Exploring Semantic-Enhanced Property Graphs in Network Operations

Beyza Yaman[1,*], Michael Mackey[2], Peter Cautley[2] and Declan O'Sullivan[1]

[1]ADAPT Centre, Trinity College Dublin, Dublin, Ireland
[2]Huawei Ireland Research Centre, Dublin, Ireland

Abstract
Telecommunication networks face significant challenges in performing root cause analysis due to their complexity, data heterogeneity, and high-volume event streams. Traditional semantic models like RDF and flexible property graph models such as LPG offer partial solutions but fall short when applied individually. This paper investigates the Semantic-enhanced Programmable Graph (SPG) approach, specifically through the OpenSPG engine, which integrates the structural advantages of property graphs with semantic constraints inspired by RDF ontologies. We evaluate its applicability in a telecom root cause analysis use case, comparing it to RDF-based modeling. Our findings investigate the applicability of OpenSPG in operational modeling and its limitations in maturity and tooling, offering insights for future semantic-enabled network management solutions.

Keywords
ontology, schema, property graphs, semantic enhanced property graphs

1. Challenges of Root Cause Analysis in the Network Operation Systems

In complex systems such as telecommunications networks, maintaining operational reliability requires not only detecting issues early but also understanding their underlying causes. Root cause analysis (RCA) in the telecom domain faces several challenges due to the scale, complexity, and heterogeneity of modern network infrastructures[1]. Telecom networks generate significant volumes of event data from diverse and distributed sources, often in inconsistent formats and without unified semantics. Events are highly interdependent, with cascading effects across different layers (e.g., management, control, and data), making it difficult to isolate the origin of faults. Correlating this data across different sources, understanding service and customer impact, and preserving contextual information such as interface roles or link functions are particularly difficult. Additionally, the dynamic and real-time nature of network operations requires RCA methods that are not only accurate but also scalable and responsive. Integrating domain knowledge and semantic context remains a key challenge for building explainable and automated RCA systems in telecom. Addressing these limitations demands graph-based approaches that can effectively represent event dependencies and domain knowledge.

In the domain of graph-based knowledge representation, two primary models are RDF [1] and Labeled Property Graphs (LPG) [2], which differ in their trade-offs between semantic richness and performance [3]. Building on this foundation, recent research has explored applying such semantically enriched graph models to real-world network management scenarios, highlighting the growing interest in telecom operations. Martínez *et al.* [4] introduced the YANG Server Ontology to model YANG-based network servers and their NETCONF interactions, providing integration guidelines for RDF Mapping Language (RML) mappings[2]. They also proposed CANDIL [5], a federated data fabric that unifies heterogeneous

Posters, Demos, and Industry Tracks at ISWC 2025
*Corresponding author.
beyza.yaman@adaptcentre.ie (B. Yaman); michael.mackey@huawei.ie (M. Mackey); peter.cautley@huawei.ie (P. Cautley); declan.osullivan@adaptcentre.ie (D. O'Sullivan)
0000-0002-0877-7063 (B. Yaman); 0000-0002-9421-8566 (D. O'Sullivan)

[1]https://datatracker.ietf.org/doc/html/draft-mackey-nmop-kg-for-netops-02#name-data-overload-from-network-
[2]https://rml.io/specs/rml/

network data into a knowledge graph using shared ontologies, validated through Edge-Cloud analytics use cases. The NORIA-O model [6] supports representing network infrastructures and events, though its generality poses challenges for consistent multi-domain integration.

RDF enables reasoning and interoperability through subject–predicate–object triples and ontologies, while LPG provides flexibility and scalability with key-value properties on nodes and edges. However, RDF's query inefficiencies [7] and LPG's lack of formal semantics limit their standalone applicability in domains like telecom. Recently, a hybrid approach, namely Semantic-enhanced Programmable Knowledge Graphs (SPG), has been proposed to bridge this gap [8]. SPG builds on LPG's structure while incorporating semantic constraints similar to ontologies, making it well-suited for event-driven domains such as the telecom domain, where modeling causal, temporal and hierarchical relations between events is critical. This involves augmenting property graphs with semantic schemas, resulting in a hybrid model that integrates the structural simplicity of LPG with semantic constraints derived from RDF ontologies [3, 8]. In the following section, we will give a brief introduction to the OpenSPG engine investigated within this project.

