Received April 21, 2020, accepted May 8, 2020, date of publication May 12, 2020, date of current version June 9, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2994237

Enhanced Logical Representations of a Real Network Based on an Algebraic Model

YI PAN, SUIXIANG GAO, AND WENGUO YANG
University of Chinese Academy of Sciences, Beijing 100049, China

Corresponding author: Wenguo Yang (yangwg@ucas.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 11571015.

ABSTRACT The algebraic model and logical representations of a real network are important but very challenging issues in automatic network verification. In this paper, we abstract the concrete network to its corresponding abstract graph with enhanced vertices and edges by splitting the vertices according to the protocols that they run. Based on classical routing algebra, we consider combining the interactions of different protocols and routing records and then give a newly modified algebraic structure. To apply the abstract modified algebra routing into the concrete network, we make use of the SMT solver to encode the components of algebra into logical constraints motivated by the work of Ryan Beckett *et al.* By encoding the network behaviors and properties into logical formulas, we can compute whether the experimental network that we configured satisfies any property, including reachability, routing loop, and so on. In the previous work, the routing algebra or the representation of the network is only applied to several network properties. However, in this paper, we extract all kinds of properties into exact logical formulas.

INDEX TERMS Logical formula, network verification, network protocols, routing algebra, SMT.

I. INTRODUCTION

A network is composed of a control plane configuring the behavior of the data plane, which in turn is in charge of forwarding the actual packet. Therefore, the control plane can take configuration files and the current network environment and then generate the data plane. The data plane can be treated as a director that tells a packet its next station. The data plane takes a packet and its location as input and outputs a packet at a different location.

To ensure that a large scale of the network runs correctly, engineers have to spend considerable time writing the configuration files and maintaining the network. They have to ensure that each routing protocol locally computes a set of correct routes for the routers and collaborates well with each other protocol. However, most of the time, the oversight of the configuration files may lead to a terrible disaster. These issues can be addressed by adding a level of abstraction separating the high-level decisions of what a routing protocol should compute and how it should be configured from the low-level implementation files of how it performs the necessary computation [1]. The behaviors of each device in a real network are complicated due to the different mechanisms of each routing protocol and the interactions among them. Obtaining a unified and abstract algebraic representation of the network at a high level is worth researching. The purpose of routing algebra is to model the complex routing protocols into a high-level abstraction structure to obtain the stable state of the network. As is known, network devices use several protocols to exchange messages and forward packets according to their destinations. The configuration files of each device decide how these routers or switches process packets and select paths for them. These files usually have thousands of command lines and are vendor specific, which makes it hard to determine the behavior of the network. Based on this approach, the low-level configurations of devices are modeled into concise specifications. The abstract models of kinds of tools enable engineers to handle these tasks and networks with high efficiency and reliability.

The earliest network diagnostic tools, such as traceroute and ping, can analyze whether and how a given packet reaches its destination to find the configuration errors. The issue is that they have poor data plane coverage. In other words, they just analyze the forwarding behavior for only a single packet for the data plane but ignore the corresponding packets.

HSA [2] and Veriflow [3] cover the data plane better to analyze the data plane behaviors. The problem is that it is too

The associate editor coordinating the review of this manuscript and approving it for publication was Eyuphan Bulut.

late to detect errors from the data plane because errors have occurred. An effective way is to translate network configuration files into logical formulas capturing the stable states to which the network forwards. This method will converge as a result of interactions between routing protocols, such as OSPF [4], BGP [5], [6], and static routes. Minesweeper [7] tries to achieve both high network design coverage and high data plane coverage while remaining scalable enough to enable verification of many real networks. This approach models the whole network instead of a single path by using a graph-based model. Encoding all interactions of the route messages, there are many logical constraints on these nodes and edges of the network. Moreover, these network properties, such as reachability, isolation, and routing loop, can also be translated to logical formulas according to the constraints of the network behaviors. By using an SMT solver [8], which combines the constraints of both network behaviors and the network property that we check, we can obtain the computational result.

Covering the control plane is hard work because devices may run different kinds of routing protocols. From this point of view, our goal is to modify the basic routing algebra to model the dynamic network and explore more abstract and concise logical representations of network behaviors and properties based on our approach of the algebraic model.

Much research focuses on raising the level of abstraction of network configuration files and the mechanism of routing protocols, which is an effective and unified way to synchronize the structure of different networks [9], [10]. However, it is difficult to contact algebra with real network behaviors. In this paper, we advocate an approach to the algebraic structure of routing based on the aforementioned algebraic models. There are mainly two significant improvements to our contributions. The first is the definition of the network graph. We split the nodes in the directed graph according to the multiple protocols that they run. In this way, we can clearly analyze and represent the interactions of protocols and forwarding behaviors. Moreover, we add the routing records part into the algebraic model to catch the interactions between different routes rather than following a single route. The second point is that we try to concretize the abstract algebraic model by logical representations. Based on the modified routing algebra, we complete the encoding of the network behaviors and properties into logical formulas and obtain all the concise logical representations of the main ten network properties.

The paper is organized as follows. Section II illustrates the procedures of modeling a real network using our modified routing algebra. First, we model the real network into a directed graph G as basic routing algebra does. Then, we illustrate how we split the nodes for the different protocols that they run. Thus, we give a new model of the network as $\mathcal{G}$ and show the structure of our approach to algebra routing by adding the routing records and filtering function. Thus, a high-level abstract structure to model the dynamic route changes in the real network is obtained. In section III, we apply the modified routing algebra to encode the whole network. At the highest level, the operating progress of the network can be understood as the mechanism and policies of routing protocols and the way that the devices process them. Then, we can divide the encoding model into three parts: data plane, control plane, and routing policy. We show the encoding details of network properties in section IV. It is important to point out that we encode all ten kinds of network properties in abstract logical formulas, which are not done completely in the previous work. Section V concludes with our contributions, a discussion of directions for future research, and practical use case scenarios.

II. ROUTING ALGEBRA

Recalling the basic routing algebra, Griffin *et al.* showed that network control planes solve the stable path problem [11], and these paths can be described by constraints on edges. In their work, each node represents an autonomous system, and these nodes process the BGP. BGP is referred to as a path-vector protocol that implements its routing policies by route announcement passed between routers or switches [12]. BGP announcements are represented as records that have several attributes. The basic routing algebra models the network into a directed graph and comprises a set of labels, a set of signatures, and a set of weights. Each network link has a label, and the path in the network has a signature. There is an operation to obtain the signature of a path from the labels of its constituent links and a function mapping signatures to weights [13]. Some related work is enriched by properties that combine its elements, allowing stronger statements about protocol convergence to emerge and permitting a computationally easy characterization of free cycles. Motivated by previous work [1], [10], [14], [15], we modify the routing algebra to combine routing records with basic algebraic structure, which is more helpful to verify the network properties. For ease of encoding and computing, we assume that the network is already convergent. We prefer the relation of routing algebra and logical formulas to the convergence of the network.

