Received 1 May 2024; revised 10 June 2024; accepted 1 July 2024. Date of publication 10 July 2024; date of current version 30 July 2024.

Digital Object Identifier 10.1109/OJCOMS.2024.3425831

Application of Category Theory to Network Service Fault Detection

PEDRO MARTINEZ-JULIA (Member, IEEE), VED P. KAFLE (Senior Member, IEEE), AND HITOSHI ASAEDA (Senior Member, IEEE)

Network Architecture Laboratory, Network Research Institute, National Institute of Information and Communications Technology, Tokyo 184-8795, Japan

CORRESPONDING AUTHOR: P. MARTINEZ-JULIA (e-mail: pedro@nict.go.jp)

ABSTRACT Network automation has become crucial in supporting services in 6G networks. This mainly derives from the complexity of the composition of numerous distributed virtual network functions (VNFs) in creating highly flexible virtual network services. Therefore, a network service automation system is a key technology enabler for 6G. However, the added complexity renders network service automation systems particularly sensitive to faults, some of which cause network outages that harm the smooth operation of basic societal services. Current state-of-the-art (SotA) solutions for fault detection can barely detect hidden faults. Herein, we propose a mechanism for automated network service analysis (ANSA), which constructs and analyzes a digital twin of a network service. The digital twin represents the available information about the network service based on category theory. It uses the properties of category theory to perform an analysis through which the faults of the network service are identified. We evaluate a prototype of a network service automation system that incorporates ANSA to demonstrate 1) the benefits of using digital twins for analyzing network services, 2) the benefits of using category theory for constructing digital twins of the network services, and 3) the resulting improvements in fault detection. Overall, ANSA can detect an average of 94% of the faults present in a network service. In comparison, previous SotA solutions can detect only 30%–50% of all faults.

INDEX TERMS Network automation, fault detection, category theory.

I. INTRODUCTION

SOCIETAL improvements are tightly interlinked with technological advances. To achieve these improvements, new applications and services are required that rely on complex and dynamic interactions between each other and that are based on collaborative and distributed schemes. New network services are required to realize these interactions. These services include those promised by 6G and related network technologies, which result in constantly increasing management complexity, thus rendering network automation essential.

Many initiatives have been proposed to provide key technology enablers for 6G and network automation. A key initiative is promoted by the European Telecommunications Standards Institute's zero-touch network service creation and management industry specification group (ETSI ISG ZSM) [1]. This initiative conceives that high-level network automation can be realized on top of the widespread network virtualization and softwarization paradigms [2], [3]. Of these, network function virtualization (NFV) [4] and software-defined networking (SDN) [5] have been proposed. Together, these support the creation of highly flexible network services, including those that involve multiple sites or domains, thereby achieving a high degree of resilience and optimization.

Automating network services based on NFV and SDN requires a network service automation system that implements ZSM specifications and ETSI's NFV management and orchestration (ETSI's NFV-MANO) [6] framework specifications. A reference implementation of ETSI's NFV-MANO is open-source MANO (OSM) [7]. It supports managing network services based on approaches such as NFV and SDN, which are instantiated on typical underlying infrastructures based in OpenStack [8] and the OpenFlow protocol [9]. To automate OSM operations, it has been extended using the autonomic resource control architecture (ARCA) [10]. ARCA uses artificial intelligence (AI) and machine learning (ML) algorithms to analyze the state

of a network service and determine whether some reactive or preventive change is required.

Network outages represent major threats to the smooth operation of basic societal services. Therefore, they must be prevented when possible or promptly resolved when they occur. Major network outages occur owing to hidden faults, which cannot be detected by analyzing common metrics and variables. Hidden faults differ from exposed faults, which can be detected by analyzing common metrics and variables. Avoiding major network outages requires detection of actual and potential exposed and hidden faults. The effects of faults are worse in network automation contexts because automated systems consider only the events and actions included in their code. Therefore, network service automation systems must detect faults present in running or planned network services, exploiting the large amount of monitoring information and configuration information that they can obtain from network services.

We examined fault detection mechanisms used in state-of-the-art (SotA) network service automation systems as well as general SotA solutions for fault detection [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25]. In general, we found that they do not particularly consider hidden faults in their processes and do not represent qualitative information in the system models they analyze. We aim to resolve these shortcomings in the present study.

Network service automation systems must be enabled using hidden fault detection, where a formal definition of hidden faults would be highly beneficial for these systems. We propose to address the aforementioned issues through a newly developed mechanism called automated network service analysis (ANSA). ANSA collects and analyzes all available information about a network service to construct a network digital twin (NDT) that represents the network service. ANSA follows the constraints of category theory [26] to build the NDT and utilizes category theory properties to perform the analysis. This ensures that the computation yields the desired results, i.e., ANSA can ensure that the network service state is valid or that faults have been detected. Application of category theory enables ANSA to produce computation results that are verifiable via proofs. With ANSA, SotA network service automation systems can detect hidden faults, which represents a considerable improvement over previous SotA network service automation systems.

The major contributions of this study are as follows. First, the formal definition of the NDT is provided using category theory along with the definitions required for populating and typing the NDT objects. Thus, the NDT we propose is an evolutionary mathematical structure armed with constraints and computational morphisms (i.e., functions) that ensure its consistency. The NDT enhances the application of AI methods by providing a consistent and formal structure as well as constraints that can be studied and/or used in simulations based on AI methods. Second, the study formally defines a knowledge object (KO) as a set of knowledge items. This set is used by 1) management solutions to provide configuration information about network services, 2) monitoring sources to provide information about network services, and 3) ANSA to build an external representation of the NDT as a large KO.

Third, the study formulates new theorems for characterizing and identifying hidden faults by analyzing the configuration and state information of a network service structured in a knowledge graph. The hidden fault raising theorem relates exposed and hidden faults and relies on the application of the Yoneda lemma [27], which is a key aspect of category theory. The strong emergent typing theorem allows for implicitly obtaining strong relations between KOs and types, thereby avoiding the need to explicitly declare the type of each KO. The fault qualification theorem refers to the definitions of particular monitoring and data collection strategies that maximize the probability of fault detection. The fault polymorphism theorem states that under the influence of faults, KOs expose multiple types simultaneously. This theorem is used to find a set of elements that must be particularly observed in compliance with the fault qualification theorem because they would most likely be related to a future fault.

Finally, the key contribution of this study is the exploitation of the previous definitions and formulations for formally defining the algorithm that ANSA uses for fault detection. It tackles the huge space of network service states, which cannot be fully investigated using SotA techniques for fault detection, by selecting states with some type of fault and extending them using the aforementioned theorems. Therefore, ANSA can explore network service states not considered by SotA solutions, particularly those related to hidden faults.

In particular, ANSA resolves the aforementioned issues as follows. On the one hand, the hidden fault raising and fault qualification theorems provide two views to identify hidden faults. On the other hand, application of the hidden fault raising and strong emergent typing theorems enables the identification of multiaspect faults. The benefits of these theorems are delivered through the construction of a knowledge graph governed by semantics and with it an NDT that permits simultaneously analyzing qualitative and quantitative data as well as conveying multiparameter simulations. In addition, through semantic techniques combined with the application of the aforementioned theorems, the generation of simulation parameter values is guided so that parameter values are maintained within plausible boundaries.

We implemented a proof-of-concept of ANSA within a network service automation system and evaluated it to demonstrate that ANSA can detect exposed and hidden faults and show the cases in which it performs better than SotA techniques in fault detection. We conducted our experiments on a real deployment based on OpenStack [8] and Open-Source MANO (OSM) [7]. Our results confirmed that the fault detection ratio obtained by ANSA is considerably higher than that obtained by previous SotA solutions. By

comparing the number of knowledge items associated with faults introduced in the experiment and the number identified by ANSA and other SotA solutions, we found that ANSA could detect an average of 94% of the faults present in a network service, whereas previous SotA solutions detected only 30%–50% of the faults.

The remainder of this paper is organized as follows. In Section II, we present the background for this study, introduce related studies, and formulate the target problem. In Section III, we introduce our theorems and ANSA as our solution to the target problem, while in Section IV, we describe the network service automation system into which we integrate ANSA for evaluation. In Section V, we discuss our validation of ANSA through a real implementation to obtain experimental results. Finally, in Section VI, we provide our concluding remarks and present directions for future research.

II. BACKGROUND

In this section, we describe studies related to our proposal and prior knowledge we use to construct our solution.

Fig. 1 shows the environment and context of the present study as well as the position of the ANSA component, which is detailed in Section III. It depicts a typical network service comprising several virtual network functions (VNFs) interconnected through virtual links. The network service is embedded in a multidomain underlying infrastructure, which is constructed from physical or logical platforms deployed on multiple sites interconnected through a virtual private network (VPN). The platforms are usually managed by virtual infrastructure managers (VIMs), such as OpenStack and SDN controllers, which can provide a large amount of monitoring data (i.e., telemetry) and configuration information related to the overlying network services. This information is essential for detecting exposed and hidden faults.

A. RELATED STUDIES

1) FAULT DETECTION

We next discuss the most outstanding proposals for fault detection identified in the literature.

The work in [11] examined the application of adversarial inference, which relies on analyzing states stochastically selected based on a particular random distribution and evaluated by a sequential Monte Carlo algorithm. It has relatively high potential for fault detection owing to its stochastic nature. However, it does not consider qualitative states and does not focus on arbitrary or hidden faults.

A similar stochastic sampling of posterior states of a system (in this case applied to a robot), is discussed in [12]. It proposes to obtain the samples from a provided transition kernel and transition probabilities; therefore, the evolution of the simulation was somewhat guided. This proposal also lacked support for qualitative aspects and arbitrary or hidden faults.

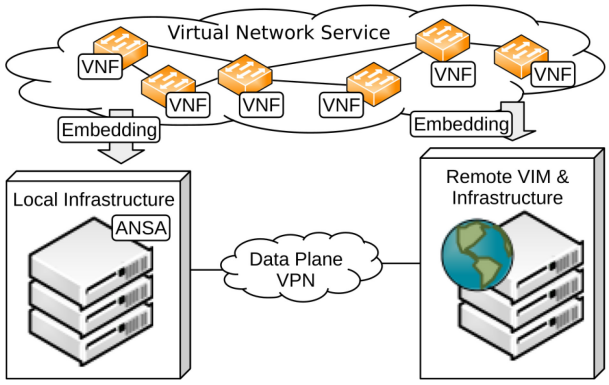


FIGURE 1. Target context: network service embedded over a multidomain infrastructure consisting of two separate sites.

Fault detection has been also studied in the context of vehicular systems, whose network structure has similarities to our target. Particularly, researchers in [13] propose to use an adaptive importance-sampling method to evaluate the probability of event occurrence. This allowed its algorithm to project certain behaviors, a feature that is particularly useful in case of vehicular systems, and incorporate learning using imitation techniques.

There are many works that propose to use ML algorithms to detect faults. A survey of ML algorithms used to detect faults in wireless sensor networks is presented in [14]. Although the paper showed that several ML methods can detect exposed faults, these methods function poorly in case of hidden faults owing to the limited amount of learning data regarding hidden faults available for building learning models, which are essential for ML methods.

Aiming the improvement of the application of ML methods to fault detection, researchers in [15] proposed to transform the fault detection problem into a classification problem using ML to learn the classes linked to faults and later detect them in new data. Although the proposal improves fault detection, it still has many drawbacks and is comparable only with other ML-based solutions. On the other hand, researchers in [28] targeted fault detection for the Internet of Things (IoT), with particular focus on industrial scenarios. Similar to the proposal in [15], this study proposed using an algorithm based on neural networks to define the problem as a classification problem. This solution still lacks a suitable definition of faults for modeling neural networks. Therefore, it is comparable only to other ML-based solutions and has the same drawbacks.

From a different perspective, the work in [16] investigated the application of trust evaluation for fault detection in wireless sensor networks and built a trust-based formal model to detect faults without requiring a simulation. Although this method is suitable for detecting exposed faults, it is unsuitable for use in detecting hidden faults because the formal model defines the fault shape individually. This is the case even though hidden faults cannot be defined by themselves alone; otherwise, they would be exposed faults.

In the same line, researchers in [17] proposed a distributed method for fault detection in wireless sensor networks that exploits the knowledge provided by node neighborhoods to determine if a node is faulty. The extrapolation of information from a certain point to identify the state of another point is interesting. However, it is considerably limited with respect to hidden fault detection. Nevertheless, the processing and transmission overhead is substantially reduced because the application of extrapolation, inference, and in-place computation reduces the amount of data that must be transmitted.

Researchers in [18] targeted 5G systems, which are particularly based on virtualization, and proposed that virtual elements can collaborate to construct a model of their own network system. Although the solution tackles the inability of underlying control and data elements to know the particularities of the virtual system, its application to fault detection is limited to the faults supported by the model.

Studying fault detection in wireless sensor networks at the node level, the work in [19], proposed classifying vertices and considering faulty vertices from the outset. This approach has the benefit of incorporating the notion of faults into the normal lifecycle of the network elements such that all elements have some type of faulty state linked even though it can be zero-fault. The approach is used to distribute jobs among network vertices using vertices that are faulty but not completely broken for the tasks they are able to perform, such as fault detection. Although this technique is considerably lightweight and has potential for detecting faults, it cannot perform deep analysis on faults with all available information when searching for more complex exposed and hidden faults.

Fault detection also has been approached by applying ML together with time-series analysis. Researchers in [20] proposed the application of long- and short-term time-series analyses, which are suitable for finding outliers from systems with marginal training. Although this mechanism is based on ML, it can detect some faults that other methods cannot. However, this method does not provide either the required accuracy or reliability because the model it creates does not have a sufficiently complete view of the target system, i.e., it focuses on single values and cannot tackle systems of complex values and their relations.

The application of blockchain mechanisms to fault detection has also been studied. The work in [21] proposed to manage data obtained from IoT scenarios in a blockchain, which is used by random forest algorithms based on particle swarm optimization to determine faults in a network. The outstanding feature of this solution is the use of blockchain to safely decouple the objects of study, namely, the vertices of the IoT network, from the elements that execute the somewhat power-hungry analytical methods (i.e., edge computing vertices). However, the algorithms used to perform the analysis are unsuitable for detecting hidden faults despite the fact that using particle swarm optimization endows the algorithms with the ability to detect some faults not detected by other algorithms based on neural networks.

In line with classification methods for fault detection, researchers in [22] proposed using Kalman filters to define fault patterns and build a classifier. Although the method is extremely fast compared with other algorithms, its ability to detect arbitrary and hidden faults is questionable because it is completely bound to the patterns and cannot infer beyond them (e.g., via simulation and/or regression), which is required for detecting hidden faults.

The work in [23] targeted fault detection in wireless sensor networks using a deep learning mechanism that analyzes the variables associated with the soft state of nodes and using a three-tier mechanism to detect hardware faults. Although this method is outstanding in terms of its low rate of false positives and low energy consumption, its construction binds it to the detection of faults after they have already affected the operations of the network. Moreover, the method does not detect hidden faults and cannot be modified to do so without a drastic change in its functions.

Some works studied fault detection through the application of ML to events occurring outside a network that affect the operation of a network [10], [24]. Although considering external events is somewhat unique, its plain use of ML, which does not focus on system modeling, makes those works unsuitable for detecting complex or hidden faults.

Another recent trend is the application of in-band network telemetry for network management tasks. Researchers in [25] targeted fault detection using in-band network telemetry to retrieve data and metadata and a generative adversarial network to recover lost telemetry. Although in terms of its accuracy, the use of in-band network telemetry is promising, the proposal is focused on detecting faults in the form of packet losses, thereby ignoring other faults, including hidden faults.

Overall, many techniques have been proposed that target wireless sensor networks because these networks are sensitive to small faults. However, as summarized in Table 1, these proposed techniques do not consider the existence and detection of hidden faults. Moreover, nearly every fault detection method in the related studies relies on the quantification of system states as direct representations of performance metrics and state parameters, and they lack support for non-quantifiable items or item relations (e.g., links connected to switches or encryption algorithm types supported by server software) to promote qualitative system modeling.

Furthermore, because the actions they take during each step of the analysis are selected randomly, fault paths will arise that will not be analyzed. These can include paths that are later taken and result in undetected system faults. Only a few solutions have considered a type of guided state exploration, although they also are deficient in other respects, such as supporting arbitrary faults. Finally, the constraints and cost functions handled by the aforementioned studies are overly coupled with particular features of the systems they analyze. A different set of definitions is required for each type of what-if exploration, making these proposed

TABLE 1. Summary of related studies.

Proposal	Key Method	Qualitative System Modeling	Random State Study	Guided State Study	Exposed Fault Scrutiny	Arbitrary Fault Scrutiny	Hidden Fault Scrutiny	Efficient Data, CPU Usage
[11]	Monte Carlo	✗	✓	✗	✓	✗	✗	✗
[12]	Transition kernel	✗	✓	✓	✓	✗	✗	✗
[13]	Learning by imitation	✗	✓	✗	✓	✓	✗	✗
[14]	Machine learning (ML)	✗	✗	✗	✓	✗	✗	✗
[15]	Detection → ML classification	✗	✗	✗	✓	✗	✗	✗
[28]	Det. → Neural net. classif.	✗	✗	✗	✓	✓	✗	✗
[16]	Trust-based formal model	✗	✗	✓	✓	✗	✗	✗
[17]	Distributed extrapolation	✗	✗	✗	✓	✗	✗	✓
[18]	Collab. model constr.	✗	✗	✗	✓	✗	✗	✓
[19]	Node notification	✗	✗	✗	✓	✗	✗	✓
[20]	Time-series analysis, ML	✗	✓	✗	✓	✓	✗	✗
[21]	Blockchain, rnd. forest	✗	✓	✗	✗	✓	✗	✗
[22]	Kalman filter pattern	✗	✓	✗	✓	✗	✗	✓
[23]	Deep learning	✗	✗	✗	✓	✗	✗	✓
[24]	External events, ML	✗	✗	✗	✗	✓	✗	✗
[25]	In-band network telem.	✗	✗	✗	✓	✗	✗	✗

techniques unfeasible for application to the detection of faults in systems with many variables, such as network service automation systems.

2) APPLIED DIGITAL TWINS

A digital twin is a digital representation of a system intended to be highly synchronized for investigating the behavior of the system in certain scenarios without altering the system itself. It accomplishes this by applying different mechanisms, such as running simulations to answer what-if questions. Thus, an NDT is a digital representation of a network that is used as a type of regression structure that can carry over several parameters at a time (i.e., simulate what-if scenarios) and provide a concise view of the analyzed network service. Digital twins have been proposed in previous research for fault detection [29]. Although the nature and final purpose of these solutions are considerably different from ours, their results set a baseline for the accuracy to be obtained through a comparison of real measurements with simulated predictions.