2. OpenSPG: A Semantic-Enhanced Programmable Graph Engine

OpenSPG is a knowledge graph engine based on the SPG approach to improve industrial applicability [9, 8]. In this study, the OpenSPG engine[3] was chosen to explore our use case and compare the SPG approach with the well-established RDF model. OpenSPG was selected for its open-source availability as the first engine based on the SPG approach, and for its suitability in enhancing property graph-based projects with semantic capabilities. OpenSPG integrates logical-form parsing, symbolic computation, and retrieval-augmented generation to enable accurate and interpretable question answering in domain-specific contexts using KAG (Knowledge Augmented Generation) which is a reasoning framework developed on top of OpenSPG. It supports a unified representation of structured data, unstructured text, and expert-defined knowledge within a schema-constrained knowledge graph. The system architecture comprises two primary modules: the *kg-builder*, which extracts and integrates knowledge from heterogeneous sources through schema-guided or schema-free methods; and the*kg-solver*, which translates natural language queries into executable logical forms, enabling a combination of graph traversal, symbolic reasoning, and neural generation. This architecture enables the system to respond to natural language queries by leveraging large language models (LLMs) within a retrieval-augmented generation (RAG) framework, combining symbolic reasoning over structured knowledge graphs with neural language generation to support accurate and contextually grounded answers.

3. Lessons Learned

As both RDF and SPG offer distinct modeling philosophies and capabilities, we compare them along several criteria relevant to building explainable and performant root cause analysis systems in telecom (see Table 1). From this comparison as well as a proof of concept implementation for a preliminary router operations use case given in Github repository[4], several key lessons have emerged.

OpenSPG aims to support rapid development of enterprise-level knowledge systems by modeling the full knowledge lifecycle [10]. While the preliminary use case was successfully implemented, a number of lessons are worth sharing. The tool presents a steep learning curve, requiring several weeks to learn and master the main concepts for a developer. Limited community support and documentation at the moment further hinder usability.

OpenSPG emphasizes event-centric domain modeling, using context-specific and action-oriented relationships to represent entities and their interactions in dynamic systems such as networks. The relationships in OpenSPG engine are more focused on practical domain modeling rather than formal

[3]https://github.com/OpenSPG
[4]https://github.com/beyzayaman/openspg

logical reasoning. They are designed to capture the semantic meaning of entities and events in specific contexts (such as supply chains, business processes, or network systems). OWL relationships (properties) are defined within a formal ontology and support logical reasoning. It provides a more rigorous logical framework for relationships, including subclassing, inverse properties, and cardinality constraints. OpenSPG engine relationships are often more action-oriented (e.g., “locateAt”, “mannerOf”) and tailored for real-world systems where event-driven analysis is key. Both support inheritance (*is_a* in OpenSPG engine, *rdfs:subClassOf* in RDFS/OWL) and enforce domain and range constraints for semantic consistency.

Our use of SPG’s built-in schema and property constraints (e.g., enforcing node types and expected properties) helped uncover obvious data issues like missing identifiers or incorrect links. However, these validations are relatively shallow compared to what RDF technologies like SHACL can provide. SHACL allows declarative, expressive constraints (e.g., cardinality, value ranges, conditional constraints) and supports reusable constraint templates. Thus, currently OpenSPG lacks the mature data validation capabilities of RDF+SHACL, which are crucial in high-integrity domains like telecom.

In terms of uplift of data, although OpenSPG supports loading data from relational tables (MySQL), the alignment of column names to schema properties was found to be non-trivial in our use case. Schema mismatches or missing property definitions led to runtime errors or silent omissions in graph construction. Successful data ingestion required iterative tuning of schema-property alignment and index strategies.

In the implementation of our use case, we generated sample instances for each class and loaded them into OpenSPG. A total of 9 classes (including events, entities, and concept types) and 35 relations were defined for the use case. Among these relations, 4 were logically inferred rather than physically created, meaning they were derived through reasoning instead of being explicitly stored in the data. Due to the built-in vectorization for natural language querying, the data insertion into Neo4j was time-consuming (e.g., 15 minutes for 40KB on an 8GB RAM system). Additionally, query performance degraded significantly when primary keys were omitted, highlighting some current limitations in OpenSPG’s query optimization. Overall, despite our initial findings, our experience of using OpenSPG for our network scenario holds promise, and further exploration will be undertaken as the engine and supporting documentation evolves and matures. Despite its current immaturity, OpenSPG offers value for projects already based on property graphs seeking enhanced semantic capabilities.