A. MODEL A NETWORK TO A GRAPH

Existing algebraic models mostly reason a single path from a certain source node to its destination. In data plane analysis, this method can effectively obtain results, such as HSA. Because of ignoring the interference between different routes, this approach can cause problems in real network analysis, especially in control plane verification. For example, several routes may use a common edge to forward traffic. If the edge is down, these routes all fail to reach destinations. Encoding a network into a graph can combine associated routing messages to analyze the current route. To represent the network in a graph, we first abstract the network to graph G,

$$\begin{aligned} &G = (V, E, W) \\ &V : \textit{Nodes in the network}. \\ &E : \textit{Links between each pair of nodes}. \\ &W : \textit{Metrics of each link}. \end{aligned} \tag{1}$$

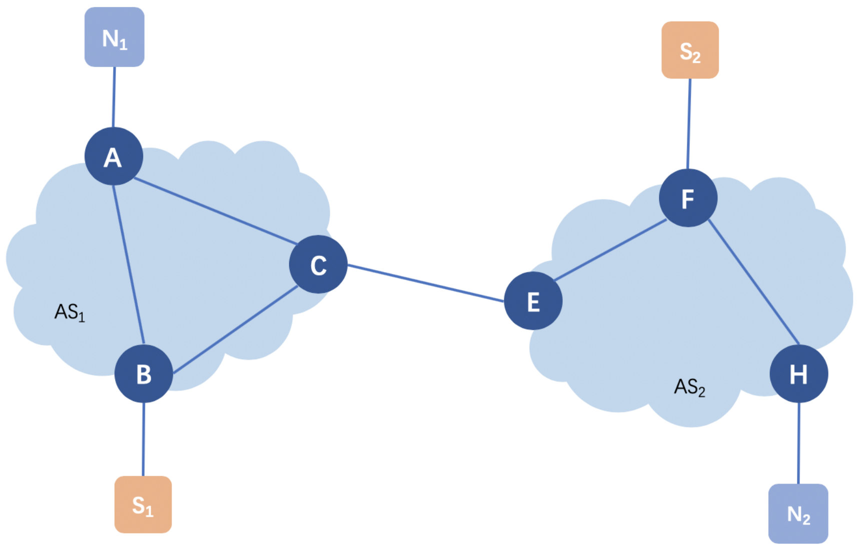

FIGURE 1. **Sample network.**

Then, we need many logical constraints to model the nodes and links. The more complicated the network behavior is, the more logical formulas are needed, but we only compute those relevant constraints to obtain the result.

Figure 1 presents a sample network to illustrate the dynamic state of the network. There are two autonomous systems, AS1 and AS2. Each AS has three internal routers, A to C and E, F, and H, which runs OSPF protocol separately. For AS1, A connects to an external neighbor N_1 via an external BGP. C connects AS2 through E running the external BGP as well. A, B, and C share the external BGP route messages from N_1 via the internal BGP. For AS2, H receives routes from external neighbor N_2. E, F and H process the OSPF protocol to deliver route messages.

Suppose N_1 tries to establish a communication with N_2. What will the routing path then be? First, A receives external BGP routes from N_1 and then announces to the routers in AS1. Next, A chooses a proper path to send the route out of the current AS. According to the short path first algorithm of OSPF, A selects a lower-cost path to reach C, which is the border router of AS1. The route will then be passed to E via the external BGP and passed through AS2, finally reaching N_2.

Let us assume that each link in AS1 has the same cost. Then, the shorted path from A to C is the edge between them. Thus, the path from N_1 to N_2 is $N_1-A-C-E-F-H-N_2$. However, if the link between A and C breaks down, A will choose to reach C by way of B. Then, the path will be $N_1-A-B-C-E-F-H-N_2$. We represent all the possible outcomes to logical formulas and to compute whether N_1 can reach N_2. By using modern SMT solvers, we can obtain the result to check the reachability.

To model the routing messages in more detail, the form and definition of graph G are modified. First, we extend the topology of the graph by splitting the nodes [7], [16]. The splitting number of a node depends on the number of protocols associated with the current route running on this node. For example, in figure 1, router C runs OSPF with its companies in AS1 and external BGP exchanging messages with router E in AS2. Router C splits to C_{OSPF} and C_{eBGP} two vertices. The number of edges E will increase with the splitting of nodes. The edge between C_{OSPF} and C_{eBGP} represents the protocol interaction of OSPF and the external BGP process on C. Moreover, the metric on this edge is also meaningful. Then, we can modify the graph as $\mathcal{G}$,

$$\begin{aligned}&\mathcal{G}=(\mathcal{V},\mathcal{E},\mathcal{W})\\&\mathcal{V}: \textit{Nodes splitted in the network}.\\&\mathcal{E}: \textit{Links between each pair of splitting nodes}.\\&\mathcal{W}: \textit{Metrics of each link}.\end{aligned}\tag{2}$$

Hence, we have $|\mathcal{V}|\geq|V|$, $|\mathcal{E}|\geq|E|$.

In general case, different paths can interfere with one another. To make a proper and correct analysis, all possible interactions should be taken into consideration. In our modified graph model, all possible paths can be constructed into a synthesized topology model, $\mathcal{G}$. Then, the complexity of this structure should be $\mathcal{O}(\mathcal{V}+\mathcal{E})$. However, in many existing network models, the complexity is up to $\mathcal{O}(V^{\frac{E}{V}})$.

B. ABSTRACT THE GRAPH TO ALGEBRA STRUCTURE

The control plane defines the behaviors of each router and the rules regarding how they treat different routing messages. The enhanced graph of a network is a set of nodes and edges. Based on the algebraic models of routing [10], [13], [17], [18], we try to combine the routing records and algebraic models to advocate a new destination-oriented algebraic structure. The destination-oriented refers to the fact that though the nodes of the graph model all routers in the network, we only consider those associated with the destination that we need to check. To model different routing protocols, we use a set of attributes to distinguish the different attributes of each protocol, such as the local preference and as-path of external BGP, the hop counts of RIP, and the cost OSPF. Then, we use $\mathcal{R}$ to model the routing records. When a router receives several routing messages to the same destination, it may select the best one and process it according to some filtering rules as configured. Thus, we also need a notation Σ to model the routing attributes and a function $\mathcal{F}$ to model filters and routing policies. Then, we can define the algebraic structure as

$$\begin{aligned}&(\mathcal{G}, \Sigma, \preceq, \mathcal{R}, \mathcal{F})\\ &\mathcal{G}: \quad (\mathcal{V}, \mathcal{E}, \mathcal{W})\\ &\Sigma: \quad \textit{routing attributes.}\\ &\preceq: \quad \textit{comparison order over } \mathcal{R}.\\ &\mathcal{R}: \quad \textit{routing records.}\\ &\mathcal{F}: \quad \textit{filtering function.}\end{aligned} \tag{3}$$

Here, r is a routing record associated with a certain destination d, and $r \in \mathcal{R}$. To model the records in more detail, each r is a tuple of attributes. For example, for an external BGP routing record, r can be written as a vector of attributes, such as $r(ad, lp, as\_path, med, \ldots)$. We use ϕ to represent a missing attribute. Suppose r_0 is a BGP routing record with *med* value missed; then, r_0 can be denoted as $r(ad, lp, as\_path, \phi, \ldots)$.