As shown in [30], NDTs are effectively used to improve predictions of some network performance metrics, such as traffic, by performing analysis using multiple data; hence, a proper structure as with that of an NDT is essential. The application of ML in predicting some valuates of NDTs has been studied earlier [31]. We incorporate similar predictions in our NDT, particularly in step 5 of our algorithm.

The work in [32] constructs a quality-of-transmission model to represent a set of devices and their traffic, the parameters for which are tuned for the model output to fit the data. Although very simple, this method resembles the construction of a digital twin and is a step forward from the simple application of ML, which is partly affected by the modeled system. However, this digital twin (i.e., quality of transmission) cannot be applied to higher-order network systems with multiple elements, parameters to study, etc.

The work in [33] proposed to build a digital twin for assessing the proper operation of optical networks. Although it does not target fault detection specifically, its abilities to accurately model a network and forecast future states are powerful for the implementation of fault detection. However, because it lacks this implementation, it is unable to detect hidden faults.

3) CATEGORY THEORY

A key aspect of our proposal is the application of category theory to the construction of the NDT. Category theory is a branch of mathematics that provides mechanisms for describing, formulating, and formalizing mathematical structures using particular objects and relations between them [26], [34]. The fundamental concepts of category theory enable the structures to be analyzed through these relations without having to assess the objects.

By definition [34], a category $\mathcal{C}$ has the following constituents:

- A collection $\mathbf{Ob}(\mathcal{C})$ of objects that are arbitrary elements such as sets or semantic atoms.
- For every two objects $c, d \in \mathbf{Ob}(\mathcal{C})$, a set $\mathcal{C}(c, d)$ of morphisms from c to d exists that are also defined as $f : c \to d$, which are relations, maps, etc. from c to d.
- For every object $c \in \mathbf{Ob}(\mathcal{C})$, we have an identity morphism on c defined as $\mathbf{id}_c \in \mathcal{C}(c, c)$.
- For every three objects $c, d, e \in \mathbf{Ob}(\mathcal{C})$ as well as morphisms $f \in \mathcal{C}(c, d)$ and $g \in \mathcal{C}(d, e)$, we have the morphism $g \circ f \in \mathcal{C}(c, e)$, which is a composite of f and g.

Furthermore, the aforementioned constituents must satisfy the following conditions:

- Unitality: Composing any morphism $f : c \to d$ with the identity on c from the left or the identity on d from the right does nothing. Thus, $f \circ \mathbf{id}_c = f$ and $\mathbf{id}_d \circ f = f$.
- Associativity: For every three morphisms $f : a \to b$, $g : b \to c$, and $h : c \to d$, we have that $(h \circ g) \circ f = h \circ (g \circ f)$.

Another important concept of category theory is the functor. A functor is a mapping between categories, wherein objects are sent to objects and morphisms to morphisms. Formally, considering categories $\mathcal{C}$ and $\mathcal{D}$, the functor from $\mathcal{C}$ to $\mathcal{D}$ is denoted as $F : \mathcal{C} \to \mathcal{D}$ and has the following conditions:

- For every object $c \in \mathbf{Ob}(\mathcal{C})$, we have the object $F(c) \in \mathbf{Ob}(\mathcal{D})$.
- For every morphism $f : c \to d$ in $\mathcal{C}$, we have the morphism $F(f) : F(c) \to F(d)$ in $\mathcal{D}$.

Similar to the aforementioned constituents, the functor constituents must satisfy the following conditions:

- For every object $c \in \mathbf{Ob}(\mathcal{C})$, we have $F(\mathbf{id}_c) = \mathbf{id}_{F(c)}$.
- For every three objects $c, d, e \in \mathbf{Ob}(\mathcal{C})$ as well as morphisms $f \in \mathcal{C}(c, d)$ and $g \in \mathcal{C}(d, e)$, we have that $F(g \circ f) = F(g) \circ F(f)$ holds in $\mathcal{D}$.

A concept of category theory that is particularly important for our study is the monad. A monad on the category $\mathcal{C}$ is an endofunctor, a functor from a category to itself, and is defined as $T : \mathcal{C} \to \mathcal{C}$. It is accompanied by two natural transformations that are morphisms on the category of functors that map one functor to another. The natural transformations are as follows:

- First, we have $\eta : 1_{\mathcal{C}} \to T$, where $1_{\mathcal{C}}$ is the identity functor on $\mathcal{C}$.
- Second, we have $\mu : T^2 \to T$, where T^2 is the functor $T \circ T : \mathcal{C} \to \mathcal{C}$.

The following conditions must hold for a monad to be correctly defined:

- First, it must hold that $\mu \circ T\mu = \mu \circ \mu T$, where $T\mu$ and μT are obtained using horizontal composition [26]. Here, both sides are $T^3 \to T$.
- Second, it must hold that $\mu \circ T\eta = \mu \circ \eta T = 1_T$, where 1_T is the identity $T \to T$, and the other expressions are also from T to T.

In summary, the simple but powerful concepts defined by category theory allow reasoning on considerably more complex concepts, as we discuss throughout this paper. Recently, category theory has been applied to resolve research problems in computer networks, particularly in network management. For instance, researchers in [35] investigated the application of category theory to support the implementation of an intent-based networking solution. In this study, we propose to apply category theory to the formulation of theorems that give way to our algorithm and later to our implementation. Thus, our proposal uses category theory extensively in both defining and implementing our theorems and algorithm.

4) TYPE AND INFORMATION THEORIES

To complement the concepts and conditions defined by category theory toward improving fault detection, we apply type theory [36] and information theory [37] in our study.

Through propositions, type theory supports the definitions of rules and axioms that determine the types of objects. Propositions are statements that can be tested on structures and objects to determine whether they pertain to certain types. The fact that an object is of a certain type is the proof of these propositions. We use this property to determine that objects associated with faults prove the existence of fault types in semantic structures. This enables us to formulate our theorems and algorithm.

Information theory mainly focuses on the quantification of information contained in media or transmitted through a channel. Entropy represents the major result derived from information theory, which is information quantity measured as the amount of uncertainty of an event.

Entropy is estimated by studying the probability that a symbol is used among all symbols that form a language. The entropy H in bits per symbol is given by

$$H = -\sum_i p_i \log_2 p_i \tag{1}$$

where p_i represents the probability of occurrence of symbol i. Here, the base two in log ensures the result is in bits.

In this study, we use this definition in our theorems to study the amount of information contained in all knowledge items and compare it to the amount of information in knowledge items associated with faults.

5) SPECTRAL GRAPH THEORY

As discussed in [38], knowledge graphs are the best data structures to use in representing NDTs and knowledge bases. The knowledge graphs can be helpful in gaining insights about the knowledge they represent. Spectral graph theory is excellent for studying the properties of graphs [39]. It can be used to analyze the structure of a knowledge graph, such as NDTs produced by ANSA, as described later in detail. This analysis relies on the normalized Laplacian graph spectrum and computes its eigenvalues and the number of eigenvectors linked to each. Although this method has a considerably complex run time and relatively high computational burden, it is convenient because the Laplacian matrix can be obtained from individual graphs and thus does not require that all possible pairs be evaluated as with graph editing methods. Moreover, the method enables a comparison of graphs of different sizes [40].

Researchers in [41] proposed the use of a neural network to extract low-dimensional features from original data using the eigen-decomposition technique, which preserves most of the graph structure of the data as required in the application of spectral graph theory [39]. Researchers in [42] proposed using the same technique as that used in [41] while considering global information; this approach increases the number of detected faults. However, the construction of a

neural network, as proposed in both alternatives, limits the number of faults detected to only that of exposed faults. In addition, as these techniques are ML-based techniques, they considerably depend on the input data vectors togged as faults to learn how faults behave.

B. PROBLEM STATEMENT

Considering the previous overview of related studies and promising techniques, we next formulate the problem we attempt to solve in this study as follows.

Given a set of state data from a network service, detect basic faults and faults related to multiple network service configurations and operational parameters. The detection includes the present and predicted values of the operational parameters, which originate from several what-if scenarios obtained after investigating the most probable combinations of parameters that change simultaneously.

Finally, the state data used as input comprises monitoring information, configurations of network services and/or their status, and the relations among them. The configuration data typically take the form of a network service definition (NSD) provided to OSM. Moreover, the monitoring information is provided by OpenStack and/or OSM. The input data must be either quantitative or qualitative, where the latter will be either quantifiable or non-quantifiable data items.

C. RESEARCH CHALLENGES

Solving the aforementioned problem has many challenges. The network service model must be constructed to consider quantifiable and non-quantifiable data items and the relations among them. The model and associated functions must be able to assess changes in multiple parameters either simultaneously or independently, where some parameters change and others do not, when used by the network service. In addition to randomly generated changes, the investigation of network service changes must assess changes in network service configurations, states, and knowledge relations.

A fault detection algorithm must have two key features. First, it must be able to identify the information and/or knowledge items from network services that are directly linked to fault or quasi-fault states. Second, it must be able to infer the information and/or knowledge items that are indirectly linked to other information and/or knowledge items that are directly linked to fault or quasi-fault states. The network service element representation in relation to its real network service element counterparts must have high fidelity (i.e., high precision and low delay) and must not contain incoherent information to avoid fault detection errors. Moreover, it must be able to identify the relationships among heterogeneous information and support complex information items and complex relations. Finally, the solution must be able to process a large amount of information in a considerably short time.

III. AUTOMATED NETWORK SERVICE ANALYSIS

We have thus far introduced the problem we aim to solve using ANSA and discussed related studies and our proposed theorems as part of our solution. We next discuss these theorems and how ANSA uses them for fault detection. Moreover, we present and discuss our results and discuss its key aspects and benefits.

We propose to solve the problem through ANSA, a mechanism that uses the network service configuration and state information to construct an NDT representing the network service. The NDT enables the investigation of what-if scenarios while considering relations among input data items multiple parameters that change simultaneously, and changes to some parameters linked to changes to other parameters. We construct the NDT as a knowledge graph. The vertices of the knowledge graph are network elements (e.g., switches), components (e.g., interfaces), metrics (e.g., bandwidth), values (e.g., measurements), etc. The edges of the knowledge graph are the predicates that relate the vertices (e.g., the predicates shown in Table 8).

The information provided by the network service is contained in an NSD that is then combined with monitoring and state reports as well as external common vulnerabilities and exposures (CVEs) reports. For its proper representation in the NDT, it must first be structured based on an ontology that applies knowledge and semantic methods for ensuring coherence among input data items. The ontology contains the definitions of data-item-type relations and supports the guided selection of parameters that change and that of their allowable values. We then apply category, type, and information theories to add additional structure to the information and knowledge retrieved from the network service monitoring elements and later contained in the NDT. Our aim is to exploit the properties of this information to derive long-chained connections between all information and knowledge items, thereby guiding the selection of parameters to change. In addition, we propose to use these theories to ensure coherence among input data items.

The results obtained with category, type, and information theories are complemented using AI methods to analyze the NDT and potential what-if scenarios. In particular, we propose the use of ML and other regression methods to project several historic measurements on time and apply the NDT to the what-if scenarios, which are analyzed to obtain conditional predictions. We also use polynomial fit (PolyFit) and autoregressive integrated moving average (ARIMA) as regression methods and multilayer perceptron (MLP) as an ML method.

A major aspect of our solution is the identification of NDT as a monad of the KO category. It allows formal definitions of theorems to be used in a verifiable implementation. This ensures that the results obtained with our NDT and algorithm are predictable and verifiable and include what-if computations. This means that identified faults will have formal justifications and validated configurations either for new network services or previous network services that have

changed. These are associated with formal constructions that represent strong evidence of their correctness and support their deployment with confidence.

The results of ANSA can be exploited by service providers to improve network services through their management systems. ANSA reports contain information that can be used to modify network service configurations. Particularly, the reports indicate the identifiers used by management systems to identify the elements affected by detected faults, such as a network element (e.g., a link) or a metric (e.g., bandwidth use), as shown in Fig. 2 and Table 8.

A. THEOREMS

Before formulating the algorithm proposed to solve the problem defined earlier, we must first formulate the following definitions and theorems. All equations are presented using the common notation of category theory [26]. In brief, stylized uppercase letters denote categories, other uppercase letters denote objects from a category, lowercase letters denote morphism names, and arrows denote morphisms. All other symbols have their common mathematical meanings.

Definition 1: The NDT is based on a monad in the category of KOs with some additional relations—constraints.

We begin in detail with the categories of knowledge items $\mathcal{C}$ and KOs $\mathcal{D}$. We first use the monad definition introduced earlier to identify the state-like monad $(T, \eta, \mu) \in \mathcal{D}$, which is a proto-NDT, as follows:

$$\begin{aligned} T &: \mathcal{D} \to \mathcal{D} \\ \eta &: 1_{\mathcal{D}} \to T \\ \mu &: T^2 \to T \end{aligned} \tag{2}$$

For our purposes, we want the monad T to follow the same behavior as that of the well-known state-like monad [26]. For this, we define the endofunctor T as follows:

$$T : a \to (s \Rightarrow a' \times s') \tag{3}$$

where s and s' represent states, the relation $\Rightarrow$ represents the state change, and the objects a and a' represent arbitrary objects associated with the states and state change, respectively.

Obtaining an NDT in $\mathcal{D}$ requires considering additional relations in the form of two slightly more complex morphisms in $\mathcal{D}$. On the one hand, these morphisms express the property that the NDT is updated with additional information; on the other hand, they express the property that the NDT is stabilized after incorporating new information. Thus, we formulate the stabilization morphism f_{sta}, the update morphism f_{upd}, and the resulting NDT as follows:

$$\begin{aligned} f_{\text{sta}} &: s \to s' \\ f_{\text{upd}} &: a \to (s \to (a', s')) \\ T\, f_{\text{upd}} &: T\ a \to T\ (s \Rightarrow a' \times f_{\text{sta}}\ s') \\ N &\equiv (T, \eta, \mu) \\ \text{NDT} &\equiv (N, f_{\text{sta}}, f_{\text{upd}}) \end{aligned} \tag{4}$$

Both morphisms f_{upd} and f_{sta} are meant to be particularized and linked to functions when the NDT is implemented. In particular, ANSA enhances the stabilization morphism by formulating two separate properties wherein the hidden faults are raised and that the objects are typed. Thus, the hidden fault raising morphism f_{HFR}, the typing morphism f_{typ}, and the ANSA-NDT structure are formulated as follows:

$$\begin{aligned} f_{\text{HFR}} &: s \to s' \\ f_{\text{typ}} &: s \to s' \\ f_{\text{sta}} &= f_{\text{typ}} \circ f_{\text{HFR}} \\ \text{ANSA-NDT} &\equiv (N, f_{\text{typ}} \circ f_{\text{HFR}}, f_{\text{upd}}) \end{aligned} \tag{5}$$

Notably, f_{upd}, f_{sta}, f_{HFR}, and f_{typ} are mappings related to the properties of category theory, but their implementation involves inspecting the contents of objects to determine the destination object for each source object. This is totally compliant with category theory constraints. Thus, the implementation of the NDT is guided by computability and verifiability constraints, as provided by the monadic structure of the NDT. It is obtained by delving within abstraction layers until obtaining the final functional-style code with highly verifiable correctness.

Furthermore, because the structure of T follows the so-called state monad, the implementation of the NDT can reuse that of the state monad such that issues can be properly distinguished and the quality of produced code can be improved. Therefore, implementation logic relies on functions that handle data, such as implementing the logic for applying the hidden fault raising theorem as straight functions, and they avoid needing to manage the complexity of container data structures and prevents the occurrence of any side effects of complex functions.

Definition 2: An NDT is populated by mapping objects and predicates from a knowledge graph.

Given that $\mathbb{K}$ is the collection of unprocessed and uncategorized knowledge items, based on the tuples containing subject, predicate, object—named triples in semantic methods—as $\{s_i, p_j, o_k\} \in \mathbb{K}$, we formulate the population and predicate mapping to the category of knowledge items $\mathcal{C}$ as follows.

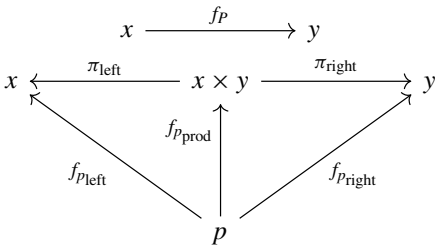
Beginning with three objects $x, y, p \in \mathbf{Ob}(\mathcal{C})$ and associating x with the subject s_i, p with the predicate p_j and y with the object o_k, we obtain the morphisms in $\mathcal{C}$ that correspond to $\{s_i, p_j, o_k\} \in \mathbb{K}$ as:

$$\begin{aligned} f_P &: x \to y \\ f_{p_{\text{prod}}} &: p \to x \times y \end{aligned}. \tag{6}$$

Although $f_{p_{\text{prod}}}$ does not need to be unique because it relates the three elements x, y, p, we must check the relation between x, y, and the resulting categorical product $x \times y$. Thus, we formulate:

$$\begin{aligned} &f_P : x \to y\ \forall \{s_i, p_k, o_j\} \in \mathbb{K}, \\ &\exists (x \to y)\ \in \mathbf{Hom}(\mathcal{C}),\ \exists (p \to x \times y)\ \in \mathbf{Hom}(\mathcal{C}), \end{aligned} \tag{7}$$

which verifies the following commutative diagram:



(8)

We next formulate the additional property of f_{upd} that connects $\mathcal{C}$ with $\mathcal{D}$ through the functor $F : \mathcal{C} \to \mathcal{D}$ as follows:

$$\begin{gathered} f_{\text{upd}} : a \to (s \to (a', s')), \;\; a, s, a', s' \in \mathbf{Ob}(\mathcal{D}) \\ \Longleftrightarrow \\ \exists x \in \mathbf{Ob}(\mathcal{C}), F : x \to s', F : x \not\to s \end{gathered} . \tag{9}$$

Verifying that an empty NDT is given by $F\ s = \emptyset$ when no object in $\mathcal{C}$ is incorporated into the NDT is trivial. The NDT is then populated by subsequent verification of as many f_{upd} as needed. Therefore, we can state, for instance, that $s_{\mathcal{D}} = (f^3_{\text{upd}}\ f^2_{\text{upd}}\ f^1_{\text{upd}}\ F\ s_{\mathcal{C}})$ has some population regardless of the initial s.

