A Logical Approach to Representing and Reasoning About Interdomain Routing Policies

Anduo Wang and Zhijia Chen

Temple University

Abstract. The Internet paths connecting independently operated networks, also called domains or autonomous systems (ASes), are driven by semantically rich policies: the interdomain routing protocol that computes the Internet paths allows the ASes to influence path selection with their local policies, such as economic concerns or operational constraints. An AS can promote a policy compliant but globally longer path by carefully tweaking lower level path attributes that are used in the routing protocol. Such operational policies are notoriously complex and hard to understand. *This paper takes a step back and asks whether a more principled logical approach can lead to AS policies that are easier to understand, reuse, evolve, and coexist.* To this end, we propose to represent policies by database integrity constraints, in the form of headless datalog rules about what are the acceptable Internet paths. The simple datalog expression unifies a wide spectrum of AS policies, ranging from classic examples seen in today's routing practice to futuristic ones proposed in various extensions to Internet routing. More importantly, by leveraging datalog's connection to the theorem proving technique called the residue method, we developed a new technique for understanding the interactions among the policies — whether a policy can inadvertly affect another, and how to resolve the conflict. We also evaluated our logical policies, showing promising result with small overhead for conflict resolution on realistic large networks.

1 Introduction

The Internet today is more than connecting remote hosts with valid paths. Rather, it is a unique artifact connecting independently operated networks [1], also called domains or autonomous systems (ASes), with rich AS-set policies [1]. Each AS needs to realize specific objectives driven by its own concerns on economics, performance, security and more, by configuring policies within its network to influence the overall route (a route is simply a path that carries end to end traffic) selection. To support such policies, border gateway protocol (BGP) [13], the de facto and the only interdomain routing protocol that establishes paths on the global Internet introduces various route attributes that can be set and compared to alter route selection. Over the years, the BGP route attributes have enlarged into an ordered list which implicitly coordinates network objectives by the ordering of the attributes [1, 26]; and new extensions to BGP are proposed to enable more flexible policies [18, 20, 12].

[1] In this paper, we use *networks* to refer to the networks under a single administrative domain that interconnected to form the global Internet.

Alongside the growing support for more flexible AS policies, the policy expression in terms of the route attributes remains largely ad hoc — the choice and use of the attributes are often arbitrary. For example, the `AS Path` (`ASP`) attribute — a list of ASes (each identified by a number) along the AS-level path — can implement the shortest path policy: when a route is first imported, the network attaches its own AS number to the `AS Path`, selects a best route with the shortest ASP, and exports to its neighbors this best route. However, a network can arbitrarily increase the length of `ASP` by appending its AS number multiple times to to make a longer path more preferable. An alternative to achieve the same affect is to leverage the `Local Preference` (`LP`) attribute which is compared first in route selection and thus overrides `ASP`: a network can promote the use of a longer path by assigning to it a higher `LP`. But `LP` is highly abused: many policies of disparate policies translate to `LP` only because it outweighs everything else.

Besides, a network is often influenced by many policies — separate teams within the network may issue policies addressing disparate aspects of the network, neighboring networks might also exercise policies that attempt to influence its route selection. These policies can interact in complex ways and run into subtle conflicts. The onus of combining them into a coherent route selection policy is left to the network admin who is required to carefully examine every component policy as well as their joint intent. For example, a security policy for avoiding a suspicious neighbor and an optimization policy for reducing route cost, though concerning distinct aspects of the network, can conflict due to the underlying topology: a router that connects a low-cost internal path (within the local network) to the suspicious neighbor can correlate the two policies with distinct purposes. In the presence of such conflicts, the admin must carefully re-factor and/or reset the corresponding policy attributes that are used in the prefixed BGP protocol.

