle $\{320, 32140\} \rightarrow 4320 \rightarrow \{140, 14320\} \rightarrow 2140 \rightarrow$

$\{320, 32140\}$, while $\mathcal{PD}(\mathcal{S})$ is acyclic. The cycle contains two multinodes whose incoming and outgoing arcs (which are part of the cycle) are inflationary. This is expected due to Theorem V.8. Path $p_a$, shown in Fig. 7d, is path 32140 for multinode {320, 32140} and path 14320 for {140, 14320}. Path $p_b$ is 320 and 140 for each multinode respectively. Due to the same Theorem, we know that the refinement whose $\mathcal{PD}(\mathcal{R}(\mathcal{S}))$ has a cycle is the one where 32140 and 14320 are more preferred than 320 and 140 respectively. This is illustrated in Fig. 4.

**Example VI.3.** Fig. 9a shows a specification where the nodes follow policies that are not recommended by the Gao-Rexford guidelines [5]. In particular, nodes 1 and 4 prefer peer routes equally to customer routes, while nodes 3, 4, and 7 announce to their providers or peers (nodes 2, 3 and 4 respectively) routes they have learned from a peer (4, 7, and 6 respectively). $\mathcal{MD}(\mathcal{S})$, shown in Fig. 9b, is acyclic, therefore $\mathcal{S}$ is safe.

ISPs' incentives to implement non-traditional policies are not well-known or understood, but we know that such policies exist [13]. Here is an example scenario of why they could contribute to ISPs' flexibility. Node 7 may agree to announce peer routes to its provider 4, knowing that 4 will not necessarily prefer customer (routes from 7) over peer (routes from 3 or 5). Node 4 may find it beneficial to have a greater variety of routes available in exchange for ceasing to always prefer customer routes. Peers 3 and 4 could have an agreement to mutually exchange not only routes learned from customers but also from peers for similar reasons. Finally, node 3 may have successfully negotiated a discounted provider service with 2 in return for propagating peer routes to its provider.

### B. Accounting for Unknown Paths

Constructing the complete $\mathcal{MD}(\mathcal{S})$ requires router configuration files from all ISPs. In our model, we have assumed that only a set of neighboring ISPs are sharing such information. Therefore, there will be parts of $\mathcal{MD}(\mathcal{S})$ which are unknown.

Assume that ISPs 1, 3, and 4 of Fig. 9a share their configurations with a third party, while other ISPs do not. We select these nodes because they use non-traditional policies, as described in Example VI.3. For this reason, they are more likely to be concerned about the safety of their policies than the other ISPs of Fig. 9a.

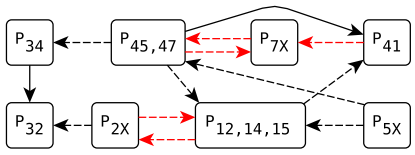



Fig. 10. $\mathcal{MD}(\mathcal{S})$ constructed with partial information. It corresponds to the specification in Fig. 9a when only nodes 1, 3, and 4 share their router configuration files.

Fig. 10 presents the $\mathcal{MD}(\mathcal{S})$ that can be constructed with partial information. The notation $P_{AB}$ denotes the set of paths that node A learns from node B. Similarly, $P_{AB,AC}$ denotes the set of paths node A learns from either B or C. Nodes 2, 5, and 7 are ISPs whose policies are unknown. Their paths are represented as $P_{2X}$ ($P_{5X}$ and $P_{7X}$ respectively), where X stands for any of known or unknown neighbors through which they learn a route. All paths of the $P_{2X}$ set are placed in the same equivalence class, a single node of $\mathcal{MD}(\mathcal{S})$, since there is no information on 2's preferences. There exists an inflationary arc from $P_{2X}$ to $P_{32}$, because the configuration of 3 permits paths announced from 2. Similarly, there is an inflationary arc from $P_{41}$ to $P_{7X}$ since node 4's policies announce to 7 any routes learned from 1 and it is not known whether 7 permits or denies this announcement. The equivalence classes and ranking arcs of the ISPs which share their configuration files are known. For instance, 4 ranks last any routes announced from node 1, while node 1 equally prefers all its available paths. Although the equivalence classes are split based on the neighbor announcing a path in this example, the methodology is independent of the criterion on which paths are ranked.