Definition 3: Items forming an NDT are typed, and the typing follows the monad and category theory constraints.

Given the category of knowledge items $\mathcal{C}$, which is the basis for formulating the category of KOs $\mathcal{D}$, we formulate the category of KO types $\mathcal{T}$ as a subcategory of $\mathcal{C}$, a functor that maps objects from $\mathcal{C}$ to $\mathcal{T}$, and the equations that ensure their consistency as follows.

We begin with a type $s \in \mathcal{T}$, which is the typing functor that associates the type with an object. This is formulated as:

$$\begin{aligned} &F_{\mathcal{CT}} : \mathcal{C} \to \mathcal{T} \\ &F_{\mathcal{CT}} : (x \to y) \;\; \to (s_x \to s_y) \\ &F_{\mathcal{CT}} : \mathbf{id}_x \to \mathbf{id}_s \\ &\qquad : (x \to x) \;\; \to (s_x \to s_x) \\ &F_{\mathcal{CT}} : x \to s \end{aligned} \tag{10}$$

This means that an object exists in $\mathcal{C}$ and $\mathcal{T}$, which is known as the typing predicate $p_{\mathcal{T}} \in \mathbf{Ob}(\mathcal{T})$. This typing predicate is related to $s \in \mathbf{Ob}(\mathcal{C})$ such that it meets the following relations:

$$\begin{aligned} f_{p_{\text{prod}}} &: p_{\mathcal{T}} \to x \times s \\ f_{p_{\text{prod}}} &: p_{\mathcal{T}} \to x \times (F_{\mathcal{CT}}\ x) \end{aligned} . \tag{11}$$

Our NDT formulation requires a means of identifying the object type. Because our NDT is based on a knowledge graph, we must formulate an approach to specify and identify the predicates that must be linked to an object for it to be a certain type s. Thus, we formulate the morphism h^p and its implication on f_p as follows:

$$\begin{aligned} h^p &: s \to p \\ h^{p_k} &: s_i \to p_k \\ &\qquad \forall k \in \{1 \ldots n\}. \\ f_{p_{\text{prod}}} &: p_k \to s_i \times s_j \\ f_{p_k} &: s_i \to s_j \end{aligned} \tag{12}$$

The last morphism relates a type object S_i to another type object S_j. This indicates that even a type has a type. The top-level type would be known as the one that references itself.

Ensuring the consistency of types requires applying the reasoning from our categories to the trivial Boolean category $\mathbb{B}$, the objects of which are `false` and `true` and which have only identity morphisms. Additionally, typing requires two types of morphisms: one to indicate that all predicates of a set must hold and another to indicate that some predicates of a set must hold. We call them $h^{p\ \forall}$ and $h^{p\exists}$, respectively. For simplicity, we consider that $b \in \mathbf{Ob}(\mathcal{C})\ \forall b \in \mathbb{B}$. Thus, given $x, p, y, b \in \mathbf{Ob}(\mathcal{C})$ and $b \in \mathbb{B}$, we have:

$$\begin{gathered} h^{p\ \forall} : s \to (x \times p \times y \Rightarrow b) \\ h^{p\exists} : s \to (x \times p \times y \Rightarrow b) \\ \exists (x \to s,\;\; p_{\mathcal{T}} \to x \times s) \\ \Longleftrightarrow \\ \exists (p_k \to x \times y) \;\; \in \mathbf{Hom}(\mathcal{C}) \\ \forall (s \to p_k) \in \mathbf{Hom}(\mathcal{T}) \\ \wedge \\ h^{p\forall}_j\ s\ (x \times p \times y) \;\; = \texttt{true} \\ \forall (x \to y) \;\; \in \mathbf{Hom}(\mathcal{C}) \\ (p \to x \times y) \;\; \in \mathbf{Hom}(\mathcal{C}) \\ \forall (h^{p\forall}_j : s \to (x \times p \times y \Rightarrow b)) \in \mathbf{Hom}(\mathcal{T}) \\ \wedge \\ \exists h^{p\exists}_j\ s\ (x \times p \times y) \;\; = \texttt{true} \\ \mid (x \to y) \;\; \in \mathbf{Hom}(\mathcal{C}) \\ (p \to x \times y) \;\; \in \mathbf{Hom}(\mathcal{C}) \\ \mid (h^{p\exists}_j : s \to (x \times p \times y \Rightarrow b)) \in \mathbf{Hom}(\mathcal{T}) \end{gathered} \tag{13}$$

$$\begin{gathered} \forall p_k \in \mathbf{Ob}(\mathcal{C}) \\ \exists (p_k \to (F_{\mathcal{CT}}\ x) \;\; \times (F_{\mathcal{CT}}\ y)) \in \mathbf{Hom}(\mathcal{C}) \\ \Longrightarrow \\ \exists (p_k \to x \times y) \;\; \in \mathbf{Hom}(\mathcal{C}) \end{gathered} \tag{14}$$

$$\begin{gathered} \forall x, y \in \mathbf{Ob}(\mathcal{C}) \\ \exists (p_k \to (F_{\mathcal{CT}}\ x) \;\; \times (F_{\mathcal{CT}}\ y)) \in \mathbf{Hom}(\mathcal{C}) \\ \Longrightarrow \\ \exists (f_{p_k} : F_{\mathcal{CT}}\ x \to F_{\mathcal{CT}}\ y) \;\; \in \mathbf{Hom}(\mathcal{C}) \end{gathered} \tag{15}$$

$$\begin{gathered} \forall x, y \in \mathbf{Ob}(\mathcal{C}), \\ \exists (p_k \to x \times y) \;\; \in \mathbf{Hom}(\mathcal{C}) \\ \Longrightarrow \\ \exists (f_{p_k} : x \to y) \;\; \in \mathbf{Hom}(\mathcal{C}) \end{gathered} . \tag{16}$$

A major contribution of this study is the elaboration of structures required to detect hidden faults. It is based on the existence of at least one semantic path that connects the knowledge items that evidence a hidden fault and those that evidence one or more exposed faults. This theorem makes ANSA particularly beneficial for network service providers in preventing potential faults by knowing the indirect impact that their decisions can have on their network services. (17)

For instance, in terms of intent-based networking, which is an additional scenario for the application of ANSA, this theorem informs network service tenants that their intentions produce an invisible faulty state (i.e., a hidden fault). We will investigate this in future research.

The theorem is formulated as follows.

Theorem 1 (Hidden Fault Raising): Hidden faults can be raised because an isomorphism exists between an object collection taken from the presumably bigger collection of objects linked to exposed faults of a network service as well as the collection of objects linked to hidden faults.

Proof: A KO $\mathbb{K}_i$ is a collection of knowledge items that are semantically structured according to a particular ontology. Many equivalent methods exist for representing items. We represent each item as a triple, which is the common name in semantic methods for a tuple whose components are subject, predicate, and object (i.e., $\{s, p, o\} \in \mathbb{K}_i$).

The category $\mathcal{C}$ is derived from $\mathbb{K}$. Subjects, predicates, and objects in $\mathbb{K}$ provide objects in $\mathcal{C}$; triples in $\mathbb{K}$ also yield morphisms in $\mathcal{C}$. Hence, if a triple $\{s, \_, \_\}$ exists in $\mathbb{K}$, an object x_s exists in $\mathbf{Ob}(\mathcal{C})$ corresponding to s; if a triple $\{\_, p, \_\}$ exists in $\mathbb{K}$, an object x_p exists in $\mathbf{Ob}(\mathcal{C})$ corresponding to p; if a triple $\{\_, \_, o\}$ exists in $\mathbb{K}$, an object x_o exists in $\mathbf{Ob}(\mathcal{C})$ corresponding to o; and, finally, if a triple $\{s, p, o\}$ exists in $\mathbb{K}$, a morphism $f_p : x_s \to x_o$ exists in $\mathbf{Hom}(\mathcal{C})$ corresponding to the relation existing between s and o, which is defined as being through p. Although x_p would be compounded to either the source or destination of the morphism f_p, we chose to represent this as defining different morphisms f_p for each p.

KOs $\mathbb{K}_i \subset \mathbb{K}$ have a triple $\{s, \mathrm{id}, s\} \in \mathbb{K}_i$, where id is the identity predicate that relates $\mathbb{K}_i$ to itself using s as a self-reference. These triples are translated to identity morphisms in $\mathcal{C}$. The mapping of $\mathbb{K} \to \mathcal{C}$ considers that identity triples exist for every object in $\mathbb{K}$ without incurring in any information degradation. Therefore, all objects in $\mathcal{C}$ have an identity morphism. This verifies the identity property and unitality condition introduced earlier, which are necessary for properly defining category $\mathcal{C}$.

Semantic relations in $\mathbb{K}$ are transitive. A semantic transition between $\{s, p, o\}$, $\{s', p', o'\}$, and $\{s'', p'', o''\}$ exists if $s' = o$ and $s'' = o'$. Therefore, the mapping from predicates to morphisms results in $f : s \to s' \in \mathcal{C}$, $g : s' \to s'' \in \mathcal{C}$, and $h : s'' \to o'' \in \mathcal{C}$. These morphisms can be composed; therefore, it holds that $h \circ g \circ f : s \to o''$. The composition is associative; therefore $(h \circ g) \circ f : h \circ (g \circ f)$. Associativity means that in $\mathbb{K}$, the order in which predicates are followed in a path is irrelevant to the resulting trace. This holds by definition of the predicate and trace in $\mathbb{K}$ and verifies the composition property and associativity condition introduced earlier, which are also necessary for defining category $\mathcal{C}$.

With the properties defined earlier, the derivation of category $\mathcal{C}$ from $\mathbb{K}$ is straightforward and consistent. However, the properties that ensure this consistency must be checked by the applications that use the concepts of this theorem and proof while they construct these structures. We next elaborate this base to find faults.

The set $z \in \mathbf{Ob}(Set)$ is an object of the category *Set* whose elements $(\forall z_i \in z)$ are linked to some fault, where we use lowercase for z to mark it is an object of the category *Set* and as not just a set. Elements not linked to any fault are $\forall w_j \in \emptyset$ considering $\emptyset \in \mathbf{Ob}(Set)$. This indicates that some $\emptyset \to w_j$ exists, which is absurd. Therefore, no element can be linked to any fault in z.

The functor $F : \mathcal{C} \to Set$ maps; on the one hand, each object $x \in \mathbf{Ob}(\mathcal{C})$ to $(F\ x) \subset z$ if x is linked to a fault. Otherwise, it collapses the element to the empty set $F\ x = \emptyset$. We assume that $\emptyset \in z$. Thus, we have $F : x \to F\ x\ \forall x \in \mathbf{Ob}(\mathcal{C})$. On the other hand, the functor F maps each morphism $f : x \to y \in \mathbf{Hom}(\mathcal{C})$ to $F\ f : F\ x \to F\ y \in \mathbf{Hom}(Set)$, which obviously can be $F\ f : F\ x \to \emptyset$ or $F\ f : \emptyset \to F\ y$.

The functor $\mathcal{C}(a, -)$ maps each element $x \in \mathbf{Ob}(\mathcal{C})$ to the set of morphisms $\mathcal{C}(a, x)$, which represent the set of bound predicates $\{a_{\mathbb{K}}, \_, x_{\mathbb{K}}\} \in \mathbb{K}$. If no morphism $a \to y$ exists, meaning no predicate in $\mathbb{K}$ links $a_{\mathbb{K}}$ to $y_{\mathbb{K}}$, the functor $\mathcal{C}(a, -)$ maps y to the empty set $\emptyset$.

A natural transformation $\alpha \in \mathbf{Nat}(-, -)$ maps the functor $\mathcal{C}(a, -)$ to F. The component of α at x, namely, α_x, maps the set of morphisms $\mathcal{C}(a, x) = \{(a \to x)\ \forall x \in \mathbf{Ob}(\mathcal{C})\}$ to the set $F\ x$ or $\emptyset$. Thus, we have:

$$\begin{array}{l} \alpha \in \mathbf{Nat}(\mathcal{C}(a, -), F) \\ \alpha : \mathcal{C}(a, -) \to F \\ \alpha_x = \alpha\ x \\ \alpha_x : \mathcal{C}(a, x) \to F\ x \\ \alpha_x : Set \to Set \end{array} . \tag{18}$$

From Yoneda's lemma [26], [27], we know that if we find an element $v \in F\ x$, $\alpha_x : \mathcal{C}(a, x) \to F\ x$ exists. To show how it holds, we solve the following problem:

$$\begin{array}{l} \text{Given:} v \in F\ x \\ \text{Get:} \alpha_x \in \mathbf{Nat}(\mathcal{C}(a, -), F) \end{array} \tag{19}$$

The solution is obtained as follows:

$$\begin{array}{rcl} \exists v \in F\ x & \Longrightarrow & \exists x \in \mathbf{Ob}(\mathcal{C}) \mid x \to F\ x \\ \mathcal{C}(a, -) & : & x \to \mathcal{C}(a, x) \\ \exists x \in \mathbf{Ob}(\mathcal{C}) & \Longrightarrow & \exists \mathcal{C}(a, x) \\ \exists \mathcal{C}(a, x),\ \exists F\ x & \Longrightarrow & \exists \alpha_x : \mathcal{C}(a, x) \to F\ x \\ \alpha_X & : & \mathcal{C}(a, x) \to F\ x \\ f & \in & \mathcal{C}(a, x) \\ f & : & a \to x \\ \alpha_x\ f & = & v \end{array} . \tag{20}$$

The solution is easily proven as valid by checking that the following diagram commutes:

$$\begin{array}{ccc} \mathcal{C}(a,a) & \xrightarrow{\mathcal{C}(a,f)} & \mathcal{C}(a,x) \\ \downarrow{\scriptstyle \alpha_a} & & \downarrow{\scriptstyle \alpha_x} \\ F\ a & \xrightarrow{F\ f} & F\ x \\ a & \xrightarrow{f} & x \end{array} . \tag{21}$$

From this we know that $\alpha_x\ f = F\ f\ \alpha_a\ \mathbf{id}_a$ holds and that the natural transformation α is completely determined by α_a, which is a key consideration in proving Yoneda's lemma. We show in the following that the twin of the problem is also formulated and solved in this manner.

Consequently, if we find $v \in F\ x$, we obtain α_x, which, by definition, relates a and x and which is $\alpha_x : \mathcal{C}(a,x) \rightarrow F\ x$. We notice that $F\ y = \emptyset \iff \nexists v \in F\ y$, and thus $\nexists \alpha_y \in \mathbf{Nat}(\mathcal{C}(a,-),F)$.

Given the definition of the functor F, the set $F\ x$ exists if (and only if) a fault exists in the element that originates x through $\mathbb{K}$ and $\mathcal{C}$. Therefore, $a \rightarrow x \in \mathbf{Hom}(\mathcal{C}) \implies F\ a \neq \emptyset$. Therefore, clearly, a is also linked to a fault. This is verified by replacing x by a in the aforementioned problem, which provides the solution $\alpha_a : \mathcal{C}(a,a) \rightarrow F\ a$ We then observe that $f = \mathbf{id}_a$.

Once a is linked to a fault ($F\ a \subset z$), we know that any other object mapped from a is linked to a fault. Thus, $F\ y \subset z\ \forall y \mid a \rightarrow y \in \mathbf{Hom}(\mathcal{C})$. This is verified by solving the following problem:

$$\begin{aligned} \text{Given:} & \alpha_a \in \mathbf{Nat}(\mathcal{C}(a,-),F) \\ \text{Get:} & y \in F\ y \end{aligned} \tag{22}$$

A solution is obtained as follows:

$$\begin{aligned} \alpha_a &: \mathcal{C}(a,a) \rightarrow F\ a \\ f &: a \rightarrow y \\ F\ f &: F\ a \rightarrow F\ y \\ F\ f\ \alpha_a &: \mathcal{C}(a,a) \rightarrow F\ y \\ F\ f\ \alpha_a\ \mathbf{id}_a &= y \in F\ y \\ y &= F\ f\ \alpha_a \end{aligned} . \tag{23}$$

The solution is easily proven as valid by checking that the following diagram commutes:

$$\begin{array}{ccc} \mathcal{C}(a,a) & \xrightarrow{\mathcal{C}(a,f)} & \mathcal{C}(a,y) \\ \downarrow{\scriptstyle \alpha_a} & & \downarrow{\scriptstyle \alpha_y} \\ F\ a & \xrightarrow{F\ f} & F\ y \\ a & \xrightarrow{f} & y \end{array} . \tag{24}$$

In summary, for a given a triple $\{s,p,o\} \in \mathbb{K}$, we have that the subject is an element $a \in \mathbf{Ob}(\mathcal{C})$, the object is another element $x \in \mathbf{Ob}(\mathcal{C})$, and their relation through the predicate forms their mapping, namely, $(a \rightarrow x) \in \mathbf{Hom}(\mathcal{C})$. We know that if $(F\ x) \subset z$, all other objects to which a is connected are also in z; that is:

$$\begin{aligned} (F\ x) \subset z &\implies (F\ y) \subset z \\ \forall y \in \mathbf{Ob}(\mathcal{C}),\ y \neq x,\ (a \rightarrow y) &\in \mathbf{Hom}(\mathcal{C}) \end{aligned} . \tag{25}$$

Therefore, hidden faults exist in all other objects to which a is connected. On the other hand, if we find some $\alpha_z \in \mathbf{Nat}(\mathcal{C}(a,-),F)$, we will also find an object $a' \in F\ a$ that allows us to follow the procedures described earlier to find new elements of z by hidden fault raising. ■

A critical aspect that supports the stability of our NDT is the recognition that, although typing is strong, an object type emerges from the object context. This yields the following theorem.

Theorem 2 (Strong Emergent Typing): KO types are dynamic but unequivocal.

Proof: From the previous theorems, we recall that a KO K is a collection of knowledge atoms (subset) from a knowledge base $\mathbb{K}$, and thus $K \subset \mathbb{K}$. We also recall that a knowledge atom is an element of knowledge defined in the knowledge base $\mathbb{K}$ as a triple: {subject, predicate, object} $\in \mathbb{K}$, or simply $\{s,p,o\}$.

Every KO is initially untyped. Its type is determined by the atoms constituting it. To assign types to KOs consistently, we define the category $\mathcal{C}$. Every knowledge atom is translated to $\mathcal{C}$ by defining objects for the ends s and o and a morphism for the predicate p. Thus, we have the relations listed in Table 2.

TABLE 2. $\mathbb{K}$-$\mathcal{C}$ relations.