Indeed, it is generally agreed that the policy practice of the Internet today, tightly coupled with the BGP protocol, is indirect and hard to understand. *This paper takes a step back and asks whether a more principled logical approach can lead to policies that are easier to understand, reuse, combine, and evolve?*

To this end, we investigate the use of database integrity constraints (ICs) — in the form of headless datalog rules about what are the preferable paths — for managing routing policies. The key idea is that, rather than treating policy making and route decision as a procedure that maps a set of incoming routes to one single best (outgoing) route, we view policies as logical constraints that prune the candidate routes until only the best route is left [2]. To validate its feasibility, we study the expressiveness and overhead of the IC representation with extensive case studies and experiments. More importantly, we show the benefits of this logical approach by a new conflict detection and resolution technique via the theorem proving technique called the residue method. Overall, we believe this is a first step towards a more principled approach to interdomain routing policies with automated reasoning support.

Specifically, this paper makes the following contributions:

[2] Or break tie by randomly select one route from a set of equally good routes

- We introduce database ICs as a logical representation for routing policies. Through extensive examples ranging from classic network policies to futuristic ones, we show how the IC representation unifies many disparate policies.
- We leverage and extend the residue method to study interactions and conflicts among policy ICs: we present a formal characterization of policy conflicts in terms of the notion of policy residue, a fragment of a policy that anticipates its impact on the other policies. To our best knowledge, this is the first formal attempt to characterize multiple routing policies that jointly control the same path. Our formalization is logically checkable and enables conflict detection. It also offers a method of conflict resolution with rewrites — when augmented with the residue, a potentially conflicting policy is semantically constrained and resolves conflict.
- We study the performance overhead of the proposed logical policies by evaluating a prototype of the residue method. Our result is promising, the conflict resolution delay is $< 100ms$ for large policies.

2 A unifying logical representation

This section develops a logical representation that attempts to unify a wide range of policies previously buried in disparate route attributes. More importantly, the representation will enable automated reasoning as a means to understand and combine policies, laying a firm foundation for automated conflict resolution 4.

2.1 A deductive database abstraction

We propose to represent network policies as database integrity constraints (ICs). We draw on the insight that viewing the network state as factual data — database tables and views — allows us to capture network policies as non-factual data — semantic information or ICs about what are the acceptable data.

Network state as relations and rules

A factual network state is a finite set of ground facts, denoted by I. We can also derive new information from the network state by rules, denoted by R. The complete network state is denoted by $N = R \cup I$. Suppose N always includes ground facts from the two route relations $r_i/3, r_o/3$, both of which have three attributes $destination, next\_hop$ and $path\_vector$. $r_i(x, y, z)$ stands for an incoming route learned from a neighbor y to reach a destination x by a path z (the list of ASes along the path to x). r_o stands for the output route selected by the network's routing policy.

For example, a network state that has 3 incoming routes and 2 outgoing routes is represented by the ground rules I_{1-5}, the derived information of all available paths to a particular destination `1.2.3.4` is represented by the rule R_1.

```
Network facts
I1: ri(1.2.3.4, router1, [AS2,AS3,AS5]) :-
I2: ri(1.2.3.4, router2, [AS2,AS4,AS5]) :-
I3: ri(1.2.3.5, router1, [AS3,AS5]) :-
I4: ro(1.2.3.4, router2, [AS2,AS4,AS5]) :-
I5: ro(1.2.3.5, router1, [AS4,AS5]) :-
```

```
Derived path information
R1: rpath(z) :- ri(x,y,z), x=1.2.3.4.
```

Policies as integrity constraints (ICs)

A routing policy is a finite set of ICs, denoted by P. We divide the policy ICs into two categories: the generative ICs whose head is non-empty, and denial ICs whose head is empty. In particular, a denial IC can be read as a body condition — conjunction of literals that cannot be simultaneously true — that must be avoided for any correct network state; a generative IC can be read as a head condition that must also hold if the network state is consistent with the body condition. A basic requirement for any routing system is path validity — any selected outgoing route must also correspond to some incoming route. A network cannot invent a route that does not belong to it. This can be captured by a path validity policy encoded by the generative IC $P_{validity}$, equivalently, it can be formulated as the denial IC $P_{validity'}$.

```
Path validity policy
ICvalidity:  ri(x,y,z) :- ro(x,y,z).
ICvalidity': :- ro(x,y,z), ¬ri(x,y,z).
```

2.2 Examples

In this section, we use more examples to illustrate the expressiveness of the IC representation.