The $\preceq$ operates on the routes in lexicographic order. In other words, $\preceq$ can be treated as the total order that ranks and models the routing decision procedure using some combination of record fields [19], [20]. For example, if a router receives two records of different protocols, we just compare the *ad* values of the two records to obtain a better one. For the records of a same protocol, suppose we have two BGP routing records $r_1(lp_1, as\_path_1, med_1, \cdots)$ and $r_2(lp_2, as\_path_2, med_2, \cdots)$. To compare r_1 and r_2, we first compare the values of lp_1 and lp_2. If they are the same, we then compare the values of as_path_1 and as_path_2 and so on. Thus, we compare the two records in lexicographic order, and only the corresponding attributes can be compared, such as lp_1 and lp_2. Then, lp and as_path cannot be compared. Although a record r contains different attributes, any two records can be compared to obtain a better one.

The function $\mathcal{F}$ maps pairs with a node and a routing record to a record, which is

$$\begin{aligned}\mathcal{F}: &\quad (\mathcal{V}, \mathcal{R}) \longrightarrow \mathcal{R}\\ f: &\quad (x, r) \mapsto r\end{aligned} \tag{4}$$

where $x \in \mathcal{V}$ is a node and $r \in \mathcal{R}$. The filtering function describes how routing records are modified, especially dropped by the current router.

A well-defined algebra structure of routing needs to satisfy two properties [20], *loop free* and *faithful*. The *loop free* is different from the network property routing loop.

Property 1 (Loop Free): A graph must not contain self-loops.

$$\forall x \in \mathcal{V},\ (x, x) \notin \mathcal{E}$$

Property 2 (Faithful): The routing records are strictly mapped from the routing messages from the network. If router x blocks the current record, we have

$$f(x, r) \longrightarrow r_\phi$$

Here, r_ϕ represents that the record is dropped. Then, router x will not advertise a record to its neighbor.

Since we model the network to a graph with a well-defined algebraic structure, we need an approach to concrete the algebraic structure to calculate the real network behaviors and verify some network properties. Motivated by the previous work [7], we try to encode the algebraic structure to logical constraints. Then, it is possible to transfer the symbols of the algebra to the corresponding logical variables.

III. THE BASIC NETWORK MODEL

In this section, we model the algebraic network into a set of SMT constraints. We first traverse the signatures in Σ to their corresponding variables. The variables are set proper ranges to encode certain attributes of protocols or packets. For example, *local preference* is an important attribute of BGP; then, we can encode it with *lp*, and $lp \in \Sigma$. According to different routing records, these variables of attributes form constraints, which include the network behaviors for a certain packet we might concern and the network environment that might influence the behaviors. To model the network state in detail, we also set constraints to the nodes in $\mathcal{V}$ and edges in $\mathcal{E}$, such as variables encoding the *active* or *down* state of a node or a link. For ease of calculation, we denote these constraints to the notation $\mathcal{C}$ and denote the network property that we want to check as $\mathcal{P}$. By using a modern SMT solver, such as z3, we combine $\mathcal{C}$ and $\neg\mathcal{P}$ to obtain the verification result. Why do we solve $\neg\mathcal{P}$ instead of $\mathcal{P}$? The principle of z3 is interesting. If the network behaviors of a packet $\mathcal{C}$ are consistent with the network property, we do not need a specific outcome. In contrast, if $\mathcal{C}$ violates the property $\mathcal{P}$, we do need to locate the problem; thus, the computation result shows the counterexample.

Based on the process of passing a packet from the source to its destination, we model the network into three parts.

1) data plane
2) control plane
3) routing policy

A. DATA PLANE

To model the data plane into symbolic representation, we encode the header space of the packet. There are five important elements: source IP address, destination IP address, source port, destination port, and protocol. When a packet arrives at the router, the router needs to know where the packet is from and where to go. The symbolic notation we turn to the work of Beckett *et al.* [7]. The first is the source IP address of the current packet, which is modeled by an integer variable *sIp*. The destination IP address is modeled as *dIp*. Both of them range from 0 to $2^{32}-1$. We also need the source port and destination port of the packet. Then, they are denoted as *sPort* and *dPort*, which both range from 0 to $2^{16}-1$. Additionally, the protocol in which the routers process is encoded as *prot*. Apart from the integer variables, the forwarding decision in the data plane should also be encoded. Thus, a Boolean viable $choice_{x,y}$ encodes whether router x forwards traffic to router y in the data plane. If router x finally decides to forward a packet to router y in the data plane, then $choice_{x,y}$ equals 1. If not, it equals 0. Because routers will be equipped with filtering policies, some routes to certain destinations may be blocked, and then, we also need a variable to encode the effectiveness of the routes. We list the main variables in table 1.

TABLE 1. Dataplane variables.

Variable	Representation
sIP	the source IP address of the packet.
dIP	the destination IP address of the packet.
$sPort$	the source port of the packet.
$dPort$	the destination port of the packet.
$prot$	the protocol of the packet.
$choice_{x,y}$	router x decides to forward y.
r_{valid}	whether the routing record is blocked.

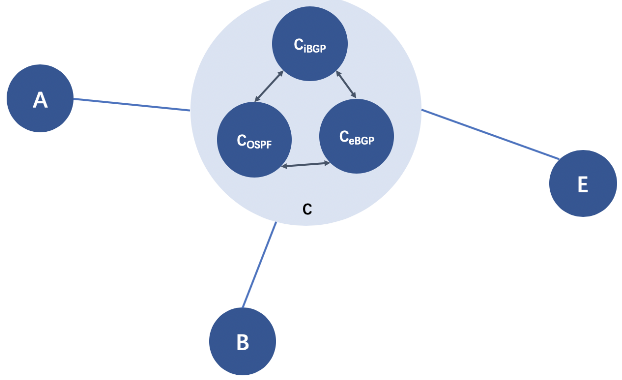

FIGURE 2. Multiple R_2.

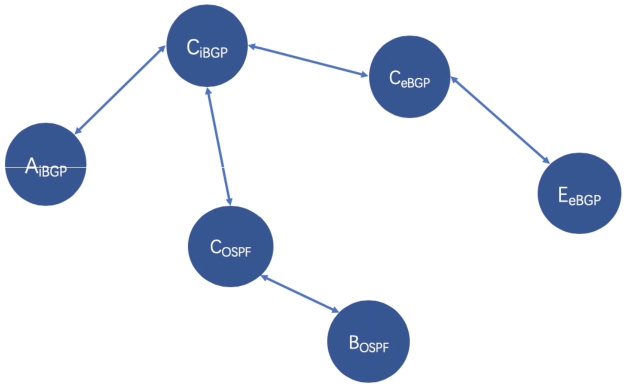

FIGURE 3. Exchanging messages.