From $\mathbb{K}$	To $\mathcal{C}$
s	$a \in \mathbf{Ob}(\mathcal{C})$
p	$p \in \mathbf{Ob}(\mathcal{C})$
o	$b \in \mathbf{Ob}(\mathcal{C})$
$\{s,p,o\}$	$f_p : a \rightarrow b$ $f_p \in \mathbf{Hom}(\mathcal{C})$

As the s component of an atom fully identifies the KO to which the atom pertains, only a single s will be contained in each KO. The o component is an independent element or one that corresponds to the s component of another atom. If an element x is independent, it can reside in the right half of some morphisms but cannot be in the left half of other morphisms. Thus, $(a \rightarrow x) \in \mathbf{Hom}(\mathcal{C})$, $(x \rightarrow \_) \notin \mathbf{Hom}(\mathcal{C})$, or equivalently $\mathcal{C}(x,-) = \emptyset$.

Second, we define the category $\mathcal{T}$ that has an object for each type that can be assigned to a KO and thus identified in $\mathbb{K}$. We then use $\mathcal{T}$ to assign types to the objects from $\mathcal{C}$. The type assignation is defined by the functor G as follows:

$$\begin{aligned} G &: \mathcal{C} \rightarrow \mathcal{T} \\ G &: \emptyset \rightarrow v \\ G &: x \rightarrow w_{\mathrm{P+C}} \end{aligned} \tag{26}$$

We define $v \in \mathbf{Ob}(\mathcal{T})$ as the void type and $w_{\mathrm{P}\times\mathrm{C}} \in \mathbf{Ob}(\mathcal{T})$ as a type of $x \in \mathbf{Ob}(\mathcal{D})$. Types are can be network elements,

machines, etc. The type is determined by the predicates and conditions defined in {P + C}. Thus, we have:

$$\begin{aligned} &\forall\, x \in \mathbf{Ob}(\mathcal{C}), \\ &(x \to w_{\mathrm{P+C}}) \iff (f_p : x \to y) \in \mathbf{Hom}(\mathcal{C})\ \forall p \in \mathrm{P}\,. \\ &\qquad\qquad \wedge\ (f_c\ y) = \text{true}\ \forall c \in \mathrm{C} \end{aligned} \tag{27}$$

This results in a bidirectional typing system. A type will be associated with an object if its definition in $\mathbb{K}$ has some particular predicates and the objects to which it is connected accomplish a particular condition. In addition, a type will exist if a set of predicates and conditions is fulfilled by an object.

Any change to the predicates or objects linked to a subject will result in a change in the type of KO identified with that subject. Thus, the type emerges from the knowledge items associated with an object and not vice versa. However, as shown in the categorical definition, which refers only to straight or collapsing functors, each KO has an unequivocal path from $\mathbb{K}$ to $\mathcal{T}$ (i.e., $\mathbb{K} \supset K \to \mathcal{C} \to \mathcal{T}$). Therefore, as stated by the strong emergent typing theorem, the type of each KO has a strong type that emerges unequivocally from its knowledge items. ■

One of the major findings of this study is the relation between the quantity of monitoring data obtained and processed for each type and the ability to identify faults. This led to the following theorem.

Theorem 3 (Fault Qualification): Knowledge elements provided to a fault-detection NDT must equally target system faults and normal system operations.

Proof: From the definition of category $\mathcal{C}$, we identify two meta-elements, $\mathcal{X} = \mathbf{Ob}(\mathcal{C})$ and $\mathcal{Y} = \mathbf{Hom}(\mathcal{C})$, where the latter is a meta-morphism that takes the form $\mathcal{Y} : \mathbf{Ob}(\mathcal{C}) \to \mathbf{Ob}(\mathcal{C})$. Therefore, it is $\mathcal{Y} : \mathcal{X} \to \mathcal{X}$, and thus $\mathcal{Y} = \mathbf{id}_{\mathcal{X}}$.

We then translate $\mathcal{X}$ and $\mathcal{Y}$ to symbols in alphabet $\mathbb{A}$. Therefore, we have $\mathbb{A}_{\mathcal{X}} \in \mathbb{A}$ and $\mathbb{A}_{\mathcal{Y}} \in \mathbb{A}$. This enables us to compose messages in the form $\mathbb{A}_{\mathcal{X}}\mathbb{A}_{\mathcal{Y}}\mathbb{A}_{\mathcal{Y}}\mathbb{A}_{\mathcal{X}}\mathbb{A}_{\mathcal{X}}\ldots$, which is homomorphic to the underlying semantic relations in $\mathbb{K}$ (as defined in the context of the hidden fault raising theorem). The result preserves in $\mathbb{A}$ the multiplicity of both objects and morphisms present in $\mathcal{C}$.

Studying the structure of $\mathbb{A}$, we find some qualities that $\mathcal{C}$ must have to characterize the detection of hidden faults. First, we recall the definition of entropy from information theory as follows:

$$H = -\sum_i p_i \log_2 p_i \tag{28}$$

This allows us to study different scenarios, each with a particular probability p_i for each symbol $z = P(\mathbb{A}_{\mathcal{X}})$ and $(1 - z) = P(\mathbb{A}_{\mathcal{Y}})$. As each morphism involves two objects, namely, $P(\mathbb{A}_{\mathcal{X}}) \geq P(\mathbb{A}_{\mathcal{Y}})$, we only study the cases that keep this relation. Thus, we obtain the entropies listed in Table 3.

The results in this table reveal that the greater the difference between the number of morphisms and objects in $\mathcal{C}$, the less information is obtained from new knowledge added to $\mathbb{K}$. Thus, the construction of $\mathcal{C}$ must be tuned to include as many predicates from $\mathbb{K}$ as possible, which is congruent with the fact that the relations between category objects (i.e., morphisms) are critical in qualifying the properties of any object collection, which is one of the essentials of category theory [26].

TABLE 3. Fault entropies.

$P(\mathbb{A}_{\mathcal{X}})$	$P(\mathbb{A}_{\mathcal{Y}})$	H (expr)	H
$\frac{1}{2}$	$\frac{1}{2}$	$-\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right)$	1
$\frac{3}{4}$	$\frac{1}{4}$	$-\left(\frac{3}{4}\log_2\frac{3}{4} + \frac{1}{4}\log_2\frac{1}{4}\right)$	≈ 0.81
$\frac{7}{8}$	$\frac{1}{8}$	$-\left(\frac{7}{8}\log_2\frac{7}{8} + \frac{1}{8}\log_2\frac{1}{8}\right)$	≈ 0.54
$\frac{9}{10}$	$\frac{1}{10}$	$-\left(\frac{9}{10}\log_2\frac{9}{10} + \frac{1}{10}\log_2\frac{1}{10}\right)$	≈ 0.47
...	...	...	...

TABLE 4. Symbol map.

$\mathcal{D}$	$\mathbb{A}$
z	$\mathbb{A}_{\mathcal{Z}}$
$\mathbf{id}_z$	$\mathbb{A}_{\mathcal{Y}}$
$\emptyset$	$\mathbb{A}_{\mathcal{V}}$
$\mathbf{id}_\emptyset$	$\mathbb{A}_{\mathcal{W}}$

Extending these results to differentiate between faults and normal operations provides greater insight into the working space for fault detection. Thus, we proceed to define a new category $\mathcal{D}$, which has two objects, z and $\emptyset$, and an identity morphism associated with each object, namely, $\mathbf{id}_z$ and $\mathbf{id}_\emptyset$, respectively.

We also define the functor $G : \mathcal{C} \to \mathcal{D}$, which maps each object $x \in \mathbf{Ob}(\mathcal{C})$ to z if the object is related to a fault; otherwise, it maps it to $\emptyset$; and it maps each morphism $(f : x \to y) \in \mathbf{Hom}(\mathcal{C})$ to $\mathbf{id}_z$ if it is related to a fault; otherwise, it maps it to $\mathbf{id}_\emptyset$.

The object $z \in \mathbf{Ob}(\mathcal{D})$ is the categorical conception of the set with the same name defined for the hidden fault raising theorem. Thus, the functor G has some commonalities with the functor F of this theorem. Although G collapses all elements from $\mathcal{C}$ into z, allowing z to be a set, we obtain $z \cong F\ x\ \forall x \in \mathbf{Ob}(\mathcal{C})$.

Although we use set semantics to define the functor F, we use category semantics to define the functor G. Thus, the set $F\ x$ exists only if x is a fault, and is absent otherwise. The absence is represented by $\emptyset$, the empty set. In addition, G maps both faults and normal operations to objects $z \in \mathbf{Ob}(\mathcal{D})$ and $\emptyset \in \mathbf{Ob}(\mathcal{D})$, respectively. For consistence, we use $\emptyset$ as the object to which normally operating objects are mapped.

To study the implications of the structure of $\mathcal{D}$ in fault detection, we map each of its elements to a symbol of the alphabet $\mathbb{A}$ to derive the map shown in Table 4:

We obtain Table 5 by studying the new structure of $\mathbb{A}$ for different scenarios and using the same procedure we previously used to ensure that more symbols are associated with objects than with morphisms

For the new $\mathbb{A}$, this table confirms that maximizing the use of predicates still produces higher entropy. It also confirms that, although a priori, a trend of adding elements that could

TABLE 5. Symbol probabilities.

$P(\mathbb{A}_{\mathcal{Z}})$	$P(\mathbb{A}_{\mathcal{Y}})$	$P(\mathbb{A}_{\mathcal{V}})$	$P(\mathbb{A}_{\mathcal{W}})$	H
$\frac{1}{2}$	0	$\frac{1}{2}$	0	1
$\frac{1}{2}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	≈ 1.79
$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	≈ 2
$\frac{1}{8}$	$\frac{3}{8}$	$\frac{1}{8}$	$\frac{3}{8}$	≈ 1.81
$\frac{1}{8}$	$\frac{7}{24}$	$\frac{7}{24}$	$\frac{7}{24}$	≈ 1.93
$\frac{1}{20}$	$\frac{9}{20}$	$\frac{1}{20}$	$\frac{9}{20}$	≈ 1.47
$\frac{1}{20}$	$\frac{19}{60}$	$\frac{19}{60}$	$\frac{19}{60}$	≈ 1.79
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{8}$	≈ 1.75
$\frac{3}{5}$	$\frac{1}{5}$	$\frac{1}{10}$	$\frac{1}{10}$	≈ 1.57
$\frac{13}{20}$	$\frac{1}{4}$	$\frac{1}{20}$	$\frac{1}{20}$	≈ 1.34
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

raise faults naturally will reduce the information and thus the quality of the system.

From these results emerges the first major conclusion of the *fault quantification* theorem, which states that knowledge elements provided to a fault-detection NDT through $\mathbb{K}$ must equally target system faults and normal system operations.

A direct result from this conclusion is that the actual information provided by a system that appears to have no fault is initially lower and it is increased by adding new elements that include some faults (while carefully avoiding any type of replacement) so that the final elements in $\mathbb{A}_{\mathcal{Z}}$, $\mathbb{A}_{\mathcal{Y}}$, $\mathbb{A}_{\mathcal{V}}$, and $\mathbb{A}_{\mathcal{W}}$ are as balanced as possible.

As the precursor of $\mathcal{D}$ and later $\mathbb{A}$, the specification of G should also be tuned to enlarge its focus on faults. ■

Both the NDT typing and strong emergent typing both support and conform to the links between objects and their types. This means that fault-related objects will have many types simultaneously. This yields the following theorem.

Theorem 4 (Fault Polymorphism): Each element of a system becomes inherently polymorphic when it is involved in or exposed to a fault, where it can play the roles of a normally operating element and a fault-related element.

Proof: Common to hidden fault raising and fault qualification theorems, we first define the knowledge structure $\mathbb{K}$. However, we also consider the precursors of $\mathbb{K}$, which are the actual physical or logical elements and their relations, which we call $\mathbb{P}$. Thus, the structure $\mathbb{K}$ has the knowledge available regarding the elements, qualities, relations, and states from $\mathbb{P}$.

We next translate the structures $\mathbb{P}$ and $\mathbb{K}$ to their category counterparts. Therefore, we have $\mathcal{P}$ and $\mathcal{C}$.

The category $\mathcal{P}$ collects the physical and virtual elements of a system as well as their relations, which are closely connected (e.g., machine A is located in laboratory B, or virtual machine V is hosted by physical machine W) or remotely connected (e.g., a link exists between machines A and C).

For example, the morphism $f_{\mathcal{P}} : a_{\mathcal{P}} \to b_{\mathcal{P}}$ represents that the machine $a_{\mathcal{P}}$ is located in laboratory $b_{\mathcal{P}}$. Similarly, the morphism $g_{\mathcal{P}} : a_{\mathcal{P}} \to C_{\mathcal{P}}$ represents that the machine $a_{\mathcal{P}}$ has a network link with the machine $C_{\mathcal{P}}$.

The category $\mathcal{C}$ collects the knowledge structures that describe the physical or virtual elements collected in $\mathcal{P}$ as well as their states and relations.

The functor $F_{\mathcal{PC}} : \mathcal{P} \to \mathcal{C}$ maps physical or virtual objects and their relations to their counterparts in the knowledge structure. Applying it to the previous example, we have:

$$\begin{aligned} F_{\mathcal{PC}} &: a_{\mathcal{P}} \to a_{\mathcal{C}} \\ F_{\mathcal{PC}} &: b_{\mathcal{P}} \to b_{\mathcal{C}} \\ F_{\mathcal{PC}} &: c_{\mathcal{P}} \to c_{\mathcal{C}} \\ F_{\mathcal{PC}} &: f_{\mathcal{P}} \to f_{\mathcal{C}} \\ F_{\mathcal{PC}} &: g_{\mathcal{P}} \to g_{\mathcal{C}} \end{aligned} \tag{29}$$

such that:

$$\begin{aligned} f_{\mathcal{C}} &: a_{\mathcal{C}} \to b_{\mathcal{C}} \\ g_{\mathcal{C}} &: a_{\mathcal{C}} \to c_{\mathcal{C}} \end{aligned}. \tag{30}$$

To further improve the formalization of the system, we use the definitions of the strong emergent typing theorem to define the category $\mathcal{T}$ that represents the object types from $\mathcal{C}$. The category $\mathcal{T}$ has its own morphisms, which have no direct counterpart in $\mathcal{C}$. These are used to define relations among types, such as inheritance and composition.

The functor $F_{\mathcal{CT}} : \mathcal{C} \to \mathcal{T}$ maps each object with its type (typing). Some morphisms from $\mathcal{C}$ are mapped to morphisms between types, particularly those representing compositions in $\mathcal{C}$, which are meaningful in the category of types. The remaining morphisms are mapped to the object that represents the "morphism" type.

Considering the previous example, we define $a_{\mathcal{T}}$ and $b_{\mathcal{T}}$ as the objects representing the host and room types, respectively, and $f_{\mathcal{T}}$, $g_{\mathcal{T}}$, and $h_{\mathcal{T}}$ as the morphisms representing their composition type, network link type, and sub-typing, respectively. With them, we have:

$$\begin{aligned} F_{\mathcal{CT}} &: a_{\mathcal{C}} \to a_{\mathcal{T}} \\ F_{\mathcal{CT}} &: b_{\mathcal{C}} \to b_{\mathcal{T}} \\ F_{\mathcal{CT}} &: c_{\mathcal{C}} \to a_{\mathcal{T}} \\ F_{\mathcal{CT}} &: f_{\mathcal{C}} \to f_{\mathcal{T}} \\ F_{\mathcal{CT}} &: g_{\mathcal{C}} \to g_{\mathcal{T}} \end{aligned}. \tag{31}$$

Adding the objects $w_{\mathcal{T}}$ and $o_{\mathcal{T}}$ to represent the type and object types, respectively. Moreover, adding the morphism $j_{\mathcal{T}} : \mathcal{T} \to \mathcal{T}$ that maps the type of inner objects (i.e., $j_{\mathcal{T}} : x \to y$ indicates that the type x is of type y) similar to the functor $F_{\mathcal{CT}}$, we have:

$$\begin{aligned} f_{\mathcal{T}} &: a_{\mathcal{T}} \to b_{\mathcal{T}} \\ g_{\mathcal{T}} &: a_{\mathcal{T}} \to a_{\mathcal{T}} \\ h_{\mathcal{T}} &: a_{\mathcal{T}} \to o_{\mathcal{T}} \\ h_{\mathcal{T}} &: b_{\mathcal{T}} \to o_{\mathcal{T}} \\ j_{\mathcal{T}} &: o_{\mathcal{T}} \to w_{\mathcal{T}} \end{aligned}. \tag{32}$$

This forms the foundation of strong typing for the categories used in fault detection. The morphism $a_{\mathcal{T}} \to b_{\mathcal{T}}$ indicates that a machine somehow contains the room in which it is located. The morphism $g_{\mathcal{T}} : a_{\mathcal{T}} \to a_{\mathcal{T}}$ indicates that a link connects two machines ($a_{\mathcal{T}}$). The morphisms

$a_{\mathcal{T}} \to o_{\mathcal{T}}$ and $b_{\mathcal{T}} \to o_{\mathcal{T}}$ indicate that both $a_{\mathcal{T}}$ and $b_{\mathcal{T}}$ are subtypes of $o_{\mathcal{T}}$. Finally, the morphism $o_{\mathcal{T}} \to w_{\mathcal{T}}$ indicates that $o_{\mathcal{T}}$ (object) is a type.

At this point, we propose to extend $\mathcal{T}$ with the object $z_{\mathcal{T}}$ that represents a failed element and the object $\emptyset_{\mathcal{T}}$ that indicates an element is operating normally. We also define the new functor $G_{\mathcal{CT}} : \mathcal{C} \to \mathcal{T}$, which maps objects from $\mathcal{C}$ to $z_{\mathcal{T}}$ if they are linked to some fault or, otherwise, to $\emptyset_{\mathcal{T}}$.

The result of this definition is that every object of $\mathcal{C}$ is potentially polymorphic, with the capability of being of different types simultaneously. The discrimination will be the functor used to "watch" the object and find its type. If an object of type $a_{\mathcal{T}}$ is considered to be linked to a fault, it will also be of type $z_{\mathcal{T}}$. This is a major result from the fault polymorphism theorem.