Non-aggregate Policies

Many policies can be specified as constraints over a single path. For example, a source host might want to specify the explicit path (a) for carrying traffic to certain destination (d), to better control end to end performance, as expressed by the simple and direct IC_{cp}.

```
Complete path policy
ICcp: z=a :- ro(x,y,z), x=d.
```

Another popular policy regulates route selection based on the business relations of the neighboring networks. A network often engages in one of three business relations with its neighbors [16]: being a customer if it pays the neighbor to get to the rest of the Internet, being a provider if it charges its neighbor and provides Internet access, or being a peer if it exchanges traffic with its neighbor for free. To maximize revenue and minimize cost, it is common practice — formulated by the Gao-Rexford Guideline (GR) — for a network to prioritize (select) routes from the customer / peers over those from the provider. With the help of three auxiliary relations $customer/1, provider/1, peer/1$ that specify the source of a path — e.g.,`peer(x)` states x is a peer path learned from a peer — the Gao-Rexford policy guideline easily translates to the $IC_{GR1,GR2}$.

```
Gao-Rexford Policy Guideline
IC_GR1 :- r_o(x,y,z),r_i(x,y',z'),provider(z),customer(z').
IC_GR2 :- r_o(x,y,z),r_i(x,y',z'),provider(z),peer(z').
```

$IC_{GR1-GR2}$ prevents the selection of a provider path when there exists a customer or peer path, respectively. Note that the operational expression of the Gao-Rexford policy in traditional BGP is a classic use case of the `local preference (LP)` attribute: the network needs to always attach a higher `LP` to the customer / peer paths as compared to the provider paths. In comparison, our declarative policy is not only more direct — we do not need to coordinate the `LP` attribute assignment, it is also inherently more flexible. When bound to `LP`, Gao-Rexford will override any other policies as `LP` overrides all the rest of the attributes. For example, it will be hard to add a security policy even though it might carry more weight than the economic concerns. In contrast, the declarative policy allows us to add new policy and freely combine it with the existing ones with automated conflict detection and resolution (more in §4).

Our logical approach also easily captures futuristic BGP policies introduced in experimental BGP extensions. For example, MIRO [27] is an extension that allows a network to negotiate path with a neighbor. A common use of MIRO is to find securer path where a network sends its neighbor request to bypass certain suspicious network (say b). With the help of the auxiliary relation $waypoint(path, network)$, such negotiation request can take the form of IC_{MIRO}.

```
IC_MIRO :- r_o(x,y,z), waypoint(z,b).
```

Aggregate Policies

The foregoing examples are all non-aggregate policies in the sense that they are expressible as constraints along a single route. Next, we illustrate how to formulate aggregate policies concerning a group of routes *without* the explicit use of an aggregate term. First, consider the shortest path policy. Among all candidate paths connecting a pair of source and destination, shortest path policy picks a path that traverses the fewest number of networks. With the help of an auxiliary function $l(path)$ which returns the length (number of ASes along the path) of the $path$, shortest path policy translates to IC_{sp}.

```
Shortest path
IC_sp :- r_o(x,y,z), r_i(x,y',z'), l(z)>l(z').
```

Now consider a complex futuristic policy with aggregation. Wiser [12] is an extension to BGP that performs traffic engineering to jointly minimize cost between two neighboring networks. A Wiser network uses the (normalized) path costs advertised by a neighbor to join with its own (also normalized) local path — of course, the two path fragments must connect at a common border router — and selects a best path that has the lowest overall cost. With two auxiliary relations `Advertised(next_hop,cost)` and `Local(destination, next_hop,cost)`, the Wiser policy translates to IC_{Wiser}.

```
Wiser policy
R_Wiser: j(x,y,z) :- Local(x,y,z1), Advertised(y,z2), z=z1+z2.
IC_Wiser: :- r_o(x,y,z), j(x,y,w), j(x,y',w'), w>w'.
```

where the R_{Wiser} rule derives the joint cost of two path fragments and IC_{Wiser} forbids the selection of a route when there exists a better alternative (`w>w'`).