Feature	**SPG-Schema (Semantic-enhanced Property Graphs)**	**Ontology-based Schema (RDFS/OWL - RDF Triplestore)**
Schema Definition	**Hybrid model**: Combines elements of **RDF schemas and Property Graphs** with controlled semantics.	**Strict ontology-driven**: Uses RDF Schema (RDFS) and OWL for defining classes, properties, and constraints.
Data Model	Property Graph with **semantic enhancements** (supports labels, relationships, and typed attributes).	**Triple-based** (subject-predicate-object) with strict ontological constraints.
Reasoning & Inference	Supports **programmable reasoning (SPG-Reasoner)** using **KGDSL**, enabling custom rules and hybrid AI-symbolic reasoning.	Strong reasoning using **RDFS entailment, OWL DL**, and SPARQL inferencing. Supports **logical inference** but can be computationally expensive.
Query Language	**KGDSL (Knowledge Graph DSL)** + Property Graph queries (Cypher, Gremlin).	**SPARQL** (standard for RDF queries).
Challenges	- **More complex** than traditional Property Graphs. - **Steep learning curve** due to hybrid schema.	- **Performance bottlenecks** at scale due to reasoning overhead. - **Difficult to manage for dynamic/fast-changing data**.
Flexibility vs. Control	**Balanced:** Provides schema control but allows extensions.	**Highly controlled:** Strict ontology-based constraints.
Expressiveness	Supports **semantic constraints** but retains flexibility for graph traversal.	High expressiveness due to **OWL-based reasoning**, supporting complex **class hierarchies, rules, and inference**.

Table 1
Comparison between SPG-Schema and Ontology-based RDF Schema

References

[1] F. Manola, E. Miller, B. McBride, et al., Rdf primer, W3C recommendation 10 (2004) 6.

[2] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, A. Taylor, Cypher: An evolving query language for property graphs, in: Proceedings of the 2018 international conference on management of data, 2018, pp. 1433–1445.

[3] S. Purohit, N. Van, G. Chin, Semantic property graph for scalable knowledge graph analytics, in: 2021 IEEE International Conference on Big Data (Big Data), IEEE, 2021, pp. 2672–2677.

[4] I. D. Martínez-Casanueva, L. Bellido, D. López, Declarative construction of knowledge graphs from netconf data sources, Semantic Web Journal, Under Review (????).

[5] I. D. Martinez-Casanueva, L. Bellido, D. González-Sánchez, D. Lopez, Candil: A federated data fabric for network analytics, Future Generation Computer Systems 158 (2024) 98–109.

[6] L. Tailhardat, Y. Chabot, R. Troncy, Noria-o: an ontology for anomaly detection and incident management in ict systems, in: European Semantic Web Conference, Springer, 2024, pp. 21–39.

[7] Z. Zhao, X. Ge, Z. Shen, C. Hu, H. Wang, S2ctrans: Building a bridge from sparql to cypher, in: International Conference on Database and Expert Systems Applications, Springer, 2023, pp. 424–430.

[8] L. Liang, M. Sun, Z. Gui, Z. Zhu, Z. Jiang, L. Zhong, P. Zhao, Z. Bo, J. Yang, et al., Kag: Boosting llms in professional domains via knowledge augmented generation, arXiv preprint arXiv:2409.13731 (2024).

[9] P. Yi, L. Liang, D. Zhang, Y. Chen, J. Zhu, X. Liu, K. Tang, J. Chen, H. Lin, L. Qiu, et al., Kgfabric: A scalable knowledge graph warehouse for enterprise data interconnection, Proceedings of the VLDB Endowment 17 (2024) 3841–3854.

[10] Semantic enhanced programmable knowledge graph white paper v1.0, source: Ant group and openkg (2023). URL: https://mdn.alipayobjects.com/huamei_xgb3qj/afts/file/A*6gpqRomoqIsAAAAAAAAAAAAADtmcAQ/Semantic-enhanced%20Programmable%20Knowledge%20Graph%20(SPG)%20%20White%20paper%20v1.0.pdf.