B. CONTROL PLANE

The control plane is difficult to model due to the complexity of each routing protocol and the interactions among them. A network equipped with relatively complete functions usually processes several different protocols. In an autonomous system, OSPF, IS-IS, or RIP may be used, and the external BGP runs across multiple autonomous systems. Additionally, static routes and connected routes can be configured in the same network. Routers in the network usually process multiple routing protocols. For example, in figure 1, router C receives a route from A via the internal BGP and then announces the route to B via the OSPF and to E via the external BGP. To model the routing messages clearly, we try to split each node in the network into multiple nodes according to the protocols that it runs.

In figure 2, we illustrate the splitting of C. Routing messages are exchanged and processed among C_{iBGP}, C_{eBGP}, and C_{OSPF}. We turn to the work of Ryan *et al.* to use records of symbolic values corresponding to protocol messages. It is difficult to model the message exchanges among different protocols because each protocol has specific values to filter and process the routes. For instance, BGP has various values to evaluate routing messages, such as as as-path, local preference, metric, and multi-exist discriminators. OSPF runs the short path first algorithm; then, it needs a cost value to record the metric of a path. RIP needs a hop value to count the metric, and in SMT constraints, the hop is less than 15.

We can roughly take three steps to follow the process of the router treating a message. First, it needs to filter the message. Then, the current protocol processes the message. Third, the message traverses an export filter and is then exported to the next router. To model the procedure, we turn to the arc-oriented symbol, and we encode the routing record r into r^o and r^i. For example, in figure 3, considering the edge between C_{OSPF} and C_{eBGP}, we give the edge a tuple of symbols, r^o represents the record r exported by the OSPF process of C. r^i encodes the message after traversing the import filter of C's external BGP process. The r^i and r^o symbols are the records of the routing message. In other words, each of them has the values associated with each protocol. Specifically, $r^i.ad$ encodes the administrative distance value of the imported record. For BGP, $r^i.lp$ encodes the local preference value of the imported record. For OSPF, $r^o.cost$ encodes the cost of the current path, and so on. The type of these variables is integer; thus, we can map them to specific values in SMT constraints.

TABLE 2. Controlplane variables.

Variable	Representation
$r.prefix$	The prefix of a route record.
$r.length$	Prefix length of a route record.
$r.ad$	Administrative distance of a route record.
$r.lp$	Local preference attribute of BGP.
$r.med$	Med attribute of BGP.
$metric$	Protocol metric.
$cost$	Weight of a path for OSPF.
rid	Router ID to break ties.
$ibgp$	Mark the route learned from internal BGP.
$canUse_{x,y}$	Router x decides to forward y in control plane.

We list the main variables in the control plane in table 2. For convenience, we encode the routing messages as records, and each record includes the variables according to different protocols. When a router receives a routing record, it first compares the prefix or the record with its prefix of the destination IP address. The longest IP prefix matched will be chosen. The variable r_{prefix} ranges from 0 to $2^{32}-1$. Administrative distance (ad) is the feature that routers use to select the best path when there are two or more different routes to the same destination from different routing protocols. Each routing protocol is prioritized in order of most to least reliable (believable) with the help of an administrative distance value. The variable ad ranges from 0 to 2^8-1. For BGP, the records include some basic attributes to integer variables. For example, local preference (lp) is the BGP attribute ranking after administrative distance. If there are several paths to the same destination, the local preference attribute with the highest value indicates the preferred outbound path from the local autonomous system. The Boolean variable $ibgp$ is used to mark the route via the internal BGP. The metric is also contained in these records, which is measured distinctly in different protocols. For BGP, the metric is the length of AS path, and for OSPF is the cost of a path. When a route is redistributed, the metric usually has an initial value, and the value depends on the configurations. After all the steps of the route selection, if several routes are still equally good, we need the router ID(rid) to break ties. Moreover, we need a variable $canUse$ to encode the forwarding behavior in the control plane.

To make a forwarding decision in the control plane, the router needs a valid routing record, which is $r_{valid}=1$, and the next hop is the best choice among all possible choices. Then, we can encode the forwarding decision in the control plane as

$$canUse_{x,y} = (r_{valid} \wedge r = best_{all}) \tag{5}$$

where $best_{all}$ denotes the highest-ranking symbolic record when comparing multiple possible choices, which is explained in detail in section *route selection*. The forwarding decision in the control plane and the rules of routing policies, such as ACLs or IP prefix lists, contribute to the final forwarding decision in the data plane. Then, the relationship between $canUse$ and $choice$ can be

$$choice_{x,y} = canUse_{x,y} \wedge \mathcal{F}(r) \tag{6}$$

If the filtering function drops the routing record for some specific ACL rules, then $\mathcal{F}(r) = r_{\phi}$. Thus, we can obtain $choice_{x,y} = 0$ in the data plane.

1) ROUTE SELECTION

When a router has multiple choices to the same destination, it selects the best paths. In the current protocol process, the available paths are ordered in a uniform standard. For example, OSPF uses Dijkstra's algorithm to compute the minimum cost paths from a source to destinations, and the router chooses the min-cost path as its local best route. For the BGP, available routes are ordered according to the rank of those attributes, and the router that processes the BGP selects the high-rank path as its local best path. When a router can reach its destination via multiple different protocols, it selects a prior rank protocol to process according to the value of these protocols' administrative distance. As mentioned above, we turned to the symbolic variable [7] r to denote the record of a routing message. Then, we modify the form of r. A record r contains a series of attributes. Taking the BGP as an example, a record r can be treated as $r(ad, lp, metric, as\_path, origin, \ldots)$. Suppose a router receives two BGP routing records r_1 and r_2, which announce that they all can reach the same destination. The router needs to select a better record. Then, the records r_1 and r_2 are compared lexicographically. $r_1.lp$ and $r_2.lp$ are first compared, and BGP prefers the higher value. If those are equal, BGP prefers the higher value of $metric$ and then shorter as_path, and so on. OSPF prefers a lower cost path. The RIP selects a lower hop count path. Encode this mechanism in SMT constraints in the following form:

$$r_1 \preceq r_2 \tag{7}$$

where $\preceq$ means that record r_1 is superior to r_2 or at least as preferred as r_2 in lexicographic order.

If a router is configured with k routing protocols, then each protocol maintains a routing information base in the control plane. Then, we can encode a symbolic variable $prior_i$ for each protocol instance i. Here, i denotes the ith protocol and $1 \leq i \leq k$. The variable $prior_i$ is equated with the highest available routing record in the order for each protocol instance i. After comparing all the possible best paths $prior_i$ of each protocol, according to the administrative value of each protocol, the router finally chooses the best one to use, which we denote as $best_{all}$. More importantly, the $best_{all}$ record is superior to any of the $prior_i$ and at least equals one of them.

$$(best_{all} \preceq \bigwedge_{i\in\{1,\cdots,k\}} prior_i) \wedge (best_{all} = \bigvee_{i\in\{1,\cdots,k\}} prior_i) \tag{8}$$

Here, the inequality corresponds with the definition of the signature $\preceq$ in routing algebra.