3 The Residue Method

An advantage of the IC representation is that it facilitates automated reasoning of the policies — predicating the interactions of multiple policies that jointly select paths. The key idea is to leverage the residue method [2] for semantic transformation, a process in which all the useful information in an IC is extracted and fully processed. Essentially, with the standard theorem proving technique called *partial subsumption*, we can anticipate the "impact" of a policy IC on another policy — a fragment of that IC called *residue*, which is then included in the second policy to eliminate any potential conflict. Notably, we extended the classic notion of partial subsumption with arithmetic and comparison, both of which are critical to routing policies.

Partial Subsumption

Given two policies C_1, C_2 of the form A_1 :- $\overline{B_1} \wedge \overline{B_1'}$ and A_2 :- $\overline{B_2} \wedge \overline{B_2'}$, respectively, where $\overline{B_1}, \overline{B_2}$ are conjunctions of relational literals and $\overline{B_1'}, \overline{B_2'}$ are conjunctions of comparison and arithmetic formulas, C_1 subsumes C_2 if there exists (1) a solver S for arithmetics and comparison, and (2) a substitution σ such that each literal in $(A_1$:- $\overline{B_1})\sigma$ occurs in A_2 :- $\overline{B_2}$ and $\neg\overline{B_2'} \wedge \overline{B_1'}\sigma$ can be reduced to False by S. Intuitively, subsumption means policy C_1 is more general — stronger — than C_2, that is, any network state compliant with C_1 is guaranteed to be compliant with C_2.

For example, consider the IC clauses IC_1, IC_2:

```
IC1: :- r_o(x,y,z), l(z)<5.
IC2: :- r_o(u,v,w), l(w)<3, w=[AS2,AS3].
```

IC_1 subsumes IC_2 with respect to a solver that is capable of arithmetic comparison because: (1) There is a substitution $\sigma = \{x = u, y = v, z = w$ that makes `:- r_o(x,y,z)` a subclause of `:- r_o(u,v,w), l(w)<3` and (2) $\neg((l(z) < 5)\sigma) \wedge (l(w) < 3)$ can be reduced to $l(w) > 5 \wedge l(w) < 3$ which is False.

What interests us is partial subsumption: C_1 partially subsumes C_2 if there exists a subclause of C_1 that subsumes C_2. Partial subsumption means the subsuming policy has a non-trivial impact on the policy being subsumed. To precisely capture such impact and to anticipate how such impact modifies another policy, we leverage the notion of residue.

Residue

A residue is a fragment of a subsuming IC that actually interacts with the subsumed IC. Residue can be obtained by the subsumption algorithm developed in [3]. The subsumption algorithm tries to construct a refutation tree with the subsuming policy as the root, and uses elements from the subsumed policy for resolution. Figure 1 shows the refutation trees for IC_{cp} and IC_2, IC_3 and IC_2, respectively. IC_{cp} and IC_3 are the subsuming clause and IC_2 is the subsumed clause. At the bottom of each refutation tree

where no resolutions or reduction (by the comparison-arithmetic solver) is possible, a fragment of the subsuming clause is left.

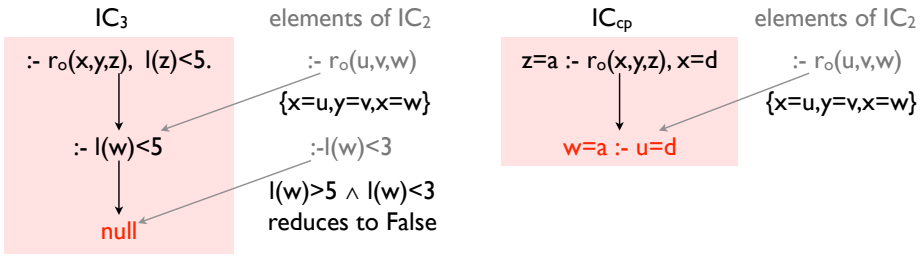

Fig. 1: Refutation tree: the residue (in red) at the bottom shows the impact of IC policy on route selection.

The residue precisely anticipates the policy "impact". The "null" residue on the left shows that IC_3 is stronger than IC_2 (more details in § 4); on the other hand, the non-trivial residue `w=a :- u=d` (right in the Figure) states that the condition $w = a \leftarrow u = d$ must be taken into account when applying IC_2: any route to d must take a as the selected path.