Once it is established that elements linked to faults are inherently polymorphic, we can choose any pair of elements $x_{\mathcal{C}} \in \mathbf{Ob}(\mathcal{C})$ and $y_{\mathcal{C}} \in \mathbf{Ob}(\mathcal{C})$ of type $z_{\mathcal{T}} \in \mathbf{Ob}(\mathcal{T})$ (hence, $G_{\mathcal{CT}}\, x_{\mathcal{C}} = z_{\mathcal{T}}$ and $G_{\mathcal{CT}}\, y_{\mathcal{C}} = z_{\mathcal{T}}$) and trace them back to category $\mathcal{P}$, identifying $x_{\mathcal{P}} \in \mathbf{Ob}(\mathcal{P})$ and $y_{\mathcal{P}} \in \mathbf{Ob}(\mathcal{P})$. We can then study them to determine their common features. This is because, based on the fault polymorphism theorem, they are interchangeable in some contexts.

Controllers that refrain from sending some information will be provided with the relations found in $\mathcal{C}$ and $\mathcal{T}$ for a pair of objects $x_{\mathcal{P}} \in \mathbf{Ob}(\mathcal{P})$ and $y_{\mathcal{P}} \in \mathbf{Ob}(\mathcal{P})$. This enables the controllers to provide the additional knowledge necessary by the fault detection system to identify the aforementioned commonalities.

Thus, the polymorphic nature of faults justifies the need for adding new operations to controller interfaces to enable management solutions to retrieve additional information from particular elements under specific circumstances. The fault polymorphism theorem enables system controllers to ensure that data remain undisclosed unless a management solution provides evidence demonstrating that the elements about which it is requesting information are related to each other and to a fault. This condition minimizes the amount of data that is shared, reducing resource usage and potential leakage of metadata that could put the managed system at risk. ■

B. ALGORITHM

Using the definitions and theorems previously discussed, we define ANSA algorithm processes as shown in Algorithm 1 and Algorithm 2. Overall, E_m_ACTIVE retrieves configuration and monitoring information from the managed systems, E_m_PASSIVE receives requests for KOs and provides them, A_m_ACTIVE informs the monitoring processes of the metrics to measure, and A_m_PASSIVE constructs the NDT, analyzes it, and sends reports. Table 7 shows the relationships between the theorems and these algorithms.

The theorems reveal the processes executed by E_m, which is a monitoring element of the network service that the NDT represents—e.g., SDN controller, VIM. The algorithms reveal the processes executed by A_m, which is the management component that instantiates ANSA.

Thus, ANSA is formed by four processes: E_m active, E_m passive, A_m active, and A_m passive. On the one hand, the active process of E_m continuously runs a loop that initializes a KO, obtains the configuration and monitoring data of each element S_k of the network service it monitors S, links the KOs to the initialized KO, and sends the resulting KO to A_m. The passive process of E_m waits for requests of KOs. When it receives a description D_j of a KO to provide, a new KO is initialized. Then, for each element of S indicated by D_j, its configuration and/or monitoring data are obtained and joined to the KO. Finally, the KO is sent to A_m.

Algorithm 1 ANSA E_m processes—See Symbols in Table 6

```
loop: E_m_ACTIVE
    K_t ← INIT_KNOWLEDGE_OBJECT
    for all S_k ∈ S do
        K_t ← K_t ∪ GET_CONF(S_k)
        K_t ← K_t ∪ GET_MON_DATA(S_k)
    end for
    Send K_t to A_m
end loop:

loop: E_m_PASSIVE
    E_m receives D_j from A_m
    K_t ← INIT_KNOWLEDGE_OBJECT
    for all D_j^conf ∈ D_j do
        K_t ← K_t ∪ GET_CONF(D_j^conf)
    end for
    for all D_j^mon ∈ D_j do
        K_t ← K_t ∪ GET_MON_DATA(D_j^mon)
    end for
    Send K_t to A_m
end loop:
```

TABLE 6. Algorithm symbols.

Symbol	Definition
N	NDT
E_m	Monitoring element
K_t	Knowledge object (KO), tagged t
S	Network service (NS)
S_k	NS element
A_m	ANSA component
D_j	KO definition, ident j
D_j^{conf}	KO config def for j
D_j^{mon}	KO monit def for j
L	List of KO definitions
f_{ply}	Get KOs to update
f_{upd}	Updates NDT with KO
f_{HFR}	Raises faults in NDT
f_{typ}	Assigns types in NDT
G_F	Knowledge graph of faults

Algorithm 2 ANSA A_m Processes—See Symbols in Table 6

loop: A_m_ACTIVE
  $L \leftarrow N \circ f_{\text{ply}}$
  for all $D_j \in L$ **do**
    $E_m \leftarrow$ GET_MON_ELE(D_j)
    Send D_j to E_m
  end for
end loop:

loop: A_m_PASSIVE
  A_m receives K_t from E_m
  $N \leftarrow N \circ f_{\text{upd}}(K_t) \circ f_{\text{sta}}$
  GET_WHAT_IF(N, (PolyFit|ARIMA|MLP))
  $N \leftarrow N \circ f_{\text{HFR}}$
  $N \leftarrow N \circ f_{\text{typ}}$
  $G_F \leftarrow$ SELECT_FAULT_TYPED_OBJS(N)
  REPORT_FAULTS(G_F)
end loop:

TABLE 7. Theorem and algorithm map.

Theorem	Algorithm	Line/s	Details
III.1	2	11, 13	Used in f_{HFR}
III.2	2	11, 14	Used in f_{typ}
III.3	2	4, 15	Metric selection
III.4	2	2	Used in f_{ply}

On the other hand, the active process of A_m continuously applies Theorem 4 (fault polymorphism) to derive a list of descriptions of the minimum number of KOs that must be requested to update the NDT. For each description, its corresponding element E_m is obtained and the request D_j is sent to it. The response is then processed by A_m.

The passive process of A_m waits for a KO to be received, which is used to update the NDT and is stabilized following the incorporation of new knowledge. The updated NDT is then analyzed to obtain what-if scenarios via the projection method, which incorporates additional knowledge into the NDT; an update and stabilization procedure is inherently conducted. As mentioned earlier, the projection methods are PolyFit, ARIMA, and MLP, all of which have been tuned for the particular objects of projection, such as time-series. Then, a function that applies Theorem 1 (hidden fault raising) is used to identify all hidden faults in the NDT, and a function that applies Definition 3 and Theorem 2 (strong emergent typing) is used to assign the types to all objects. This allows the next function to select the objects considered fault-related and generate and send a report with the faults to other management system components.

A key invariant of all ANSA processes is that the NDT follows the monad laws and Definitions 1 and 2, which ensures the validity of results. The execution of the ANSA processes and the final report issued are essentially the proofs that verify the fault definition. This is a major result in our solution with the application of category theory. Moreover,

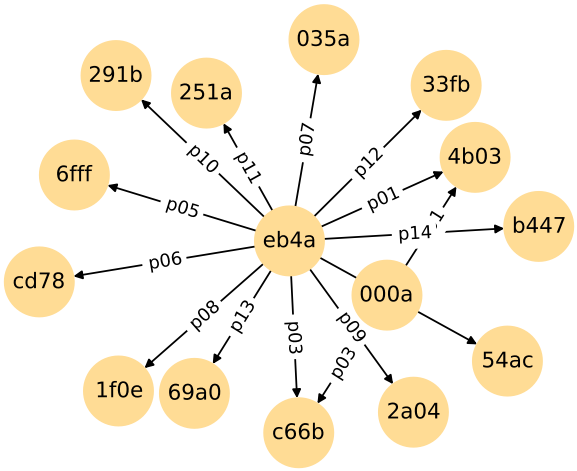


FIGURE 2. Knowledge graph contained by the NDT of the network service shown in Figs. 1 and 4. The vertices show element identifiers and the edges show the predicates that connect elements, as listed in Table 8.

by minimizing the KOs requested, ANSA minimizes the traffic in the control plane and the information disclosed by E_m, which is critical for remote sites.

Notably, because the ANSA algorithm is consistent with Theorem 3 (fault qualification), the stability of the results it obtains is uncoupled from the measurement procedure. In particular, it supports the methods used for retrieving information and knowledge while focusing on all types and not just faults. By contrast, when monitoring processes that target only data linked to faults will significantly reduce the quality of the information. This is supported by AI methods that are related to the results of this theorem.

C. PICTURE OF AN NDT

Next, we discuss how the proposed NDT operates. The NDT functions as a knowledge graph that evolves as a categorical object according to the monad laws and other constraints from Definitions 1, 2, and 3. We can query a state of the NDT to obtain a knowledge graph or, more specifically, to obtain a knowledge item collection from the monad structure. A portion of this type of graph is shown in Fig. 2. It shows some vertices and edges of the knowledge graph related to the same object, as extracted from the evaluation discussed in the following section. The vertices have the identifiers of the subject or object of knowledge item triples, and the edges show the predicate part, as indicated in Table 8.

D. KEY INNOVATIONS

The proposed solution has the following key innovations:

- It applies semantic techniques to define the KO structure, which incorporates all configuration and monitoring information made available to the monitoring element.
- It builds an NDT by applying semantic methods to available network service information (i.e., configuration and monitoring information) and maps the resulting structure to a computation structure by applying category, type, and information theories.

TABLE 8. Definitions of terms used in the edges of the NDT shown in Fig. 2. Each edge is labeled with a predicate name in the knowledge graph.

Edge	Predicate
p01	created_by_project_id
p02	created_by_user_id
p03	creator
p04	flavor
p05	image
p06	metrics.compute.instance.booting.time
p07	metrics.cpu
p08	metrics.cpu.delta
p09	metrics.cpu_l3_cache
p10	metrics.cpu_util
p11	metrics.disk.allocation
p12	metrics.disk.capacity
p13	project_id
p14	user_id

- It formalizes a fault in terms of semantic relations and attributes found in configurations and monitoring information, supporting the notion of the NDT structure as a proof of fault existence or absence.
- It applies category, type, and information theories to raise hidden relations between knowledge items and facilitate their study to detect faults, thereby supporting the hidden fault raising theorem.
- It applies category, type, and information theories, together with regression and ML-based prediction methods, to study what-if states of the NDT through feasible computations.
- It applies category, type, and information theories to ensure the analysis operations maintain high coherence and consistency; software theories are also applied to the analysis, which is a form of meta-softwarization.
- It defines a strong and consistent typing system for the KO category that, together with the monad laws, supports the ability to formally verify the resulting fault detection report; the NDT structure is the mathematical proof of the provided report and either validates a state change or exposes a fault.
- It adds information from a CVE report to the NDT.

These innovations strengthen current research trends in network management by assisting the application of semantic techniques and reasoning-focused mathematical theories, such as category, type, and information theories. These will support further innovations in solving network management problems.

E. SOTA ADVANCES

This study also contributes key advances to the SotA. These are summarized as follows:

- It defines the required elements for applying NDT methods to NA.
- It provides KO formalization, mathematization, conceptualization, and implementation.
- It proves that KOs constructed from monitored element states form categories, thereby demonstrating that KOs are categorical.
- It formulates and proves the hidden fault raising theorem.
- It proves that NDT computations can be restricted to operating a state monad instance. Therefore, they are comprehensive, complete, and deterministic computations and are appropriate for discrete decision processes such as MDP.
- It identifies strong and polymorphic types in the knowledge representation of network element states.
- It implements an ANSA prototype, which constructs an NDT of a real network service and applies the formulated theorems to detect the actual and potential exposed or hidden faults present in a real network service.

Existing SotA solutions for fault detection mainly lacks support for qualitative analysis and do not consider hidden faults, as is shown in Table 1 and discussed in Section II-A. The use of semantic techniques and NDT overcomes the lack of qualitative support, while our theorems and algorithm can be applied to enable hidden faults to be detected. This new SotA solution more adequately covers the requirements for fault detection.

IV. NETWORK SERVICE AUTOMATION SYSTEM

In this section, we discuss the components, interfaces, protocols, and data formats required to implement a SotA network service automation system, as defined by ETSI's NFV-MANO [6] and OSM [7]. We extend it to add a component that implements ANSA to enable the system to have fault detection functions. The architecture is shown in Fig. 3.

We first discuss the implementation of ANSA, which is as follows. First, the regression process applies regression and ML methods to historic values to obtain predictions of future values. Second, functions required by category, type, and information theories are processed. The +CVE process then incorporates CVE report information into the NDT, and the ontology/semantics process enforces semantic constraints (from ontology).

The output of these processes are sent to the NDT process, which computes f_{HFR}, f_{typ}, and what-if scenarios, with the input it receives. Finally, the fault reporting process produces reports about faults and posts them to the network service automation system for delivery to other interested parties or services.

Integrating ANSA into a network service automation system endows it with the ability to analyze network service configurations and state to detect faults. This ability is derived from the following internal processes in the ANSA component.

The other components of the network service automation system, as shown in Fig. 3, are as follows. First, we describe several OSM components because we propose to

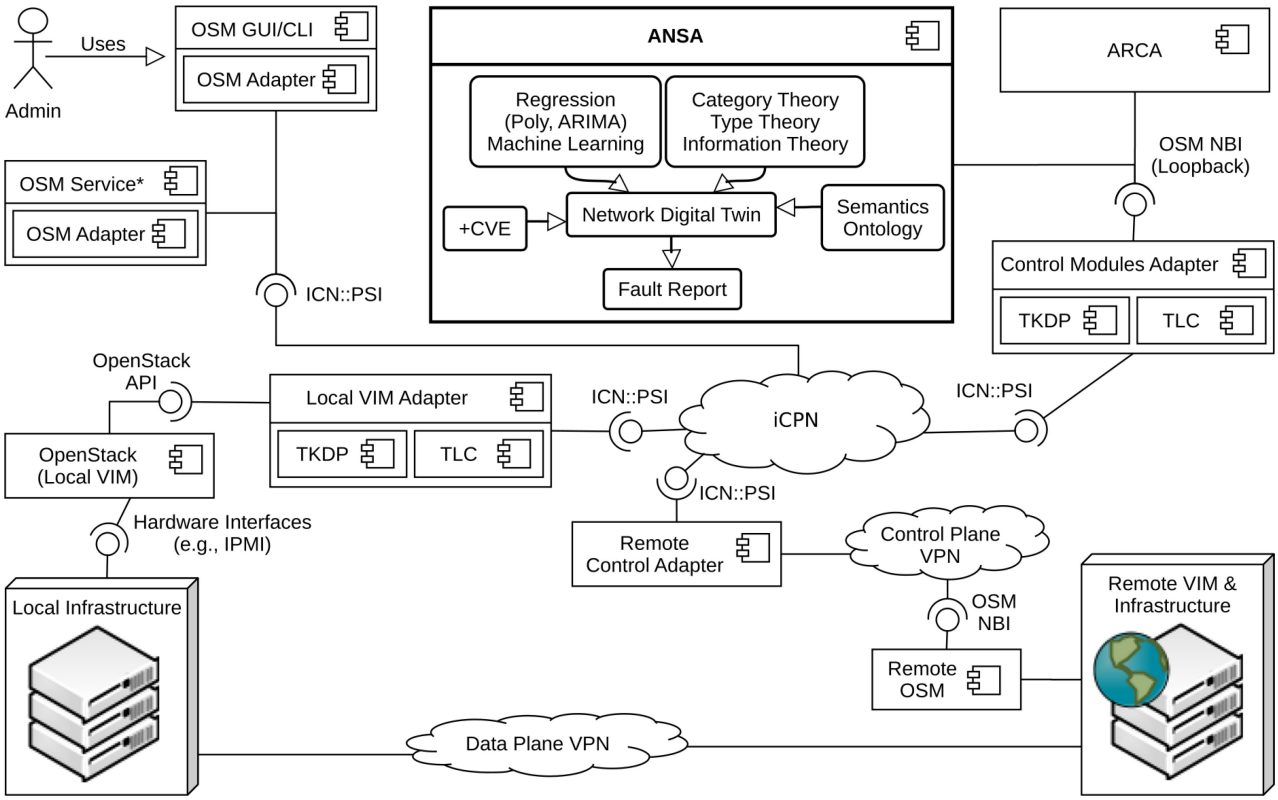


FIGURE 3. Overview of the network service automation system architecture.

interconnect with them directly. Second, we include remote OSM instances because administrative domain boundaries do not permit their underlying infrastructure to be controlled directly by local OSM. Third, we include other components needed to interconnect and control the managed system. These are described as follows.

iCPN [43] is a control plane network implemented as a service bus through a pub/sub protocol on information-centric networking (ICN) [44] to enable the seamless composition of management mechanisms, such as ANSA, to support their separation of concerns and achieve efficient many-to-many communications through name-based data transmissions, multi-destination delivery, and caching. iCPN qualities are particularly favorable for ANSA to receive required information and publish its reports without needing to specify their destinations one-by-one. ARCA [10] is a network control architecture that uses AI and ML algorithms to analyze the network service state and determine if some reactive or preventive change is required. ARCA and ANSA complement each other. ARCA decisions are communicated to ANSA in the form of KOs containing configuration and/or state knowledge. ANSA uses these decisions to validate ARCA decisions. ARCA receives ANSA reports and considers them together with other data to determine if corrective actions for the network service are needed.

OSM GUI/CLI represents both the Web-based graphical user interface and command line interface for administrative access. OSM Service* represents other OSM services: northbound interface (NBI), life-cycle management, monitoring, policy management, and resource orchestration. OpenStack is the VIM that manages the local underlying infrastructure. Remote OSM is the system that manages the remote infrastructure. The adapters enable transparent connection of existing components to iCPN. Telemetry knowledge distributed processing (TKDP) [45] is a component that enables delegating computing operations to data sources so that components can focus on their main concerns. Telemetry lossy/lossless compression (TLC) [46] is a component that enables lossy and lossless decompression/compression of monitoring data, information, and knowledge items. Lossy compression exploits the structure of information to achieve high compression rates with low degradation in fidelity.

Finally, we have the following infrastructure components. The control plane VPN enables interaction between local and remote management components. The data plane VPN enables interaction between local and remote VNFs and other data elements. The local infrastructure, namely, the computers, switches, and links are components managed by the local VIM. The remote VIM and associated infrastructure, namely, the OpenStack platform of the remote site and the computers, switches, and links it manages are additional components.