2) ROUTE REDISTRIBUTION

Route redistribution allows a network that uses one routing protocol to route traffic dynamically based on information learned from another routing protocol. Each router usually runs more than one routing protocol instance. These routing protocol instances operate independently and announce or receive the best route from others. Suppose a router R runs two protocols, p_1 and p_2. There are k_1 routers running the p_1 protocol and k_2 routers running the p_2 protocol, and the two routing protocol instances communicate through R. As mentioned above, p_1 and p_2 operated independently; then, we only consider the symbolic variables on the link between R_{p_1} and R_{p_2}. Due to the different metric standards of each protocol, such as the cost of OSPF, the length of BGP and the hop count of RIP, we modify the metric attribute of variable r only when r is to be redistributed. The metric is only a value according to the configuration files.

In figure 3, when calculating the formulas of route redistribution, we only consider the variables on the link between C_{OSPF} and C_{eBGP}. Now we consider the variable *metric*. When in an external BGP instance, *metric* represents the number of AS of the external BGP path. When the routing record is redistributed from the external BGP to OSPF, the *metric* is defined to the default value that is understandable to the receiving protocol (OSPF) to unify the measure.

C. ROUTING POLICY

All routing protocols place their routes into the routing information base or routing table. When advertising routes, the routing protocols by default advertise only a limited set of routes from the routing table. Moreover, each routing protocol exports specific routes according to the configuration files.

Routing policies enable operators to filter which routes a routing protocol imports into the routing table and which routes a routing protocol exports from the routing table. We can conclude this function as the import filter and export filter. A routing policy also enables operators to modify the information associated with a route when it is being imported into or exported from the routing table. Filtering imported routes enables the current router to control the routes used to determine active routes. Filtering routes being exported from the routing table enables the router to control the routes that a protocol advertises to its neighbors. Operators usually use ACLs or IP prefix lists to achieve filtering routes. Taking a configuration fragment of Cisco reference as an example,

$$access - list\ 1\ permit\ ip\ 192.168.1.0\ 0.0.0.255$$

The mask is used with IP addresses in ACLs to specify what should be permitted. This ACL rule will permit any packet whose destination IP prefix matches the first 24 bits of 192.168.0.1. Thus, we can convert this kind of rule into computable function $h(p, n)$ technically, which takes the first n bits of p. For the above ACL fragment, we have the following logical formula.

$$h(r_{dstIP}, 24) = h(192.168.1.0, 24)$$

The IP prefix list is more flexible for controlling routes. The operators can configure prefix lists to match an exact prefix length or a prefix range. For example, considering the following IP prefix list,

$$prefix - list\ 2\ permit\ 192.168.0.0/16\ le\ 24$$

This rule creates a prefix list that will accept a netmask of up to 24 bits (*le* meaning less than or equal to) in routes with the prefix 192.168.0.0/16. We can encode this IP prefix list as

$$(h(r_{dstIP}, 16) = h(192.168.0.0, 16)) \wedge (r_{length} \leq 24)$$

Additionally, when a routing table imports routing information from a routing protocol, a routing policy can modify the route's preference. When a routing table exports routes into a routing protocol, a policy might assign metric values, modify the BGP community information, tag the route with additional information, or prevent the route from being exported altogether. Then, in our logical formulas, we can modify the corresponding attributes of the record r. For instance, if router R is configured to change the local preference value of an external BGP route to 100, then we set $r_{lp} = 100$.

IV. NETWORK PROPERTIES

The operators usually test some common properties to verify the network runs as expected. In our representations of the network, these network properties are also encoded into logical formulas. Combining the constraints of network behaviors and properties, we can solve whether a network holds such a property that we want to check. Here, we denote the network behaviors as $\mathcal{C}$, a certain property as $\mathcal{P}$. If $\mathcal{C} \wedge \mathcal{P} = 1$, then the network holds the property. However, to capture more details of the network, we use the SMT solver to compute $\mathcal{C} \wedge \neg\mathcal{P}$. If the result is satisfiable, the network does not hold the property that we check, and a counterexample is obtained. On the other hand, if the result is unsatisfiable, the property of interest is held. We illustrate the encoding of several common network properties.

A. REACHABILITY

To verify whether a source node can reach the destination node, we set an additional integer variable $hopCount_{x,S}$[1] to each node and initiate the variable to -1 at the beginning. If a node does forward to its neighbor in the data plane, then we update the hop count of the neighbor by increasing $hopCount$. Specifically, the $hopCount_{S,S}$ of the source node equals 0. For a certain router, if its hop count is greater or equal to 0,

[1]The two variables at subscript represent the hop counts from the source node S to the current node R.

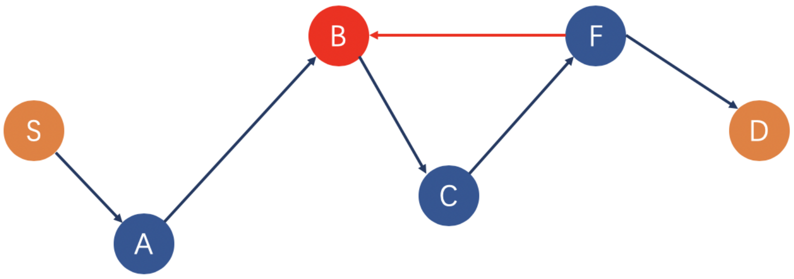

FIGURE 4. Routing loop.

the router can reach its destination. Thus, we can claim that the source node can reach the destination node. Then, we have

$$choice_{x,y} = 1 \Longrightarrow hopCount_{y,S} = hopCount_{x,S} + 1 \quad (9)$$

Here, y is an out neighbor of x.

Then, we can check the reachability of source node S to destination node D by solving the hop count of D.

$$Reachability_{S,D} \Longleftrightarrow hopCount_{D,S} \geq 0 \quad (10)$$

More generally, if we want to check whether a set of routers in δ can reach the destination, we can compute $\bigwedge_{x\in\delta} hopCount_{D,x}$.

B. ISOLATION

Isolation is the negative property of reachability. If a node is isolated from the other node, we can test the reachability between the pair of nodes. In other words, if node A cannot reach node B, node A is isolated from B. Then, the isolation property can be encoded as

$$Isolation_{A,B} \Longleftrightarrow hopCount_{B,A} < 0. \quad (11)$$

In another situation, nodes A and B cannot communicate with each other in direction, and we can check

$$(hopCount_{A,B} < 0) \wedge (hopCount_{B,A} < 0) \quad (12)$$

Similarly, if the configurations need to ensure that two regions are isolated from each other, we can check

$$(\bigwedge_{x\in\delta_1} \bigwedge_{y\in\delta_2} hopCount_{x,y} < 0) \wedge (\bigwedge_{x\in\delta_1} \bigwedge_{y\in\delta_2} hopCount_{y,x} < 0) \quad (13)$$

Here, δ_1 and δ_2 are two sets of routers.