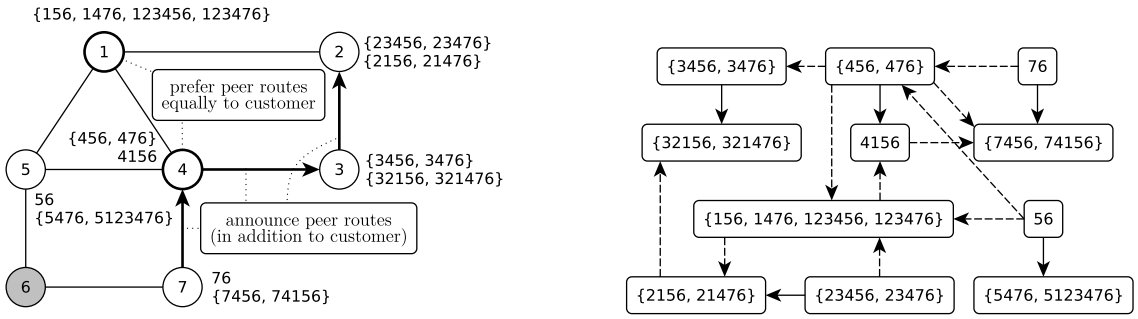
If there is cycle in $\mathcal{MD}(\mathcal{S})$ that *only* involves paths which belong to the ISPs sharing configurations, then these ISPs know their policies may create routing anomalies. Having this information, they can decide what is the economically and operationally best method to resolve such a cycle, if they choose to do so.

The $\mathcal{MD}(\mathcal{S})$ of Fig. 10 has cycles of a different kind: these cycles, *e.g.* $P_{2X} \rightarrow P_{12,14,15} \rightarrow P_{2X}$, include paths that belong to nodes of unknown policies. In such a case, node 1 has three choices. Based on Theorem V.8, it can split its own equivalence class into two so that the cycle is eliminated. More specifically, it will have to prefer the paths it announces to node 2 over the paths it receives from it. A second choice is to convince node 2 to share router configurations and check whether the potential cycle is indeed a cycle based on 2's preferences. Finally, node 1 can risk doing nothing to resolve this potential issue. It has still gained, however, the information that its interaction with node 2 could become a source of routing anomalies, which can aid in debugging. The choices node 4 has regarding its interaction with node 7 in the cycle $P_{7X} \rightarrow P_{45,47} \rightarrow P_{41} \rightarrow P_{7X}$ are similar.

Since $\mathcal{MD}(\mathcal{S})$ was created with partial information, the absence of a cycle can be illusive. There can still be arcs connecting paths that belong to nodes with unknown policies. For instance, nodes $P_{7X}$ and $P_{5X}$ may have direct links between them. Nodes 1, 3, and 4 (that share their configurations) will not know whether nodes 5 and 7 exchange routes through unsafe policies, but they will be aware of the risk. This is no different from the case that safety is ensured through guidelines: following guidelines is up to each individual ISP.

## VII. Conclusions

In this work, we propose a methodology that allows ISPs to check their router configurations for safety. We leverage prior work on the Stable Paths Problem (SPP), but we bridge the gaps needed to make SPP applicable to a real-world implementation. Specifically, we extend SPP to reduce the complexity




(a) Specification $\mathcal{S}$.

(b) $\mathcal{MD}(\mathcal{S})$

Fig. 9. A specification where some of the nodes follow non-traditional policies. $\mathcal{MD}(\mathcal{S})$ is acyclic, so the specification is safe.

involved in analyzing BGP safety from configuration files. This allows us, in many cases, to only evaluate parts of the BGP best path selection process, without losing accuracy. We propose a new data structure, the multipath digraph, that is well-suited for detecting problematic conditions, and we prove properties that allow us to demonstrate the feasibility of applying it for verifying BGP safety in practice.

We apply our methodology to the partial information problem, since we expect our tool will never have complete configuration information. When checking configurations for a few ISPs, outside paths are at best partially known. Furthermore, ISP configurations may not always be complete. Our future work includes completing the implementation and evaluation of a tool that applies our methodology to real ISP networks.

## Acknowledgments

The authors would like to thank Randy Bush for his support, and Stefano Vissicchio and the anonymous reviewers for their insightful comments. This work has been sponsored in part by Cisco grant 570873 and NSF grant CNS-0831353.

## References

[1] Y. Rekhter, T. Li, and S.Hares, "A border gateway protocol 4 (BGP-4)," RFC 4271, http://www.ietf.org/rfc/rfc4271.txt, 2005.

[2] K.Varadhan, R.Govindan, and D. Estrin., "Persistent route oscillations in inter-domain routing," *Computer Networks*, vol. 32, pp. 1–16, 2000.

[3] R. Sami, M. Schapira, and A. Zohar, "Searching for stability in inter-domain routing," in *Proc. of IEEE INFOCOM*, 2009, pp. 549–557.

[4] T. G. Griffin and G. Huston, "BGP wedgies," RFC 4264, http://tools.ietf.org/html/rfc4264, 2005.

[5] L. Gao and J. Rexford, "Stable Internet routing without global coordination," *IEEE/ACM Trans. on Netw.*, vol. 9, no. 6, pp. 681–692, 2001.