Telemetry operation in a network service automation system first monitors adapters that retrieve raw state and

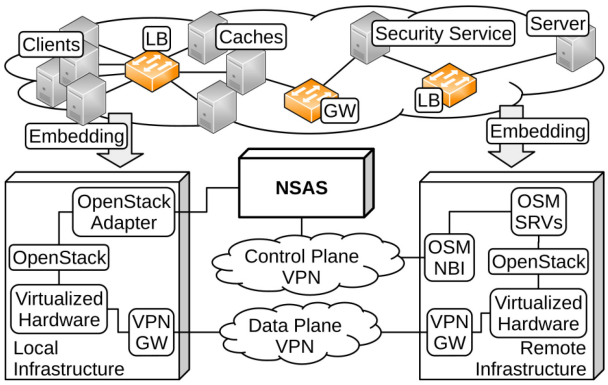


FIGURE 4. Experimentation scenario and topology of the overlying network service and underlying infrastructures. NSAS architecture, shown in Fig. 3 is located in the middle.

monitoring data from OpenStack and OSM. Then, data are processed with TKDP and/or TLC and transmitted to the control components. The monitoring plug-ins transform the raw state and monitoring data into KOs, potentially applying a processing function as requested by the control components and deliver the result to the network service automation system. The control plugins gather KOs and deliver them to ANSA and to other control components requesting them. In Section V-B, we discuss the interfaces, protocols, and data formats necessary to implement the network service automation system.

V. EVALUATION

To showcase and evaluate ANSA, we implemented a prototype of a network service automation system enabled with ANSA in a real platform. The prototype has all components shown in Fig. 3 and is instantiated with the required underlying infrastructures and services to construct the platform shown in Fig. 4. We use this platform to characterize ANSA.

We obtain evidence to showcase the overall operation of ANSA's algorithm and then characterize the following: 1) ANSA's suitability for fault detection using both qualitative and quantitative information, 2) the knowledge graph associated with the NDT that ANSA constructs, 3) ANSA performance, 4) ANSA effectiveness, 5) ANSA efficiency, 6) ANSA features compared to related studies, and 7) ANSA effectiveness as compared with other techniques.

A. APPLICATION SCENARIO

The scenario shown in Fig. 4 targets the automatic operation of a network service, which is deployed in multiple sites. The network service is constructed to support a Web service and endow it with caching and load balancing functions. A Web server is connected to one endpoint and four clients are connected to another point. The network service automation system adapts the number of caches to the requirements of the network service. The clients make requests to the server at a predefined rate, with the exception of one client that behaves differently to stress the network service. In this scenario, ANSA retrieves configuration and monitoring information about the managed network service from both sites through the network service automation system. It then analyzes the information and sends back the fault report to the network service automation system.

B. IMPLEMENTATION

The experimentation platform we constructed for the evaluation consists of a network service automation system and infrastructures from two sites interconnected by VPNs, each having a set of underlying physical machines, an OpenStack instance to manage them, and the required gateways (GWs) to implement the VPN. The local infrastructure also includes an OpenStack adapter connected to the network service automation system, allowing it to manage the local infrastructure directly.

The remote infrastructure has all components presented in Figure 3. However, ANSA is excluded, and the north-bound interface (NBI) is separated and connected to the network service automation system through the control plane VPN. This means that the network service automation system also manages the remote infrastructure, although not directly or exclusively. This is achieved by informing OSM services that some resources from the remote infrastructure are managed by the network service automation system through the NBI.

iCPN is used by all components to communicate with each other, regardless of whether they are reside the local or remote infrastructures. Elements then request iCPN to obtain required information by providing the information name (i.e., identifier). iCPN then provides it, caching and replicating the information if needed at the closest forwarding point to its destination. In particular, ANSA is enabled to obtain the KOs required at precise times. Each KO is identified by its name, derived from its definition and is given in D_j in Algorithm 1, Algorithm 2, and Table 6.

To implement ANSA algorithms, we define a strategy that applies category theory to each step required to transform the theoretical definition of the algorithm to the implementation code. This ensures that the resulting implementation has improved correction and determinism, regardless of the programming language used or programming skills of the programmer. We follow our implementation strategy to obtain an application coded in Python. Considering the trade-offs between using a purely functional programming language such as Haskell or a hybrid functional-object such as Python, we chose the latter because of the large number of libraries available for many tasks. Of these, we use some of the most common AI and ML libraries in our solution.

To implement our solution, we first formulate all computations, abstractions, and functions required by our algorithm using formal category theory constructions, which will eventually become Python expressions, structures, classes, and functions. We obtain functions to read configuration and monitoring data, to load the data in the NDT monad, to compute what-if scenarios and graph transformations, and

to derive graph representations. Finally, we compose the functions according to our algorithm to obtain the application for ANSA. It is worth mentioning that the adherence to category theory constructions is an implementation constraint that must be ensured—as we do in our implementation.

ANSA code is encapsulated in a microservice container—Docker style—and includes the elements needed to support all functions provided by iCPN. These include functions for TLC, TKDP, and KO handling. These functions enable ANSA to be seamlessly integrated into the network service automation system. Each component of the network service automation system shown in Fig. 3 is implemented in its own Docker-based container image. Both local and remote VNF instances and the network services they form are managed by ARCA and supervised by ANSA. Most components of ARCA are implemented in Python, although ARCA's reasoning engine is implemented in Java. OpenStack and other OSM services have their own separate platforms, which are accessed by their corresponding APIs.

The remaining elements of the network service automation system are defined as follows. ICN::PSI is the pub/sub interface to access iCPN. OSM NBI is a REST interface to access OSM. OpenStack API is a REST interface to access OpenStack. Hardware interfaces are used to access native hardware, local (e.g., running operating system interface) or remote (e.g., IPMI). These interfaces rely on the following protocols and data formats. The iCPN pub/sub interfaces work with the ICN protocol (i.e., CCNx protocol [47]) over TCP/IP. The REST/HTTP interfaces work over TCP/IP for remote connections and over loopback same-machine connections. The VPN uses SSL over TCP/IP to connect the local data and control planes with their remote counterparts. REST interfaces use JSON. CVE reports use the national vulnerability database (NVD) structure serialized in JSON. OpenStack state and monitoring data schemas are serialized in JSON. Finally, OSM state and monitoring data schemas are serialized in JSON.

Configuration and monitoring information is retrieved by OpenStack and OSM services and sent through iCPN to ANSA and ARCA, which analyze the information separately. The data collection procedure for ANSA is implemented as part of the local VIM and OSM adapters (i.e., microservice containers). The former uses the VIM API (i.e., OpenStack Python library) to retrieve configuration and state data from the VIM. The latter uses the OSM API (i.e., osmclient Python library) to retrieve configuration and state data from the OSM registry database. Both microservices use the library that manages the data model. Their outputs are lists of string triples that take the form: id, parameter-name, parameter-value.

C. EXPERIMENTS

We defined an experiment controller (EC) for executing the application scenario. It was used to construct the network service and to start or stop the clients following a defined pattern that invoked the participation of ARCA and altered the configuration of the network service. The EC induced both faulty and potential faulty states—i.e., induced an event that would eventually produce a faulty state if maintained over time.

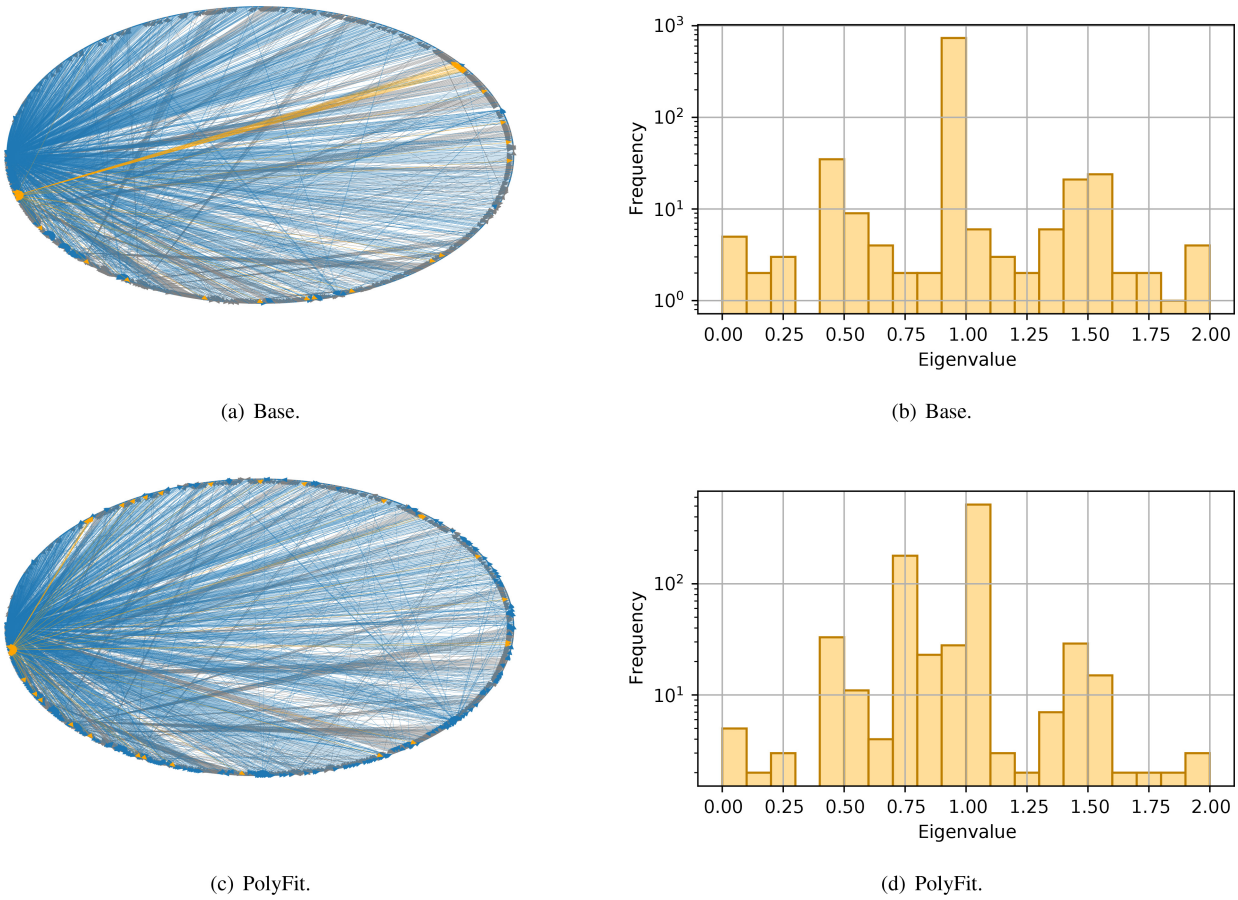
During the experiment, approximately 1800 faults were induced. In general, each fault is directly and indirectly linked to several knowledge items. For instance, detecting an erroneous value on a metric of a switch port marks the knowledge items that describe the port as faulty. These include the port identifier, the parent equipment to which it is connected, the metric associated with the port, and the measurement that yielded the erroneous value. In addition, as the knowledge items associated with the parent equipment are considered fault-related knowledge items, all other knowledge items associated with the parent equipment are considered fault-related knowledge items. Accordingly, our experiment produced more than 5000 fault-related knowledge items.

In our experiment, the EC collected the measurements required to achieve the evaluation targets. We recorded all the knowledge items and their relations that form the NDT obtained after each ANSA algorithmic step for the different simulation alternatives supported by the algorithm.

To evaluate ANSA's suitability for fault detection, the EC recorded the knowledge graph linked to the NDT in different stages of the experiment. To showcase the overall operation of the ANSA algorithm, the EC followed the evolution of a piece of knowledge related to an element affected by the induced fault (i.e., specifically, a virtual machine related to the network service) by identifying the edges connected to this piece of knowledge in the graphs representing each version of the NDT.

We applied spectral graph theory [39] to analyze and characterize the structure of the NDTs produced by ANSA methods, namely, the specialized knowledge graph enlarged with information from the what-if analyses. We characterized the knowledge graph of each version of the NDT by computing its normalized Laplacian graph spectrum. Although this method has a considerably complex run time and relatively high computational time, it is convenient for our purpose because the Laplacian matrix can be obtained from individual graphs; evaluating all possible pairs is not necessary, as with graph editing methods. Moreover, it allows graphs of different sizes to be compared ([40]).

To evaluate ANSA performance, the EC measured the size of the base NDT, the time needed to construct it, the size of the NDT resulting from each ANSA operation, and the time necessary for each operation. To evaluate ANSA effectiveness, the EC measured the number of fault-related items present in each version of the NDT, which are proportional to the faults discovered by each method. To evaluate ANSA efficiency, we computed the number of fault-related items obtained by each method per unit of time. To evaluate the overall ANSA features, we compared ANSA features with those in related studies [11], [12], [13]. To evaluate ANSA relative effectiveness in detecting faults



(a) Base. (b) Base.

(c) PolyFit. (d) PolyFit.

FIGURE 5. Knowledge graphs and eigenvalues of the base NDT and the NDT obtained using PolyFit.

(i.e., in comparison to related studies), we compared ANSA and [11] in terms of the size of the fault detection and solutions they obtained.

The experiment implemented the scenario shown in Fig. 4. Every component had its own dedicated server-type machine, and each machine had 12 cores, 24 threads, and ran at a peak speed of 2.93 GHz with 48 GiB of RAM and four 1-GE network interfaces. For independent results for ANSA, we executed ANSA on a separate machine with four cores, eight threads, running at a peak speed of 2.8 GHz with 16 GiB of RAM and a single 1-GE network interface. Because ANSA implementation uses totally parallelized underlying functions, all threads were used.

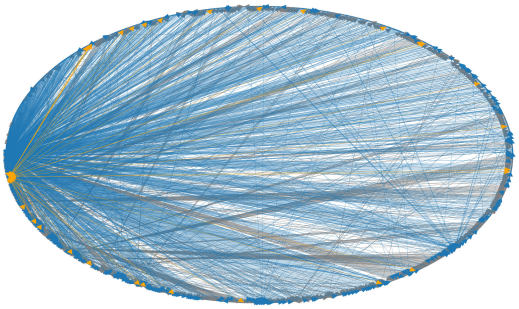
D. RESULTS

The knowledge graphs of each NDT are shown in Figs. 5 and 6. We include these graphs to assist in understanding the density of the graph that the NDT is managing and how it changes after each big what-if intervention. In particular, Figs. 5(a), 5(c), 6(a), and 6(c) capture the high degree of graph vertices representing fault-related knowledge items, which are either quantitative or qualitative. Thus, we observe that the number of edges changes following each operation of the ANSA algorithm, which corresponds to the simultaneous variation of many parameters of the NDT. The new edges are directly added as what-if simulation results as well as by the hidden fault raising procedure of the ANSA algorithm.

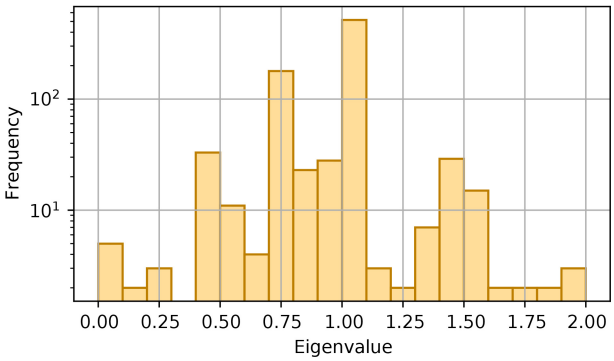
Following the highlighted edges on the resulting graphs representing the different versions of the NDT, which are connected to knowledge items related to single elements affected by the induced faults—in particular, virtual machines that are part of the network service—, we show that the number of edges increases with the additional knowledge provided by the what-if explorations. In addition, we show that some edges have been captured in the fault-only graph of ARIMA, which confirms the detection of the potential faults that would occur on time, as induced in our experiment.

Centrality is a means of analyzing complex graphs and is measured by watching the frequency of its eigenvalues. The manner in which a semantic graph representing an NDT is constructed results in a graph with few eigenvalues showing a high centrality, indicating that few paths are intertwined with each other. As new faults are detected, the centrality of other eigenvalues increases, meaning other paths become intertwined. Thus, analyzing the centrality is a way of knowing the number of faults identified in an NDT.

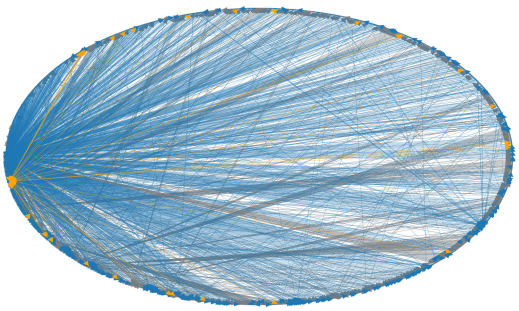
We conduct graph spectrum analysis to determine the centrality of each state of the NDT. This analysis yields a histogram for the graph representing each state of the NDT.



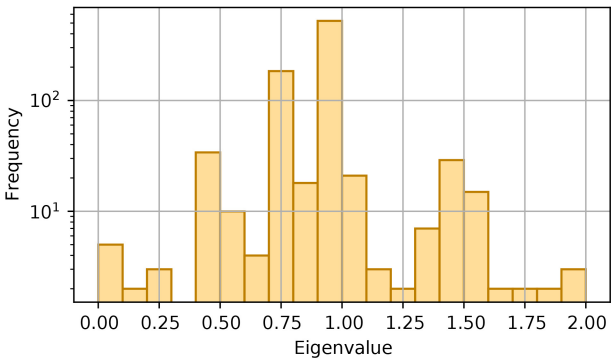
(a) ARIMA.



(b) ARIMA.



(c) MLP.



(d) MLP.

FIGURE 6. **Knowledge graphs and eigenvalues of NDTs obtained using ARIMA and MLP.**

Knowledge graphs are composed of semantic objects in the vertices and their relations in the edges. The amount of information contained in an NDT makes these graphs quite complex, but they are still standard graphs.