Semantic transformation via residue annotation

In general, a non-trivial residue obtained on a subsuming policy C_1 and a subsumed policy C_2 states a condition that must be satisfied for C_2 to be consistent with C_1. This observation allows us to use the residue of C_1 to semantically constrain (transform) C_2. We only need to attach the residue condition to the body of C_2. The result is a new policy C_2' that is guaranteed to be compliant with C_1. For example, Annotating IC_2 with the residue of IC_{cp} produces:

```
IC2 semantically constrained by Ccp:
:- ri(u,v,w),l(w)<3,w=[AS2,AS3], w = a ← u = d.
```

4 Merging Policies

This section developed a residue based method to understand the interactions between policy ICs 3. Specifically, using the notions of (partial) subsumption and residue, we give a formal characterization the relations among two policies. We also give a resolution method that automatically merges potentially conflicting policies.

4.1 Policy conflicts

First, we introduce a formal concept called *covering* to denote when a policy is stronger than another. One policy covers another if it subsumes the second policy. Intuitively, one policy is stronger if compliance with the first policy implies compliance with the second. For example, Consider a more specific form of shortest path policy that only finds

shortest path to certain destinations, with a new relation $only(destinations)$ standing for those destinations. We express this restricted shortest path policy by IC_{spr} as follows

$\texttt{IC}_{\texttt{spr}}$ `:- ro(x,y,z), ri(x,y',z'), l(z)>l(z'), only(x).`

The shortest path policy IC_{sp} subsumes IC_{spr}, thus it covers the more specific policy. Intuitively, all the shortest paths must also include the shortest path to those particular destinations.

What is more interesting is partial covering: one policy partially covers another if a subclause of the first policy covers the second. Partial covering signals conflicts — two policies overlap but neither one of them is redundant. Consider two pairs of policies — shortest path and complete path, MIRO and Wiser (in § 2). For each pair, the policies are conflicting with each other. It is straightforward to spot conflicts between simple policies like shortest path and complete path, but the conflicts between MIRO and Wiser is subtler. The formal characterization of conflicts via residue reasoning helps to guard against missing a subtle conflict.

4.2 Resolution

Conflict resolution in the Internet today is manual and operational: the network admin must understand the combined effort of all policies and refactor them into a hard wired list of route attributes that will be compared in a fixed order during the route selection process. In contrast, in this paper, we propose a facility that, takes a set of conflicting policies as input, generates a new coherent set by the semantic transformation process with the residue method. Between any pair of conflicting policies, the residue method rewrites the policy with lower priority, semantically constraining it with the other one with higher priority. Thus, the resolution facility automatically resolves conflicts by policy rewriting, freeing the admins from manual policy coordination.

Specifically, for two policies IC_i, IC_j, we use $IC_j|_{IC_i}$, to denote the semantically restricted version of IC_j with respect to IC_i, which is obtained by augmenting IC_j with the residue of IC_i. When IC_i enjoys higher priority over IC_j — IC_i is perceived more important, we can simply keep IC_i unchanged but compute $IC_j|_{IC_i}$ to rewrite IC_j to the restricted form, forcing IC_j to also account for the concerns of IC_i. Note that, by definition, IC_i covers IC_j. The resulting pair $(IC_i, IC_j|_{IC_i})$ can be viewed as a ranked list of policies where IC_i takes effect first, and the original IC_j follows.

For example, consider the complete path (IC_{cp}) and shortest path (IC_{sp}) policies. To obtain a shortest path policy that finds shortest path to all destinations except d, the path to which is pre-defined to be a, we can apply the residue method to obtain $IC_{sp}|_{IC_{cp}}$ (the residue augmented in red):

`:- ro(x,y,z), ri(x,y',z'), l(z)>l(z'),` $\neg(x = d \wedge z \neq a)$.

Thus, we obtained a shortest path policy that yields to the specific path assignment of the complete path policy.