C. ROUTING LOOP

A routing loop is a serious network problem that happens when a data packet is continually routed through the same routers over and over. The data packets continue to be routed within the network in an endless circle. Figure 4 shows a routing loop occurring in a simple network. A routing loop can cause a catastrophic impact on a network and, in some cases, completely disable the network. Due to the loop preventing mechanism of BGP and OSPF, a normal routing loop is a problem associated with distance-vector protocols.

To encode this property, we add one Boolean variable to those specific nodes that are most likely to create a routing loop. The variable *goesThrough* denotes whether a data packet has gone through this node. It is initiated to be *false* at first, $goesThrough = 0$. When a router forwards a packet to its neighbor, the *goesThrough* variable is updated to be *true*, $goesThrough = 1$. If a router forwards to any neighbor with the $goesThrough = 1$, then a routing loop occurs. We can verify whether a routing loop occurs between x and y via the following formula:

$$noLoop_{x,y} = (choice_{x,y} = 1) \wedge (goesThrough_y = 0) \quad (14)$$

To check the routing loop between the source node and destination node, we can also compute the iteration of the above formula.

D. BLACK HOLES

In networking, black holes refer to some routers in the network that can receive packets but silently drop them, without informing the source that the data did not reach its intended destination. We can verify if a node is a black hole by checking its income forwarding and outcome forwarding. Denote the Boolean variable $blackHole_x$ as whether node x is a black hole in the network. Then, we have

$$blackHole_x = (\bigvee_{y\in I} choice_{y,x}) \wedge (\bigwedge_{z\in O} choice_{x,z}) \quad (15)$$

where I is the set of in-neighbors of node x, and O is the set of out-neighbors.

E. WAYPOINTING

Under certain circumstances, operators want to ensure traffic from source node S to destination node D will traverse a certain router of a chain of specific routers. Suppose the current traffic is configured to traverse k routers. As figure 5 shows, routers from R_1 to R_k are the waypointing routers in green, and the general routers are blue nodes. First, we set each router a variable M with $k+1$ dimensions. For a router x, $M_x = (1, m_x^1, m_x^2, \cdots, m_x^k)$. The first bit 1 denotes that the origin of the traffic is S, and m_x^i denotes whether the *ith* special router has been traversed for x. We also initiate these $m_x^i = 0$. For a general router a, if $choices_{S,a} = 1$ and $M_s[0] = 1$, then $M_a[0] = 1$, other variables are still 0.

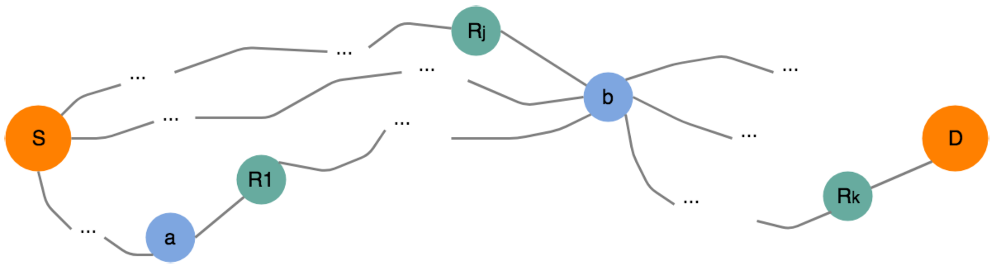

FIGURE 5. Waypointing.

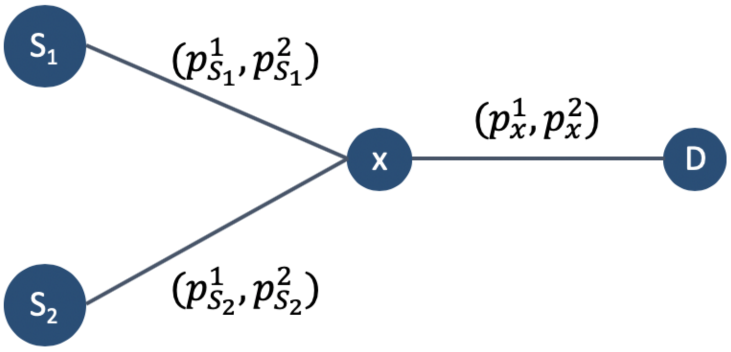

FIGURE 6. Disjoint path.

For a special router (one of the waypoints), if $choice_{a,R_1} = 1$ and $M_a[0] = 1$, then $M_{R_1}[0] = 1$. Suppose R_1 is the first node of waypoints; then, $m^1_{R_1} = 1$. Thus, $M_{R_1} = (1, 1, 0, \cdots, 0)$. Then, we can claim,

$$\begin{aligned}&(\bigvee_{R\in I} choice_{R,x}) \wedge (M_R[i-1] = 1) \wedge (M_R[0] = 1)\\ &\implies M_x[i] = 1 \; if \; x \; is \; special\\ &\implies M_x[i-1] = 1 \; if \; x \; is \; general \end{aligned} \tag{16}$$

where I is the in-neighbor set of node x.

Another way to verify waypointing is to check the reachability between S and D by shutting down these waypoints. To model the network more exhaustively, we also need to encode the state of the physical links and node in the scope of network topology. Therefore, we use $state_x$ to represent whether node x is active and $state_{x,y}$ to encode the state of the link between x and y. If $state_x = 0$, then node x is shut down. Thus, we can set $state_w = 0$ and $\exists \; w \in \{waypoints\}$ and then check $reachability_{S,D}$.

F. DISJOINT PATH

The problem of disjoint paths in the network has been researched for years, and there are many clever algorithms to solve the problem. However, encoding such a property in a well-defined network model is still challenging. There are two subproblems of this property, the node-disjoint path and the link-disjoint path. We just take the link-disjoint path to illustrate the way we encode. Take figure 6 as an example. We want to verify that sources S_1 and S_2 traverse the same link (x, D) to destination D. We add a two-dimensional variable p to each link, $p = (p_1, p_2)$, and each p_i is a Boolean variable, $i = 1, 2$. p_1 represents whether S_1 uses this link to forward traffic, and p_2 denotes whether S_2 forwards traffic via this link. A link with a variable $p = (1, 1)$ is used by two routes. We can compute the variable p via the forwarding behavior,

$$\begin{aligned}&choice_{S_1,x} = 1 \Rightarrow p^1_{s_1} = 1\\ &(choice_{x,D} = 1) \wedge (p^1_{s_1} = 1) \Rightarrow p^1_x = 1\\ &choice_{S_2,x} = 1 \Rightarrow p^2_{s_2} = 1\\ &(choice_{x,D} = 1) \wedge (p^2_{s_2} = 1) \Rightarrow p^2_x = 1\end{aligned} \tag{17}$$

G. PATH LENGTH

In some configurations, the operators may need to ensure that the packets traverse bounded-length paths. For instance, source S is configured to reach destination D, and the path length is not more than k hops. Otherwise, the traffic will occupy too much bandwidth of the network. Then, we can make use of the variable *hopCount* aforementioned to encode this property. If a source node S hops n counts to reach the destination node x, then the length of the path is n. We can check $hopCount_{x,S} \leq k$ to ensure that the path is bounded. Some routers are configured to have an equal length to reach the destination. Violations can be found if these routers have different hop counts to reach the destinations.