[6] T. G. Griffin, F. B. Shepherd, and G. Wilfong, "The stable paths problem and interdomain routing," *IEEE/ACM Trans. Netw.*, vol. 10, pp. 232–243, April 2002.

[7] J. L. Sobrinho, "Network routing with path vector protocols: Theory and applications," in *Proc. of ACM SIGCOMM*, 2003, pp. 49–60.

[8] T. G. Griffin and G. Wilfong, "An analysis of BGP convergence properties," in *Proc. of ACM SIGCOMM*, 1999, pp. 277–288.

[9] L. Cittadini, M. Rimondini, S. Vissicchio, M. Corea, and G. Di Battista, "From theory to practice: Efficiently checking BGP configurations for guaranteed convergence," *Network and Service Management, IEEE Transactions on*, vol. 8, no. 4, pp. 387 –400, december 2011.

[10] A. Dhamdhere and C. Dovrolis, "The internet is flat: modeling the transition from a transit hierarchy to a peering mesh," in *Proc. of CoNEXT*, 2010, pp. 21:1–21:12.

[11] B. Augustin, B. Krishnamurthy, and W. Willinger, "IXPs: mapped?" in *Proc. of the 9th ACM SIGCOMM conference on Internet measurement*, 2009, pp. 336–349.

[12] A. J. Gurney, L. Jia, A. Wang, and B. T. Loo, "Partial specification of routing configurations," Department of Computer and Information Science, University of Pennsylvania, Tech. Rep., 2011, http://netdb.cis.upenn.edu/papers/partial-spec-tr.pdf.

[13] M. Yoshinobu, "What makes our policy messy," 2010, http://www.attn.jp/maz/p/c/bgpworkshop200904/bgpworkshop-policy.pdf.

[14] W. B. Norton, "A Study of 28 Peering Policies," http://drpeering.net/white-papers/Peering-Policies/A-Study-of-28-Peering-Policies.html.

[15] N. Feamster, R. Johari, and H. Balakrishnan, "Implications of autonomy for the expressiveness of policy routing," in *Proc. of ACM SIGCOMM*, 2005.

[16] L. Cittadini, G. Di Battista, M. Rimondini, and S. Vissicchio, "Wheel + ring = reel: The impact of route filtering on the stability of policy routing," *IEEE/ACM Trans. on Netw.*, vol. 19, no. 4, pp. 1085 –1096, aug. 2011.

[17] M. Suchara, A. Fabrikant, and J. Rexford, "BGP safety with spurious updates." in *INFOCOM*. IEEE, 2011, pp. 2966–2974.

[18] R. Viswanathan, K. K. Sabnani, R. J. Holt, and A. N. Netravali, "Expected convergence properties of BGP," in *Proc. of ICNP*, 2005, pp. 3–15.

[19] S. Epstein, K. Mattar, and I. Matta, "Principles of safe policy routing dynamics," in *Proc. of ICNP*, 2009, pp. 254–263.

[20] M. Arye, R. Harrison, R. Wang, P. Zave, and J. Rexford, "Toward a lightweight model of BGP safety," in *Proc. of WRiPE*, October 2011.

[21] D. Wetherall, R. Mahajan, and T. Anderson, "Understanding BGP misconfigurations," in *Proc. of ACM SIGCOMM*, 2002.

[22] N. Feamster and H. Balakrishnan, "Detecting BGP configuration faults with static analysis," in *Proc. of NSDI*, May 2005.

[23] G. Siganos and M. Faloutsos, "Analyzing BGP policies: Methodology and tool," in *Proc. of IEEE INFOCOM*, 2004.

[24] T. Griffin and G. Wilfong, "A safe path vector protocol," in *Proc. of IEEE INFOCOM*, vol. 2, 2000, pp. 490 –499 vol.2.

[25] J. A. Cobb, M. G. Gouda, and R. Musunuri, "A stabilizing solution to the stable path problem," in *Proc. of the 6th international conference on Self-stabilizing systems*, 2003, pp. 169–183.

[26] C. T. Ee, V. Ramachandran, B.-G. Chun, K. Lakshminarayanan, and S. Shenker, "Resolving inter-domain policy disputes," in *Proc. of ACM SIGCOMM*, 2007, pp. 157–168.

[27] T. G. Griffin, F. B. Shepherd, and G. Wilfong, "Policy disputes in path vector protocols," in *Proc. of ICNP*, November 1999.

[28] D. Thaler and C. Hopps, "Multipath issues in unicast and multicast next-hop selection," RFC 2991, http://tools.ietf.org/html/rfc2991, 2005.

[29] A. J. T. Gurney, L. Jia, A. Wang, and B. T. Loo, "Partial specification of routing configurations," in *Proc. of WRiPE*, October 2011.