Similarly, consider the MIRO (IC_{MIRO}) and Wiser (IC_{Wiser}) policies. A network that deploys both but prioritizes the MIRO policy — security concern over performance — can apply our resolution method to obtain the MIRO constrained Wiser policy $IC_{Wiser}|_{IC_{MIRO}}$:

`:- ro(x,y,z), j(x,y,w), j(x,y',w'), w>w',` $\neg$`waypoint(z,b).`

With $IC_{Wiser}|_{IC_{MIRO}}$, the Wiser portion (first 4 literals) will take effect only when the

MIRO residue evaluates to true (when MIRO policy is honored). That is, performance concern kicks in only after security requirement is satisfied.

5 Preliminary evaluation

We evaluate the performance of our proposed logical policies in Ravel [24], a database-defined network that uses the Postgres database as the network controller, which is particularly convenient for our experiment. All experiments were performed on the macOS platform with a 3.4 GHz Intel Core i5 processor and 16 GB 2400 MHz DDR4 RAM.

To measure the overhead of the residue method, we implemented a prototype of the subsumption algorithm based residue generation in Python: for a given subsuming clause (denoted by P_1, has M literals) and a clause being subsumed (denoted by P_2, has N literals), we loop over all the subclauses of P_1 and test if the subclause subsumes P_2. We applied the standard θ-subsumption algorithm to implement the subsumption test, which has complexity $O\left(N^k\right)$ as we need to map each literal of the subclause (ranging from 1 to k for a subclause with k literals) to a literal in P_2 (ranging from 1 to N) [17]. The overall complexity of the residue generator is $O\left(M^{\frac{M}{2}} N^M\right)$, where the term $M^{\frac{M}{2}}$ is a coarse upper bound for binomial coefficients.

We evaluated the performance of our residue generator with MIRO and Wiser policies. To test its scalability, we increased the number of waypoint literals in MIRO policy with randomly generated networks (to avoid).

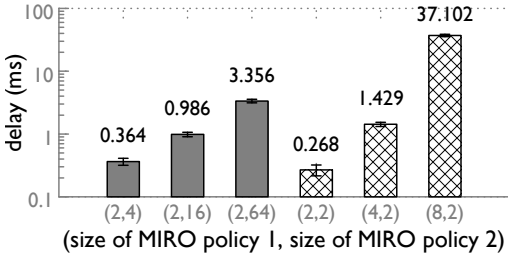

Fig. 2: Residue generation for MIRO policies.

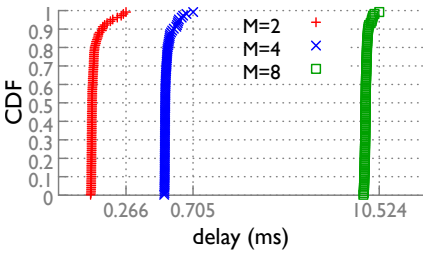

Fig. 3: Residue generation for Wiser and MIRO.

Figure 2 shows the residue generator processing delay (y axis, log scaled) for two MIRO policies with different size combinations (as indicated by the tuple below the x axis). For each combination, we plotted the average delay of 500 runs. The small bar on the top of each box indicates the standard variation. The left three boxes with grey filling show the growing trend of delay with respect to N — M is fixed at 2; and the right three boxes with grids filling show the growing trend with respect to M — N is fixed at 2. When M is fixed, the residue generator scales better — even for P_2 having very large size of 64, it finishes in 3.356 ms on average. The delay, however, increases

exponentially with M — the delay grows from 0.268ms to 37.102ms when M increase from 2 to 8.

Figure 3 presents the CDF of the residue generator processing delays (x axis, log scaled) for constraining MIRO policy with a Wiser policy — The Wiser policy has 6 literals ($N = 6$), and the MIRO policy has (M) has varied size of 2, 4 and 8. For all three sizes, 99% of all the processing finished within 0.266ms, 0.705ms and 10.524ms respectively. Note that the performance here is better than that of Figure 2 — for example, the average delay is less than 10ms for $M = 8$ and $N = 6$ compared to the average delay of 37.10ms for $M = 8$ and $N = 2$. This is because the MIRO policy shares only one common predicate with the Wiser policy, i.e., the relation r_o, which led to a faster result from subsumption test.