H. MULTIPATH EQUIVALENCE

The routers configured ECMP (equal cost multipath) may have several paths to the destination. Multipath consistency

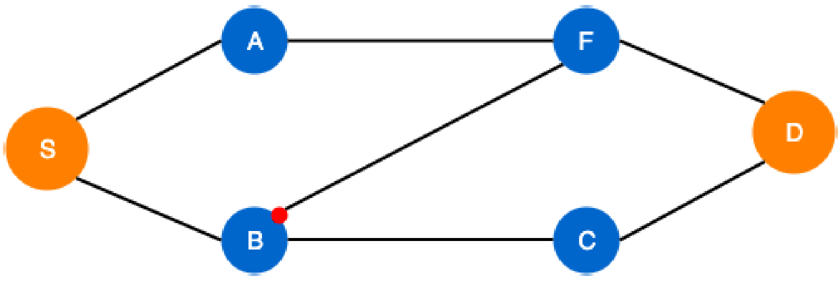

FIGURE 7. **Multipath equivalence.**

ensures that these paths have equal statuses. For example, in figure 7, source S has three choices to reach destination D. If D is reachable for S, then all three paths are valid. If one port of B fails, the path $S-B-F-D$ is invalid. Multipath consistency ensures that the other two paths are also invalid. Then, we can make use of the reachability in an iterative way for each node.

$$\begin{aligned} reachability_{S,D} &\Longrightarrow (\bigwedge_{R\in\{A,B\}} choice_{S,R} \\ &\quad \wedge reachability_{R,D}) \\ reachability_{A,D} &\Longrightarrow (choice_{A,F}) \\ &\quad \wedge (reachability_{F,D}) \\ reachability_{B,D} &\Longrightarrow (\bigwedge_{R\in\{C,F\}} choice_{B,R} \\ &\quad \wedge reachability_{R,D}) \\ &\quad \cdots \end{aligned} \tag{18}$$

Here, the variable *reachability* is defined as equation (10) shows.

Through the above formulas, B can reach D via C but not F, which violates the multipath equivalence property.

I. FAULT INVARIANCE

It is inevitable that links or ports in the network will fail for some reason. Fault invariance maintains that the network still runs as expected when one or more nodes or links fail. Then, we can add a new variable *fail* to each node and link. If $state_x$ of router x is *false*, then $fail_x = 1$. $fail_{(x,y)} = 1$ if $status_{(x,y)}$ of link (x, y) is *false*. Thus, we can encode the property to

$$\sum_{x\in\mathcal{V}} fail_x \leq k, \tag{19}$$

and

$$\sum_{(x,y)\in\mathcal{E}} fail_{(x,y)} \leq k. \tag{20}$$

The equation above means that the network still runs as expected when there are k nodes or links that fail.

J. LOAD BALANCE

Some routers are usually configured ECMP to forward traffic through multiple paths to properly make use of the bandwidth

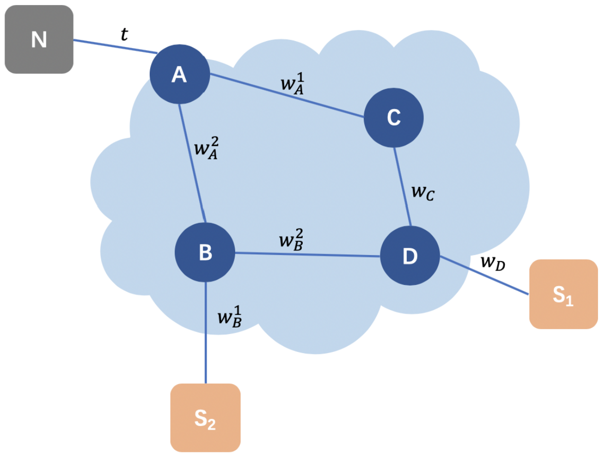

FIGURE 8. **Load balance.**

of the link. Consider the example in figure 8; routers are all configured ECMP. In figure 8, router A receives traffic from N and forwards to B and C separately. Then, B forwards to D and the subnet $S1$. C forwards to D and D forwards to the subnet $S2$. We add a variable to each link. First, there is a total traffic variable t on the link from N to A. We can initiate $t = t_0$ as a whole input according to the configuration. Then, A receives t traffic and forwards w_A^1 to B and w_A^2 to C. Thus, we have $w_A^1 + w_A^2 = t$. Similarly, we can encode other links,

$$\begin{aligned} t &= t_0 \\ w_A^1 + w_A^2 &= t \\ w_B^1 + w_B^2 &= w_A^2 \\ w_C &= w_A^1 \\ w_D &= w_B^2 + w_C \end{aligned} \tag{21}$$

where t_0 is a constance.

These variables depend on the forwarding behavior, similar to the work [7] described. Take router A as an example,

$$\begin{aligned} choice_{A,C} = 1 &\Rightarrow w_A^1 = a_1 \\ choice_{A,C} = 0 &\Rightarrow w_A^1 = 0 \end{aligned} \tag{22}$$

where a_1 is the traffic from A to C.

We can verify whether the links (A, C) and (A, B) average the traffic from N by checking $|w_A^1 - w_A^2| \leq h$, and h is the constant threshold that we set.

V. CONCLUSION AND DISCUSSION

In conclusion, our main idea is to abstract a network more practically, and combining the routing records into the algebraic structure is a way of integrating multiple routes into the model. We first redefine the definition of the graph model of the network. A node in the graph splits into k nodes, where k is the number of protocols that it runs. Thus, the information of the routes advertised in the network is clear to catch. Then, we advocate a modified routing algebra $(\mathcal{G}, \Sigma, \preceq, \mathcal{R}, \mathcal{F})$ by abstracting the routing records and filtering mechanism to $\mathcal{R}$ and $\mathcal{F}$. In this way, we can concisely and accurately analyze and model the interactions of protocols and forwarding behaviors. Based on the modified routing algebra, we instantiate each element of the algebraic model. For instance, we use different types of variables to encode the nodes and edges in the network graph and the attributes of routing protocols. These variables compose logical formulas to encode the network behaviors, such as forwarding. Then, the filtering function $\mathcal{F}$ implements the functions of routing policies to block certain routes or modify the attributes of routing records. The other important improvement of our work is to make use of the constraints and formulas to abstract all the common kinds of network properties into logical representations and obtain a complete abstract model of both network behaviors and the properties in a mathematical way. Then, it is possible to program the whole network model and verify different properties of interest for computer engineers. The main contribution of our work is to model the real network into an algebraic structure and encode the network behaviors and properties in theoretically mathematical logical formulas. Coding the whole network model into a program and using real data for implementation are complicated and challenging. As further research, we that believe our contributions can be programmed and encoded to an available verification tool.