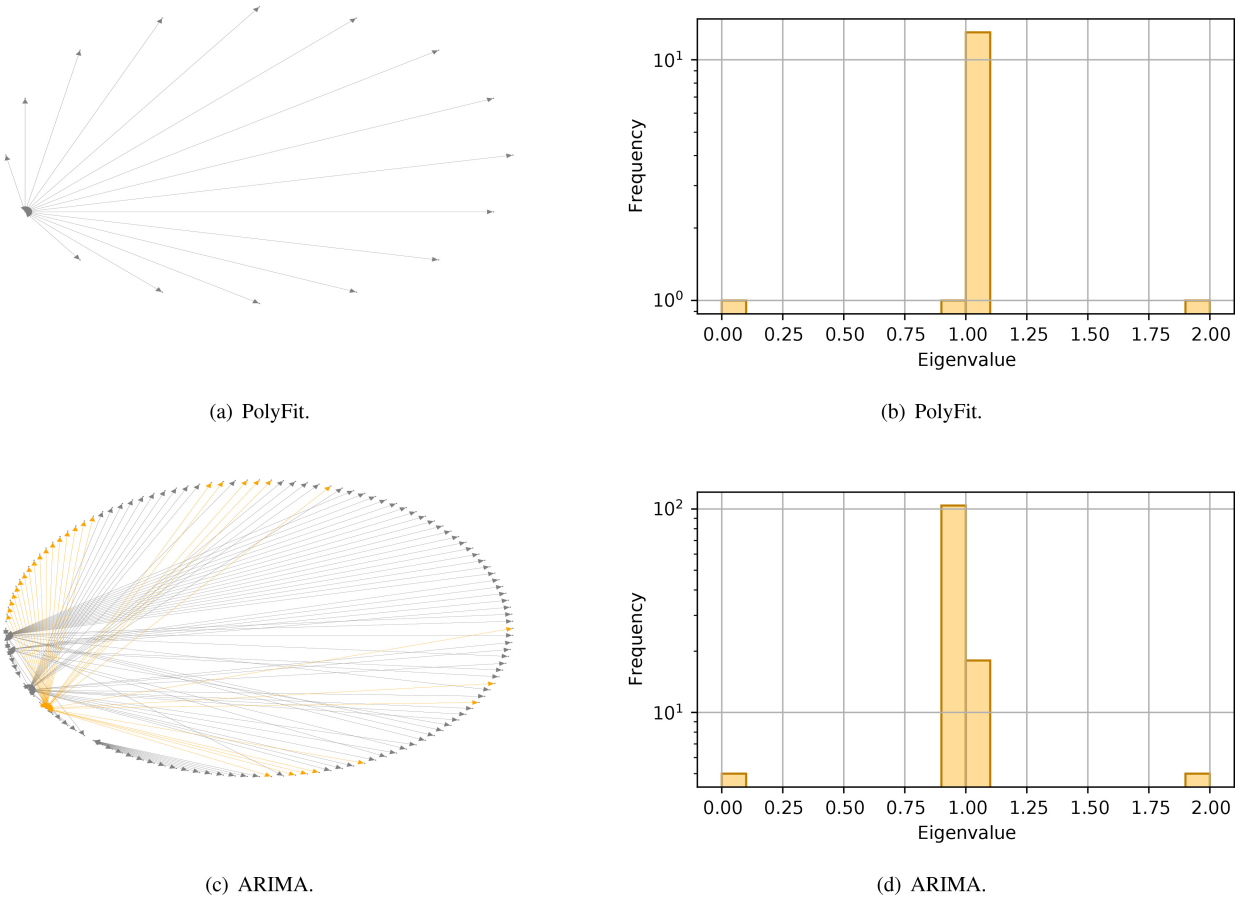
The distribution of the eigenvalues obtained from the knowledge graph spectrum of the base NDT is shown in Fig. 5(b). As there are few vertices with higher frequency, the centrality of the graph is considerably higher. In particular, the centrality of the vertices linked to the Laplacian eigenvalue (see Section II-A5) in the interval $[0.9, 1.1]$ is more than 10 times higher than the centrality of the vertices linked to the Laplacian eigenvalues in the interval $[0.4, 0.7]$ or $[1.3, 1.6]$. This is clearly shown in the concentration of the edges connected to the vertices on the left side of the graph illustrated in Fig. 5(a).

As Fig. 5(d) shows, the distribution of the eigenvalues obtained from the knowledge graph spectrum of the NDT with PolyFit predictions has many vertices whose centrality is relatively high. This is evidenced in the very high concentration of Laplacian eigenvalues in the interval $[0.7, 1.1]$ and by the existence of some spikes in the intervals $[0.4, 0.5]$ and $[1.4, 1.6]$. This behavior is reflected in the slight edge sparsity of the graph illustrated in Fig. 5(c).

Similarly, Fig. 6(b) shows that the distribution of the eigenvalues obtained from the knowledge graph spectrum of the NDT with ARIMA predictions also has many vertices whose centrality is relatively high. The same values obtained for PolyFit apply to the concentration of Laplacian eigenvalues. Thus, the same level of sparsity is seen in the graph illustrated in Fig. 6(a).

Deviating from the previous distributions, the distribution of the eigenvalues obtained from the knowledge graph spectrum of the NDT with MLP predictions shown in Fig. 6(d) reveals that more vertices have high centrality. The highly concentrated Laplacian eigenvalues are in the interval $[0.7, 1.1]$, and some spikes exist in the intervals $[0.4, 0.5]$ and $[1.4, 1.6]$. This behavior is reflected in the level of sparsity shown in Fig. 6(c).

Comparing the knowledge graphs and spectra of the base NDT and the NDT with PolyFit, ARIMA, and MLP predictions, as shown in Fig. 8(a), we find that the centrality of the NDT with predictions is higher than that of the base NDT. This shows that, following predictions, more vertices have been identified as sensitive, thus demonstrating and validating the operation of the what-if predictions and the ulterior application of the hidden fault raising theorem.



(a) PolyFit. (b) PolyFit.

(c) ARIMA. (d) ARIMA.

FIGURE 7. **Knowledge graphs and eigenvalues of the fault-related items obtained using PolyFit and ARIMA.**

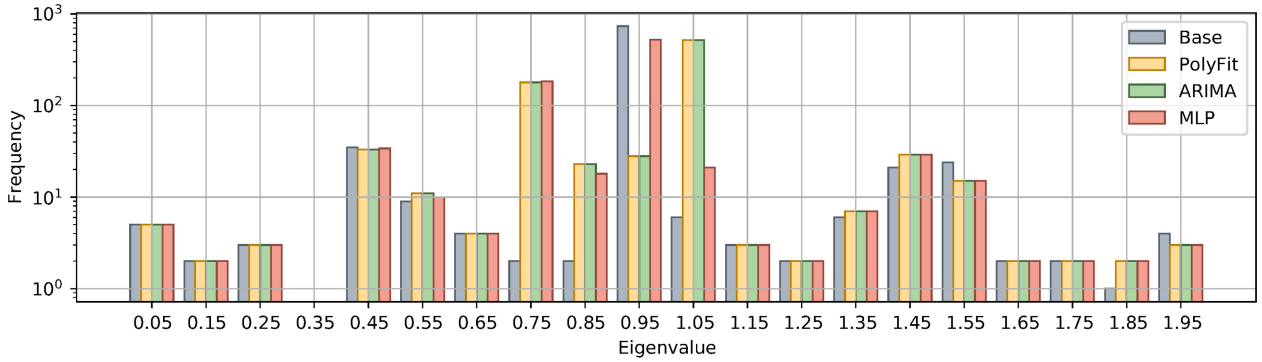
Comparing the knowledge graphs and spectra of the different NDTs produced after adding PolyFit, ARIMA, and MLP predictions, we see that, except for the interval [0.9, 1.1], they are nearly equal. For this interval, we see that the subintervals [0.9, 1.0] and [1.0, 1.1] are reversed from PolyFit to ARIMA and vice versa. These subintervals have similar shape in ARIMA and MLP, although in MLP we see that these subintervals are similar and so they express a similar centrality. In summary, these results mean that the methods are similar with only slight differences.

Regarding the structure of the knowledge graphs, shown in Figs. 5(a), 5(c), 6(a), and 6(c), we confirm that, although the graphs are very similar, their differences are still noticeable by focused sight. Particularly, comparing the graph that represents the NDT obtained after ARIMA and MLP processing, shown respectively in Figs. 6(a) and 6(c), we can see that the edges of the upper half are similar but the edges of the lower half are different. Such differences are emphasized when we extract the key edges, as shown in Figs. 7(a) and 7(c), which also show in different color the edges of the knowledge graph depicted Fig. 2.

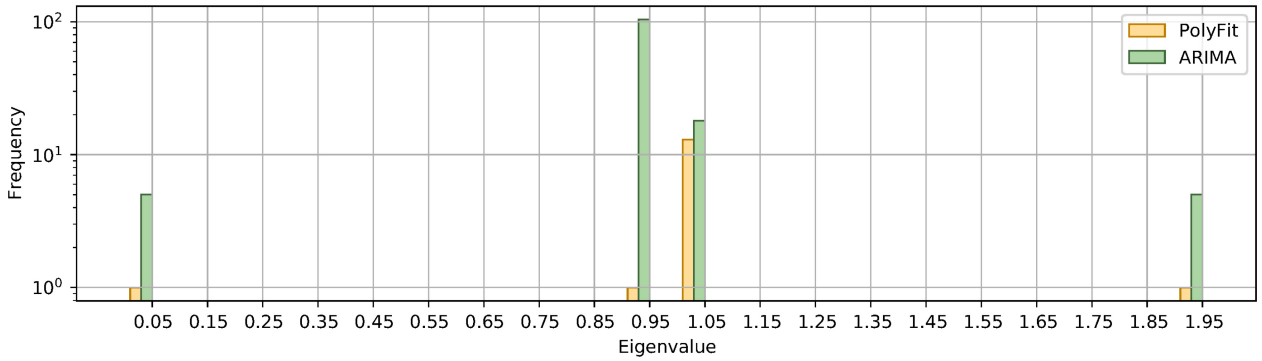
We also analized the knowledge graphs containing only fault-related edges of all edges representing the NDTs. Particularly, Fig. 7(b) shows the distribution of the eigenvalues obtained from the knowledge graph spectrum representing the isolated fault-related vertices and edges obtained after applying PolyFit to the base NDT, shown in Fig. 7(a). The graph shows that a vertex with very high centrality, where its eigenvalue interval is [1.0, 1.1]. This corresponds to one of the subintervals showing higher centrality in the NDT with PolyFit predictions, confirming the importance of this fault even though it was not present in the base NDT and thus was a hidden fault found by our algorithm.

Fig. 7(d) shows the distribution of the eigenvalues obtained from the knowledge graph spectrum that represents the isolated fault-related vertices and edges obtained after ARIMA is applied to the base NDT. The graph shows that a couple of vertices have very high centralities, where their eigenvalues fall in the interval [0.9, 1.1]. This corresponds to one of the subintervals showing higher centrality in the NDT with ARIMA predictions, confirming the importance of this fault even though it was not present in the base NDT and thus was a hidden fault found by our algorithm.

Comparing the knowledge graphs and spectra of the isolated fault-related NDT subgraphs, as shown in Fig. 8(b), we find that the ARIMA method detects the same faults detected by the PolyFit method as well as additional faults



(a) NDT graphs.



(b) Fault subgraphs.

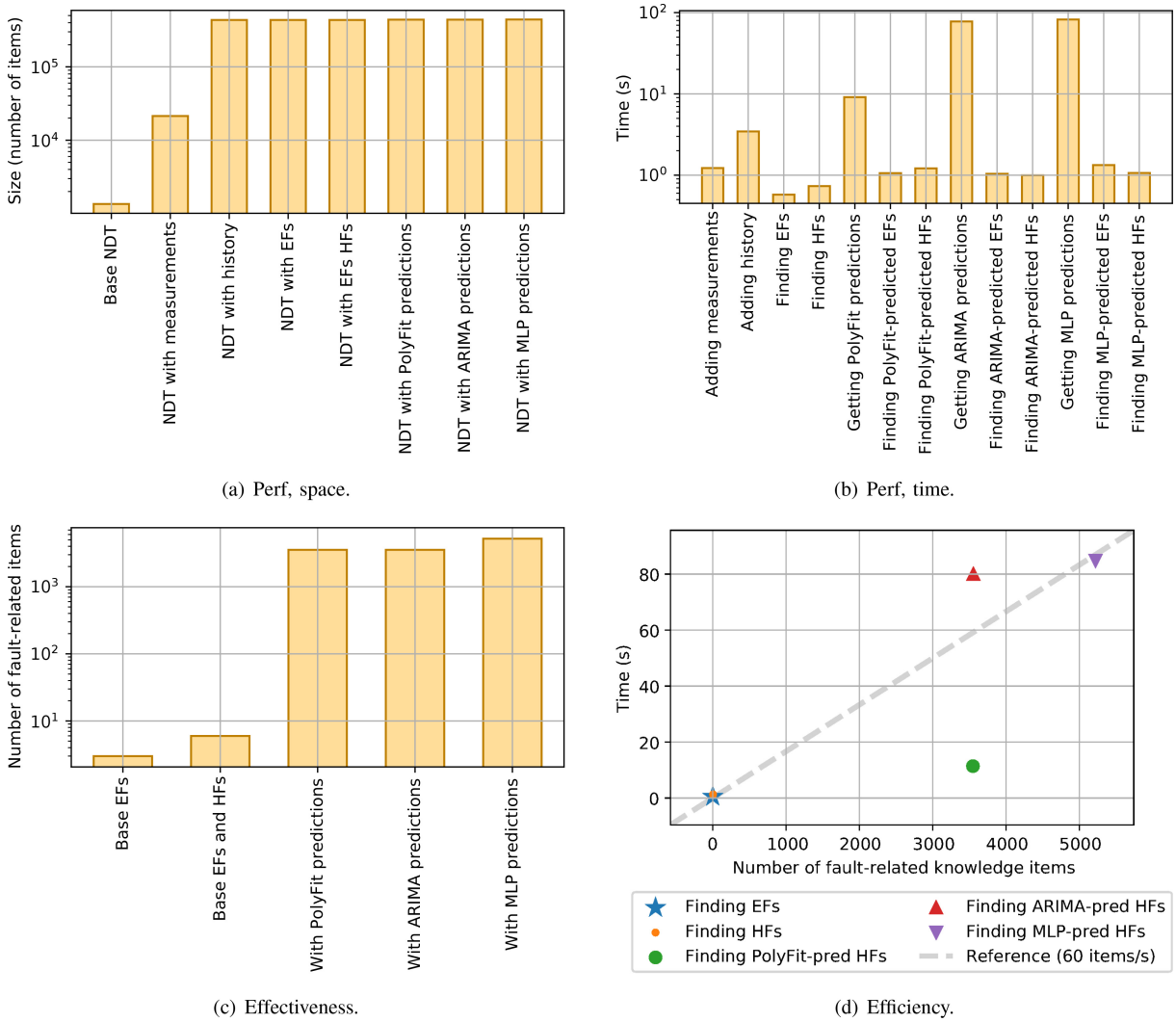
FIGURE 8. Eigenvalues of the whole NDT and fault-related graphs for all methods evaluated.

that have higher centrality, with their eigenvalues in the interval [0.9, 1.0]. This means that ARIMA is better at identifying fault-related vertices in similar circumstances. Although our solution identified many faults in the base NDT and in the NDT with MLP predictions, they did not have edges and thus were mostly isolated faults. This shows that our algorithm can detect faults with non-trivial paths in the NDT knowledge graph.

In terms of performance, as shown in Fig. 9(a), ANSA was able to construct a base NDT with state knowledge using no more than 10^4 knowledge items—of approximately 1 MiB—and could construct a complete NDT with history and endowed with different prediction structures using no more than 10^5 knowledge items—of approximately 10 MiB. On the other hand, as shown in Fig. 9(b), ANSA required approximately 5 s to construct a full NDT, including all history knowledge items. ANSA then required approximately 0.5 s and 1 s to execute the algorithm and to detect faults, respectively, in both the base NDT and prediction-enriched NDT; the fault detection process in the MLP-enriched NDT required slightly more than 1 s. The most time-hungry operations were those needed to obtain results for what-if simulations. The simplest of these operations, by means of PolyFit, required approximately 10 s, whereas the most complex operation required nearly 100.

We define ANSA's fault detection ability as the ratio of fault-related knowledge items that ANSA can identify when analyzing the NDT and the total fault-related knowledge items present in the system. A fault-related vertex represents a single parameter that is out of range, suspicious, or refers to an object that may not exist. During the experiment, as shown in Fig. 9(c), ANSA identified approximately 2 and 8 knowledge items that were linked to exposed or hidden faults, respectively, in the base NDT. After executing what-if exploration processes, ANSA identified considerably more fault-related knowledge items. In particular, for PolyFit and ARIMA, ANSA detected approximately 3500 fault-related knowledge items, whereas for MLP, it detected approximately 5000. This result showned that, during the execution of the experiment, ANSA detected a network congestion situation that ARCA could not detect, because it was due to a hidden fault that required the analysis of what-if situations to be detected.

We validated the efficiency of ANSA's internal mechanisms used in what-if simulations as functions of the relationships between their computation time and the number of identified fault-related knowledge items. As shown in Fig. 9(d), we found that the efficiency of ANSA when using MLP was somewhat balanced, consistent with a base of 60 fault-related knowledge items detected per second. When using PolyFit, ANSA requires considerably less time.



(a) Perf, space.

(b) Perf, time.

(c) Effectiveness.

(d) Efficiency.

FIGURE 9. NDT and fault detection performance and effectiveness results.

However, 30% of the fault-related knowledge items were not identified. The worst case involved ARIMA, which obtained similar results to those of PolyFit but required nearly the same amount of computation time as MLP.

ANSA features as compared with those in related studies are shown in Table 9, which shows that the major features of ANSA is its support of qualitative features through semantics and knowledge graphs and associated categorization, although it uses more memory than the other solutions. Other features of ANSA are somewhat adaptable, such as using an optimization targets. The main benefits of the comparative methods are their use of stochastic simulation methods and more advanced ML methods. We explore them in a future work.

Fig. 10 compares the related studies in terms of performance qualities. The units are within 0–4 and they are related to the specifications of the machine required to achieve feasible performance results. For instance, 0 means that a very-high-spec machine is required to achieve claimed performance whereas 4 means that a very-low-spec machine is enough. The figure shows that, although all solutions exhibited good performance, ANSA offer the highest performance, achieving the best results based on its low computational complexity. It achieves the best results in terms fidelity because the NDT mimics perfectly real environments and their states through continuous monitoring and incorporation of the NDT. It also achieves the best flexibility results because the NDT and its application of category theory enable ANSA to incorporate new metrics, configuration parameters, and other information into the analysis without having to re-implement the solution. The major drawback is that ANSA uses more memory to maintain the NDT structures. We intend to improve this in future work.

Fig. 11 compares the methods proposed in [11] (RMH and MALA) with ANSA. Although RMH and MALA show that the number of faults for different structures follows a long-tail distribution, the number of faults obtained by

TABLE 9. Qualitative comparison between related studies and ANSA.