6 Related Work

Declarative networking.

Closest to our IC representation of routing policies is declarative networking [10, 9, 8, 7, 22, 21, 23] that also uses a database abstraction for specifying networks. But declarative networking focus on the factual network states. Network policies, as admitted by the authors, are still operational — they are merely logical modifiers that regulate (and are part of) the network operations to produce the policy-compliant outcome. Besides, it relies on the programmer for policy composition: programmers must clearly understand every piece of policy, as well anticipate their interactions and conflicts, and manually rewrite the program for policy transformation. Thus, declarative networking provides a (albeit declarative) platform for programming network states with embedded policies.

BGP policy extensions.

Many proposals [5, 27, 11, 28, 12, 25, 6, 19] were made in the past to enable more flexible routing policies. While these proposals often consider compatibility with the legacy BGP system, they do not provide any support for joint route decision among themselves. It is still the responsibility of the admin to come up with a monolithic coherent policy. The only work we are aware of that considers the co-existence of many BGP extensions is D-BGP [15, 14]. D-BGP studied architectural features needed to accommodate multiple interdomain protocols to be partially deployed across non-continuous domains called islands. Still, within an island, only one monolithic policy pertaining to a particular protocol can be deployed. To our best knowledge, our work is the first systematic attempt to facilitate multiple extensions to jointly affect routing within a network (island).

7 Conclusion

In this paper, we developed a logical approach to Internet routing policies. Rather than burying policies in the route attributes that are manipulated by the prefixed BGP protocol, and relying on the network admin for manual configuration, we introduce database integrity constraints (ICs) [4, 2, 3] — logical statements about what are the preferable routes — as a unifying abstraction for representing and reasoning about routing policies, and extend the standard subsumption algorithm to reason about the policies —

allowing automatic detection and resolution of conflicts. With extensive example policies and promising evaluation result of policy resolution, we believe a logical approach can make routing policies more direct, simpler and predictable.

References

1. CAESAR, M., AND REXFORD, J. Bgp routing policies in isp networks. *Netwrk. Mag. of Global Internetwkg. 19*, 6 (Nov. 2005), 5–11.
2. CHAKRAVARTHY, U. S., GRANT, J., AND MINKER, J. Logic-based approach to semantic query optimization. *ACM Trans. Database Syst. 15*, 2 (June 1990), 162–207.
3. CHANG, C.-L., AND LEE, R. C.-T. *Symbolic Logic and Mechanical Theorem Proving*, 1st ed. Academic Press, Inc., Orlando, FL, USA, 1997.
4. GODFREY, P., GRANT, J., GRYZ, J., AND MINKER, J. Integrity constraints: Semantics and applications. In *Logics for Databases and Information Systems (the book grow out of the Dagstuhl Seminar 9529: Role of Logics in Information Systems, 1995)* (1998), pp. 265–306.
5. GODFREY, P. B., GANICHEV, I., SHENKER, S., AND STOICA, I. Pathlet routing. In *ACM SIGCOMM* (2009).
6. GRIFFIN, T. G., JAGGARD, A. D., AND RAMACHANDRAN, V. Design principles of policy languages for path vector protocols. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (New York, NY, USA, 2003), SIGCOMM '03, ACM, pp. 61–72.
7. HINRICHS, T. L., GUDE, N. S., CASADO, M., MITCHELL, J. C., AND SHENKER, S. Fml: Practical declarative network management. In *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking* (New York, NY, USA, 2009), WREN '09, ACM, pp. 1–10.
8. LOO, B. T., CONDIE, T., GAROFALAKIS, M., GAY, D. E., HELLERSTEIN, J. M., MANIATIS, P., RAMAKRISHNAN, R., ROSCOE, T., AND STOICA, I. Declarative networking: Language, execution and optimization. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 2006), SIGMOD '06, ACM, pp. 97–108.
9. LOO, B. T., CONDIE, T., HELLERSTEIN, J. M., MANIATIS, P., ROSCOE, T., AND STOICA, I. Implementing declarative overlays. In *Proceedings of the Twentieth ACM Symposium on Operating Systems Principles* (New York, NY, USA, 2005), SOSP '05, ACM, pp. 75–90.
10. LOO, B. T., HELLERSTEIN, J. M., STOICA, I., AND RAMAKRISHNAN, R. Declarative routing: Extensible routing with declarative queries. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (New York, NY, USA, 2005), SIGCOMM '05, ACM, pp. 289–300.
11. MAHAJAN, R., WETHERALL, D., AND ANDERSON, T. Negotiation-based routing between neighboring isps. In *NSDI* (2005).
12. MAHAJAN, R., WETHERALL, D., AND ANDERSON, T. Mutually controlled routing with independent ISPs. In *NSDI* (2007).
13. REKHTER., Y., LI., T., AND HARES., S. A Border Gateway Protocol 4 (BGP-4).
14. SAMBASIVAN, R. R., TRAN-LAM, D., AKELLA, A., AND STEENKISTE, P. Bootstrapping evolvability for inter-domain routing. In *Proceedings of the 14th ACM Workshop on Hot Topics in Networks* (New York, NY, USA, 2015), HotNets-XIV, ACM, pp. 12:1–12:7.
15. SAMBASIVAN, R. R., TRAN-LAM, D., AKELLA, A., AND STEENKISTE, P. Bootstrapping evolvability for inter-domain routing with d-bgp. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (New York, NY, USA, 2017), SIGCOMM '17, ACM, pp. 474–487.