As is known, the mechanism of routing protocols is quite complicated. There are also some configuration techniques we have not considered, such as route maps and route reflectors in internal BGPs. Thus, determining how to model these features into the routing algebra and encode properly is still worth researching. On the other hand, applying the routing algebra to check the correctness of the configuration files is also important but a hard problem. Encoding the algebraic structure into logical formulas is an effective method to compute and code. As mentioned above, the logical formulas that we encode for nodes and edges are all SMT constraints. However, as the scale of the network increases, the number of SMT constraints will increase exponentially. Then, the time of verification programming may be out of control. Though some tools aim to improve the efficiency of verifying the reachability of the network by compressing the scale of the large network [19] or abstracting the control plane in a systematic form [15], they are obviously at the cost of losing the possibility of verifying QoS network properties. Therefore, it is still a significant and challenging issue to model and encode the network concisely and concretely. The algebraic model is applied not only in IP network verification but also in other related research areas. In the work of [21], the abstraction network representations are used to solve the space network, such as verification of the periodically changed topology of satellites in space. We believe that it is valuable to explore more applications of network verification in other research areas.

REFERENCES

[1] P. J. Taylor and T. G. Griffin, "A model of configuration languages for routing protocols," in *Proc. 2nd ACM SIGCOMM Workshop Program. Routers Extensible Services Tomorrow*, 2009, pp. 55–60.

[2] P. Kazemian, M. Chang, H. Zeng, G. Varghese, N. McKeown, and S. Whyte, "Real time network policy checking using header space analysis," in *Proc. 10th USENIX Symp. Netw. Syst. Design Implement.*, 2013, pp. 99–111.

[3] A. Khurshid, X. Zou, W. Zhou, M. Caesar, and P. B. Godfrey, "VeriFlow: Verifying network-wide invariants in real time," in *Proc. 10th USENIX Symp. Netw. Syst. Design Implement.*, 2013, pp. 15–27.

[4] J. T. Moy, *OSPF: Anatomy of an Internet Routing Protocol*. Reading, MA, USA: Addison-Wesley, 1998.

[5] C. Hendrick, *Routing Information Protocol*, document RFC 1058, 1988, pp. 1–33. [Online]. Available: http://tools.ietf.org/rfc/rfc1058.txt

[6] Y. Rekhter and T. Li, *A Border Gateway Protocol 4 (BGP-4)*, document RFC 1771, 1995.

[7] R. Beckett, A. Gupta, R. Mahajan, and D. Walker, "A general approach to network configuration verification," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Aug. 2017, pp. 155–168.

[8] L. De Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Proc. Int. Conf. Tools Algorithms Construct. Anal. Syst.* Berlin, Germany: Springer, 2008, pp. 337–340.

[9] C. Alaettinoglu, C. Villamizar, E. Gerich, D. Kessens, D. M. Meyer, T. Bates, D. Karrenberg, and M. Terpstra, *Routing Policy Specification Language (RPSL)*, document RFC 2622, 1999, pp. 1–69.

[10] T. G. Griffin and J. L. Sobrinho, "Metarouting," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 1–12, Oct. 2005.

[11] T. G. Griffin, F. B. Shepherd, and G. Wilfong, "The stable paths problem and interdomain routing," *IEEE/ACM Trans. Netw.*, vol. 10, no. 2, pp. 232–243, Apr. 2002.

[12] A. J. T. Gurney and T. G. Griffin, "Neighbor-specific BGP: An algebraic exploration," in *Proc. 18th IEEE Int. Conf. Netw. Protocols*, Oct. 2010, pp. 103–112.

[13] J. L. Sobrinho, "An algebraic theory of dynamic network routing," *IEEE/ACM Trans. Netw.*, vol. 13, no. 5, pp. 1160–1173, Oct. 2005.

[14] J. N. Billings and T. G. Griffin, "A model of Internet routing using semi-modules," in *Proc. Int. Conf. Relational Methods Comput. Sci.* Berlin, Germany: Springer, 2009, pp. 29–43.

[15] R. Beckett, A. Gupta, R. Mahajan, and D. Walker, "Abstract interpretation of distributed network control planes," *Proc. ACM Program. Lang.*, vol. 4, pp. 1–27, Jan. 2020.

[16] A. Gember-Jacobson, R. Viswanathan, A. Akella, and R. Mahajan, "Fast control plane analysis using an abstract representation," in *Proc. Conf. ACM SIGCOMM Conf.*, 2016, pp. 300–313.

[17] M. Gondran and M. Minoux, *Graph, Dioids and Semirings: New Models and Algorithms*, vol. 41. Springer, 2008.

[18] J. L. Sobrinho, "Algebra and algorithms for QoS path computation and hop-by-hop routing in the Internet," in *Proc. IEEE INFOCOM, Conf. Comput. Commun., 20th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, vol. 2, Apr. 2001, pp. 727–735.

[19] N. Giannarakis, R. Beckett, R. Mahajan, and D. Walker, "Efficient verification of network fault tolerance via counterexample-guided refinement," in *Proc. Int. Conf. Comput. Aided Verification*. Cham, Switzerland: Springer, 2019, pp. 305–323.

[20] R. Beckett, A. Gupta, R. Mahajan, and D. Walker, "Control plane compression," in *Proc. Conf. ACM Special Interest Group Data Commun.*, 2018, pp. 476–489.

[21] F. Ye, Y. Li, Z. Wang, J. Yao, Y. Chen, T. Lan, H. Li, X. Shi, and X. Yin, "Network verification on periodically changed topology," in *Proc. 3rd Asia–Pacific Workshop Netw. (APNet)*, 2019.

YI PAN received the B.S. degree in mathematics from the Sichuan Normal University of Chengdu, in 2014, and the M.S. degree in theoretical mathematics from the Henan University of Kaifeng, Henan, in 2018. She is currently pursuing the Ph.D. degree in operational research and cybernetics with the University of Chinese Academy of Sciences. She is mainly engaged in network optimization and network verification.

SUIXIANG GAO received the Ph.D. degree in mathematics from the Institute of Applied Mathematics, Chinese Academy of Sciences, China, in 1998.

He is currently a Professor with the School of Mathematics, University of Chinese Academy of Sciences, China. He has hosted and participated in more than 20 national scientific research projects and horizontal research projects, including five key projects of the National Natural Science Foundation of China and five general projects, one subproject of the National 973 Project, and one major subproject of the National 863 Program. He has published more than 90 academic articles in core journals at home and abroad. His research interests include optimization theory and algorithms, communication network optimization, graph theory, and network flows.

WENGUO YANG received the M.A. degree in operation research and control theory from Beijing Jiaotong University, China, in 2003, and the Ph.D. degree in operation research and control theory from the University of Chinese Academy of Sciences, China, in 2006. He is currently a Professor with the School of Mathematics, University of Chinese Academy of Sciences. He has presided over two national science foundation programs and several practical applications. He has published more than 60 research articles in reputed refereed academic journals and conference proceedings, such as *Computers & Operations Research*, the *Journal of Combinatorial Optimization*, *Optimization*, IEEE Access, and the *International Journal of Computer Mathematics*. His research interests include social networks, robust optimization, nonlinear combinatorial optimization, and network optimization.