Aspect	[11]	[12]	[13]	ANSA
Prediction system	Differentiable simulation	Deep learning	Monte Carlo simulation	DT what-if exploration, regression, AI/ML/NN
Iteration goal	Optimization	Matching	Optimization	Policy compliance
Sampling	Gradient-based	Bayesian posterior	Adaptive importance	Guided by knowledge relations
Objects	Numeric	Numeric	Numeric	Knowledge items
Relations	Functional	Functional	Functional	Categorical
Operations	Math	Math	Math	Semantics
Data structure	Matrix	Matrix	Matrix	Knowledge graph
Qualitative	–	–	Learning interpretation	Integrated
Memory footprint	Low	Low	Low	High
CPU usage	High	Medium	Medium	Low
Closed loop	No	No	No	DT-based
Real plat. impl.	Partial	No	No	Yes

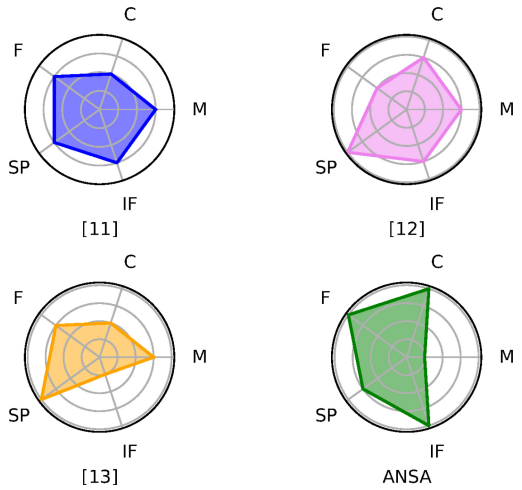


FIGURE 10. Qualitative comparison of studied solutions, where M = memory; C = CPU; F = fidelity (similarity of analyzed samples to reality); SP = support for different types of scenario predictions; IF = flexibility in specifying iteration goals. Higher values are better.

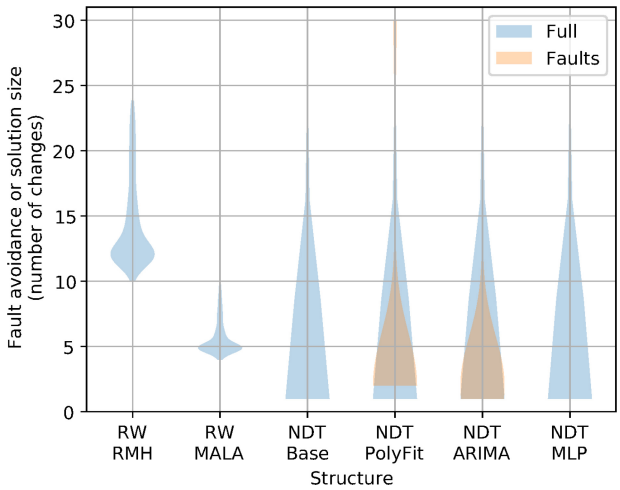


FIGURE 11. Comparison of the number of faults detected by the two strategies proposed by [11] and the four ANSA NDTs. The plot width in the horizontal axis corresponds with the number of faults for each solution size. Solution size in the vertical axis is measured as the number of changes required to implement a solution. The fault plot considers only the substructures with direct faulty vertices, ignoring indirect faulty vertices.

ANSA follows a thick-tail distribution (i.e., has a pyramid-type shape), with much higher concentration in lower zones of the spectrum. The distribution of structures of faulty vertices denotes the same distribution. In general, this shows that our solution detects many more faults. For instance, for size 5, RMH found no fault, MALA detected approximately 500 faults, and ANSA detected more than 1500 faults. For size 13, RMH detected approximately 500 faults, MALA approximately 1000 faults, and ANSA approximately 1600 faults. Overall, dividing the number of faults each solution finds by the total of 1600 faults induced in the experiment, we get that ANSA detected an average of 94% of the faults present in a network service, whereas previous SotA solutions only detected 30% to 50% of faults.

VI. CONCLUSION AND FUTURE WORK

In this study, we showcased the benefits of 1) using an NDT for analyzing the states of a network service, 2) using category theory for constructing the NDT, and 3) applying our theorems to fault detection, particularly hidden fault detection. We demonstrated that ANSA can consider network states not considered by other analytical techniques, detecting an average of 94% of faults present in a network service. In comparison, previous SotA solutions can detect only 30%–50% of faults. Furthermore, ANSA allows for new possibilities for analyzing network services and

TABLE 10. Summary of ANSA outcomes evaluation.

Feature	Achievement
Fault detection using both qualitative and quantitative information	The NDT's kdnowledge graph includes both, as shown in Fig. 2.
Knowledge graph associated with the NDT that ANSA constructs	Evaluated using spectral graph analysis, summarized in Fig. 8.
Performance	ANSA uses approximately 10 MiB and requires approximately 10 s for simple analysis methods and less than 100 s for complex analysis methods, as shown in Figs. 9(a) and 9(b).
Effectiveness	ANSA detects an average of 94% of the faults, as shown in Figs. 9(c) and 11.
Efficiency	ANSA detects an average of 60 fault-related knowledge items per second, as shown in Fig. 9(d).
Features compared to related studies	ANSA outstands in the way faults are targeted and being implemented in a real platform, as shown in Table 9.
Effectiveness as compared with other techniques	ANSA detects 94% of the faults, while previous SotA solutions detect only 30%–50%, as shown in Fig. 11.

constructing NDTs. In particular, ANSA can support intent-based networking scenarios, in which fault detection and exploration of what-if scenarios are particularly important.

In a future research, we will use ANSA to validate the outputs of an intent translation engine. The procedure is mostly the same, although it requires support in constructing an NDT with minimal or no telemetry. Therefore, the simulations must be based solely on projecting rules, an aspect to be thoroughly examined. We will provide input to standardization initiatives, which will be accompanied by an open-source solution that will include ANSA connected to OSM. Furthermore, we will add support for using rules when projecting NDT properties to explore what-if scenarios. The additional rules will be linked to the semantics of each property. We will also study the use of a complex event processor to achieve efficient detection of complex patterns of knowledge introduced into the NDT. Finally, we will integrate additional AI methods into ANSA, such as reinforcement learning and planning algorithms that exploit the structure of knowledge and its categories, as constructed and maintained by ANSA. We will also explore the application of collaborative, multidomain AI models that analyze local and global aspects of network services.

REFERENCES

[1] T. Faisal, J. A. O. Lucena, D. R. Lopez, C. Wang, and M. Dohler, "How to design autonomous service level agreements for 6G," *IEEE Commun. Mag.*, vol. 61, no. 3, pp. 80–85, Mar. 2023.

[2] A. Galis et al., "Softwarization of future networks and services-programmable enabled networks as next generation software defined networks," in *Proc. IEEE SDN Future Netw. Services (SDN4FNS)*, Washington, DC, USA, 2013, pp. 1–7.

[3] P. Martinez-Julia, A. F. Skarmeta, and A. Galis, "Towards a secure network virtualization architecture for the future Internet," in *The Future Internet* (Lecture Notes in Computer Science 7858). Berlin, Germany: Springer, 2013, pp. 141–152.

[4] D. R. Lopez, "Network functions virtualization: Beyond carrier-grade clouds," in *Proc. Opt. Fiber Commun. Conf. Exhibit. (OFC)*, Washington, DC, USA, 2014, pp. 1–18.

[5] L. Contreras, Ó. González, V. López, J. Fernández-Palacios, and J. Folgueira, "iFUSION: Standards-based SDN architecture for carrier transport network," in *Proc. IEEE Conf. Stand. Commun. Netw. (CSCN)*, 2019, pp. 1–7.

[6] "Network functions virtualisation (NFV); management and orchestration," ETSI, Sophia Antipolis, France, document ETSI GS NFV-MAN 001, 2014.

[7] "OSM release five technical overview," OSM, ETSI, Sophia Antipolis, France, White Paper, 2019,

[8] "The OpenStack project." 2017. [Online]. Available: http://www.openstack.org/

[9] N. McKeown et al., "OpenFlow: Enabling innovation in campus networks," in *Proc. SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[10] P. Martinez-Julia, V. P. Kafle, and H. Harai, "Exploiting external events for resource adaptation in virtual computer and network systems," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 2, pp. 555–566, Jun. 2018.

[11] C. Dawson and C. Fan, "A Bayesian approach to breaking things: Efficiently predicting and repairing failure modes via sampling," 2023, *arXiv:2309.08052*.

[12] Y. Zhou, S. Booth, N. Figueroa, and J. Shah, "RoCUS: Robot controller understanding via sampling," in *Proc. Conf. Robot Learn.*, 2022, pp. 850–860.

[13] M. O'Kelly, A. Sinha, H. Namkoong, R. Tedrake, and J. C. Duchi, "Scalable end-to-end autonomous vehicle testing via rare-event simulation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–12.

[14] S. Yadav and T. Poongodi, "A review of ML based fault detection algorithms in WSNs," in *Proc. 2nd Int. Conf. Intell. Eng. Manage. (ICIEM)*, 2021, pp. 615–618.

[15] H. Huang, L. Zhao, H. Huang, and S. Guo, "Machine fault detection for intelligent self-driving networks," *IEEE Commun. Mag.*, vol. 58, no. 1, pp. 40–46, Jan. 2020.

[16] N. Wang, J. Wang, and X. Chen, "A trust-based formal model for fault detection in wireless sensor networks," *Sensors*, vol. 19, no. 8, p. 1916, 2019.

[17] M. M. Gharamaleki and S. Babaie, "A new distributed fault detection method for wireless sensor networks," *IEEE Syst. J.*, vol. 14, no. 4, pp. 4883–4890, Dec. 2020.

[18] S. Cherrared, S. Imadali, E. Fabre, and G. Gossler, "SFC self-modeling and active diagnosis," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 3, pp. 2515–2530, Sep. 2021.

[19] R. Yarinezhad and S. Hashemi, "Distributed faulty node detection and recovery scheme for wireless sensor networks using cellular learning automata," *Wireless Netw.*, vol. 25, pp. 2901–2917, Jul. 2019.

[20] M. Silva, A. Pacini, A. Sgambelluri, and L. Valcarenghi, "Learning long- and short-term temporal patterns for ML-driven fault management in optical communication networks," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 3, pp. 2195–2206, Sep. 2022.

[21] W. Zhang, J. Wang, G. Han, S. Huang, Y. Feng, and L. Shu, "A data set accuracy weighted random forest algorithm for IoT fault detection based on edge computing and blockchain," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2354–2363, Feb. 2021.

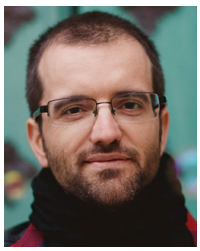
[22] P. Biswas, R. Charitha, S. Gavel, and A. S. Raghuvanshi, "Fault detection using hybrid of KF-ELM for wireless sensor networks," in *Proc. 3rd Int. Conf. Trends Electron. Inform. (ICOEI)*, 2019, pp. 746–750.

[23] G. Kaur and P. Chanak, "An energy aware intelligent fault detection scheme for IoT-enabled WSNs," *IEEE Sensors J.*, vol. 22, no. 5, pp. 4722–4731, Mar. 2022.

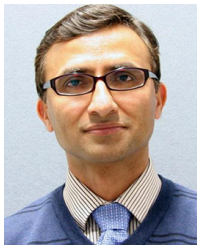
[24] A. Laiou, C. M. Malliou, S.-A. Lenas, and V. Tsaoussidis, "Autonomous fault detection and diagnosis in wireless sensor networks using decision trees," *J. Commun.*, vol. 14, pp. 544–552, Jul. 2019.

[25] L. Tan, W. Su, W. Zhang, H. Shi, J. Miao, and P. Manzanares-Lopez, "A packet loss monitoring system for in-band network telemetry: Detection, Localization, diagnosis and recovery," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 4, pp. 4151–4168, Dec. 2021.

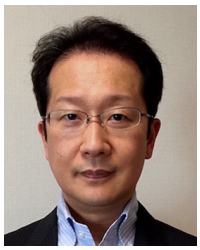
[26] B. Milewski, *Category Theory for Programmers*. San Francisco, CA, USA: Blurb, 2018.
[27] S. M. Y. Kinoshita and N. Yoneda, "The Yoneda lemma," *Math. Japonica*, vol. 47, no. 1, pp. 155–156, 1998.
[28] H. Huang et al., "Real-time fault detection for IIoT facilities using GBRBM-based DNN," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5713–5722, Jul. 2020.
[29] L. Velasco, S. Barzegar, and M. Ruiz, "Using a SNR digital twin for failure management," in *Proc. 23rd Int. Conf. Transp. Opt. Netw. (ICTON)*, 2023, pp. 1–4.
[30] H. Nan, R. Li, X. Zhu, J. Ma, and D. Niyato, "An efficient data-driven traffic prediction framework for network digital twin," *IEEE Netw.*, vol. 38, no. 1, pp. 22–29, Jan. 2024.
[31] P. Almasan et al., "Network digital twin: Context, enabling technologies, and opportunities," *IEEE Commun. Mag.*, vol. 60, no. 11, pp. 22–27, Nov. 2022.
[32] S. Barzegar, M. Ruiz, A. Sgambelluri, F. Cugini, A. Napoli, and L. Velasco, "Soft-failure detection, localization, identification, and severity prediction by estimating QoT model input parameters," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 3, pp. 2627–2640, Sep. 2021.
[33] Q. Zhuge et al., "Building a digital twin for intelligent optical networks," *J. Opt. Commun. Netw.*, vol. 15, no. 8, pp. C242–C262, Aug. 2023.
[34] B. Fong and D. I. Spivak, "Seven sketches in compositionality: An invitation to applied category theory," 2018, *arXiv:1803.05316*.
[35] D. Borsatti, W. Cerroni, and S. Clayman, "From category theory to functional programming: A formal representation of intent," in *Proc. IEEE 8th Int. Conf. Netw. Softw. (NetSoft)*, 2022, pp. 31–36.
[36] E. Rijke, "Introduction to homotopy type theory," 2022, *arXiv:2212.11082*.
[37] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Hoboken, NJ, USA: Wiley, 2005.
[38] Y. Zhu, D. Chen, C. Zhou, L. Lu, and X. Duan, "A knowledge graph based construction method for digital twin network," in *Proc. IEEE 1st Int. Conf. Digit. Twins Parallel Intell. (DTPI)*, 2021, pp. 362–365.
[39] B. Nica, *A Brief Introduction to Spectral Graph Theory*. Helsinki, Finland: Eur. Math. Soc., 2018.
[40] P. Wills and F. G. Meyer, "Metrics for graph comparison: A practitioner's guide," *PLoS ONE*, vol. 15, no. 2, 2020, Art. no. e0228728.
[41] H. Zhao and Z. Lai, "Neighborhood preserving neural network for fault detection," *Neural Netw.*, vol. 109, pp. 6–18, Jan. 2019.
[42] H. Zhao, Z. Lai, and Y. Chen, "Global-and-local-structure-based neural network for fault detection," *Neural Netw.*, vol. 118, pp. 43–53, Oct. 2019.
[43] P. Martinez-Julia, V. P. Kafle, and H. Asaeda, "iCPN: Scalable control plane for the network service automation system," in *Proc. IFIP/IEEE Netw. Oper. Manage. Symp. (NOMS)*. Washington, DC, USA, 2024, pp. 1–5.
[44] H. Asaeda, K. Matsuzono, Y. Hayamizu, H. H. Hlaing, and A. Ooka, "A survey of information-centric networking: The quest for innovation," *IEICE Trans. Commun.*, vol. 107, no. 1, pp. 139–153, 2024.
[45] P. Martinez-Julia, V. P. Kafle, and H. Asaeda, "Telemetry knowledge distributed processing for network digital twins and network resilience," in *Proc. IFIP/IEEE Netw. Oper. Manage. Symp. (NOMS)*, Washington, DC, USA, 2023, pp. 1–6.
[46] P. Martinez-Julia, V. P. Kafle, and H. Asaeda, "A novel telemetry compression method for enhancing network resilience," in *Proc. 26th Conf. Innov. Clouds, Internet Netw. (ICIN)*, 2023, pp. 174–178.
[47] M. Mosko, I. Solis, and C. Wood, "Content-centric networking (CCNx) messages in TLV format," Internet Eng. Task Force, RFC 8609, 2019. [Online]. Available: https://rfc-editor.org/rfc/rfc8609.txt



PEDRO MARTINEZ-JULIA (Member, IEEE) received the B.S. degree in computer science from the Open University of Catalonia, the M.S. degree in advanced information technology and telematics, and the Ph.D. degree in computer science from the University of Murcia, Spain. He is currently a Full-Time Researcher with the National Institute of Information and Communications Technology, Tokyo. His main expertise is in network architecture, control, and management, with a particular interest in overlay networks and distributed systems and services. He has been involved in EU-funded Research Projects since 2009, leading several tasks/activities, and has participated in IETF/IRTF for the standardization of new network technologies. He has published more than 20 papers in conferences and journals. He is a member of ACM.



VED P. KAFLE (Senior Member, IEEE) received the B.E. degree in electronics and electrical communications from the Punjab Engineering College (currently the PEC University of Technology), India, the M.S. degree in computer science and engineering from Seoul National University, South Korea, and the Ph.D. degree in informatics from the Graduate University for Advanced Studies, Japan. He is currently a Senior Researcher with the National Institute of Information and Communications Technology, Tokyo, and concurrently holds Visiting Associate Professor position with the University of Electro-Communications, Tokyo. He has been serving as a Co-Rapporteur of ITU-T Study Group 13 since 2014. His research interests include new network architectures, naming and addressing, machine-to-machine communication, IoT, and privacy, and security management in networks. He received the ITU Association of Japan's Encouragement Award in 2009 and the Accomplishment Award in 2017, and the two Best Paper Awards (second prize) at the ITU Kaleidoscope Academic Conferences in 2009 and 2014. He is a member of IEICE.



HITOSHI ASAEDA (Senior Member, IEEE) received the Ph.D. degree from Keio University. He is the Director of the Network Architecture Laboratory, National Institute of Information and Communications Technology and is also a Collaborative Professor with the Graduate School of Informatics and Engineering, University of Electro-Communications. He was previously with IBM Japan, Ltd., and was a Research Engineer specialist with INRIA Sophia Antipolis, France. His research interests include ICN, network coding, high-quality streaming, and large-scale testbeds. He received the IEICE Communications Society Outstanding Contributions Award in 2019. He was a Project Associate Professor with Keio University from 2005 to 2012. He was a Guest Editor-in-Chief of the Special Series of *IEICE Transactions on Communications* in 2016. He was the Chair of the IEICE Technical Committee on ICN from 2017 to 2019. He served as the General Chair of IEEE/ACM IWQoS 2021 and ACM ICN 2022, and has been a TPC member for premier conferences, such as IEEE Infocom, WCNC, and ACM ICN. He is a Fellow of IEICE, and a member of ACM.