16. SAMI, R., SCHAPIRA, M., AND ZOHAR, A. Searching for stability in interdomain routing. In *IEEE INFOCOM* (2009).
17. SANTOS, J., AND MUGGLETON, S. Subsumer: A Prolog theta-subsumption engine. In *Technical Communications of the 26th International Conference on Logic Programming* (Dagstuhl, Germany, 2010), M. Hermenegildo and T. Schaub, Eds., vol. 7 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 172–181.
18. SCHAPIRA, M., ZHU, Y., AND REXFORD, J. Putting BGP on the right path: A case for next-hop routing. In *ACM SIGCOMM HotNets* (Oct. 2010).
19. SUBRAMANIAN, L., CAESAR, M., EE, C. T., HANDLEY, M., MAO, M., SHENKER, S., AND STOICA, I. Hlp: A next generation inter-domain routing protocol. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (New York, NY, USA, 2005), SIGCOMM '05, ACM, pp. 13–24.
20. SUBRAMANIAN, L., CAESAR, M., EE, C. T., HANDLEY, M., MAO, M., SHENKER, S., AND STOICA, I. HLP: A next-generation interdomain routing protocol. In *SIGCOMM* (2005).
21. WANG, A., BASU, P., LOO, B. T., AND SOKOLSKY, O. Declarative Network Verification. In *PADL* (2009).
22. WANG, A., JIA, L., LIU, C., LOO, B. T., SOKOLSKY, O., AND BASU, P. Formally Verifiable Networking.
23. WANG, A., JIA, L., ZHOU, W., REN, Y., LOO, B. T., REXFORD, J., NIGAM, V., SCEDROV, A., AND TALCOTT, C. FSR: Formal Analysis and Implementation Toolkit for Safe Inter-domain Routing. *IEEE/ACM Transactions on Networking* (2012).
24. WANG, A., MEI, X., CROFT, J., CAESAR, M., AND GODFREY, B. Ravel: A database-defined network. In *SOSR* (2016).
25. WANG, Y., AVRAMOPOULOS, I., AND REXFORD, J. Design for configurability: Rethinking interdomain routing policies from the ground up. *IEEE J.Sel. A. Commun. 27*, 3 (Apr. 2009), 336–348.
26. WANG, Y., AVRAMOPOULOS, I. C., AND REXFORD, J. Design for configurability: rethinking interdomain routing policies from the ground up. *IEEE Journal on Selected Areas in Communications 27*, 3 (2009), 336–348.
27. XU, W., AND REXFORD, J. MIRO: Multi-path interdomain routing. In *ACM SIGCOMM* (2006).
28. YANG, X., CLARK, D., AND BERGER, A. W. Nira: a new inter-domain routing architecture. *IEEE/ACM Trans. Netw. 15*, 4 (2007).