Chapter 7 SPG-Programming Layer

Decoupling the domain model from the underlying engine is a fundamental capability that AI basic engines must possess. By using the programmable SDK and operator frameworks, the business domain model can be separated from the underlying engine, allowing developers to quickly build and deploy self-defined knowledge graph algorithms, schema models, and reasoning capabilities. This helps businesses to rapidly develop and deploy knowledge graph applications, improving efficiency and scalability. The underlying engine also focuses on optimizing general capabilities such as I/O, scheduling, performance, and throughput. The SPG engine is also divided into three layers: the core underlying engine layer, the SDK and operator framework layer, and the business application layer.

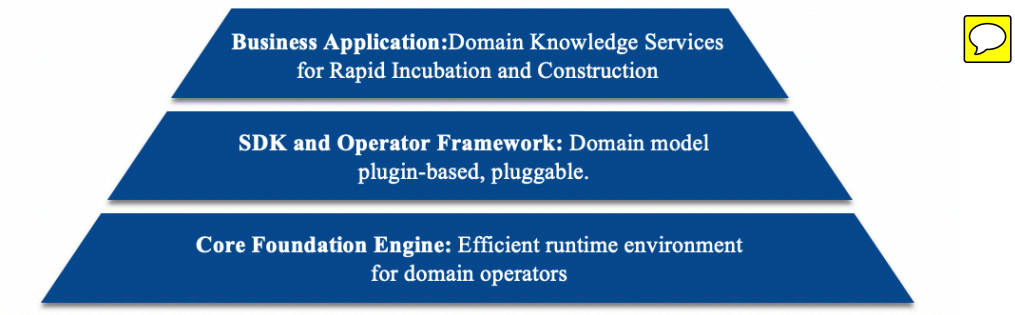

Figure 40: SPG-Programming Domain Layering

This chapter provides a brief overview of SPG-Programming. A more comprehensive introduction and the complete syntax representation of KGDSL will be officially released in the Whitepaper 2.0 version. It will also be detailed in a series of articles.

7.1 SPG Semantic Programmable Architecture

The main capability of the programmable sub-module is to provide users with a programmable interactive interface, supporting algorithm/business engineers to model, construct, and reason for the domain knowledge graphs according to the programming framework provided by SPG, and supporting the continuous iteration of the domain knowledge graphs.

There are 3 layers of SPG engine:

Core Engine Layer: Optimizes fundamental capabilities such as I/O, scheduling, performance, and throughput.

SDK and Operator Framework Layer: Provides tools and frameworks to define custom logic and operations.

Business Application Layer: Houses domain-specific models and logic, kept separate from the engine's internal details.

They isolate engine optimizations, provide customization tools, and host domain logic for efficiency and flexibility.

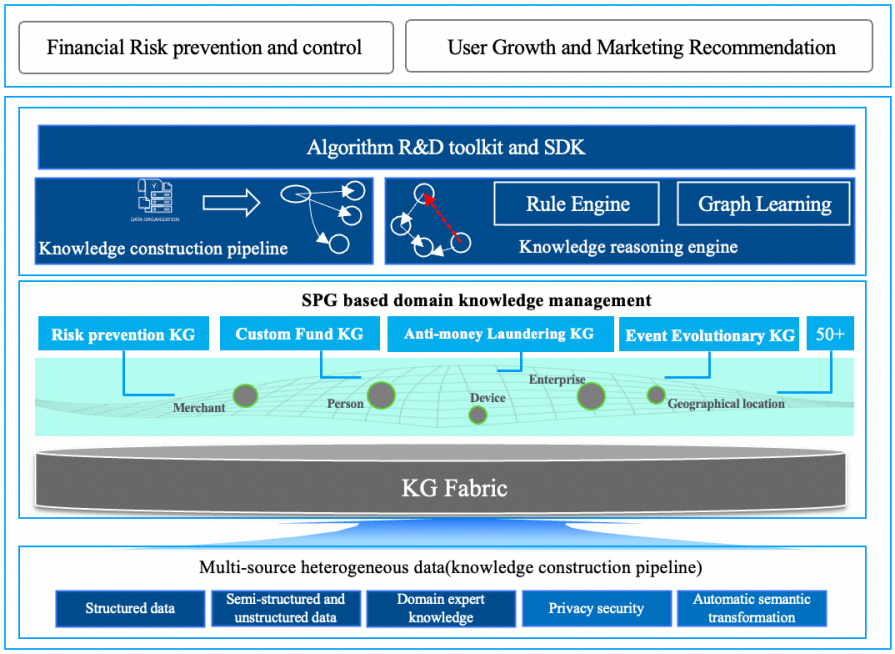

Figure 41: SPG Programmable Framework

In terms of programmability, it includes the following aspects:

- Knowledge Construction Framework: An operator framework for building the domain knowledge graphs based on structured, unstructured, and semi-structured data.
- Logical Rule Programming: Includes logical rule properties/relations, KGDSL rule reasoning, etc.
- Representation Learning and reasoning: Using knowledge graph-based data to implement graph neural networks, logic-guided reasoning, and other knowledge graph-based reasoning methods.

7.2 The Construction and Transformation from Data to Knowledge

As shown in Figure 1 in Chapter 1, with the help of the SPG framework, data can be transformed into knowledge through the programmable operators. These operators include the following categories: (1) Information Extraction Operators: These operators enable the structuring of all unstructured and semi-structured data to obtain knowledge elements. (2) Subject Mounting Operators: These operators involve mounting entities, events, and concept types, as well as mounting entity/event properties. (3) Entity Fusion Operators: These operators address issues related to the construction of entities from multiple heterogeneous sources and the fusion of knowledge across knowledge graphs. (4) Dynamic Classification Operators: These operators support the dynamic definition of business types at a granular level using logical rule-based classification expressions.

The use of standardized operator frameworks and property elements can significantly reduce the cost of preparing raw data. Operators are only related to target entities, concept types, and event types, thus greatly reducing the cost of redundant development. With the introduction of standardized property elements, the types of properties in schema modeling transition from text (Text), integer (Long), and float (Double) to the domain model modeling.

Constructing knowledge from data through detailed data structure, entities, events, concepts, and relationships significantly reduces data processing costs.

The pseudo-code representation based on LPG modeling is as follows:

```
class User {
  id TEXT(Text);
  name TEXT(Text);
  phoneNo LONG(Integer);
  poc     TEXT(Text);
  homeAddr TEXT(Text);
}
```

Pseudo-code representation based on SPG modeling is as follows:

```
class User {
  id UserNormId(User standard ID);
  name TEXT(Text);
  phoneNo ChnMobilePhone(Mainland China phone number);
  poc     AdminArea/Province/City/District(administrative division);
  homeAddr POI(Standard POI);
}
```

By preparing a User table, users can construct four types of entities and four types of relations, thereby significantly reducing the cost of preparing raw data. With the introduction of knowledge construction operators, the operator framework and operator implementation are separated, and the knowledge construction process is defined as standard components, providing a unified runtime framework for these components. The algorithm developers can quickly implement knowledge construction operators based on the python operator framework and bind them to target types. At the same time, Link Function/Fuse Function or Normalize Function can be specified for each type, facilitating the continuous iteration of the knowledge graph and addressing the issues of continuous construction and evolution of incomplete data and instance resolution in industrial applications. A simple example of the operator definition in pseudo-code is shown below, which can be bound to specific types using “bind_to”:

```
# -*- coding: utf-8 -*-

from knext.api import EntityLinkOp
from knext.api.base import BaseOp
from knext.models.runtime.vertex import Vertex

@BaseOp.register("AdminAreaNormOp", bind_to="AdminArea", is_api_iface=True)
class AdminAreaNormOp(PropertyNormalizeOp):
    def eval(self, property: str, record: Vertex) -> EvalResult[str]
        traces, errors = [], []
        result = ""
        try:
            result = adminNorm(property)
        except Exception as e:
            errors.append(f"property:{property}, error_msg:{e.__repr__()}")
        return EvalResult(result, traces, errors)
```

SPG supports the adaptive propagation of properties and relations during the query phase to minimize user usage costs. When users provide GQL/KGDSL expressions, if propagation is necessary, it will be automatically expanded. If propagation is not required, the standardized property values will be extracted by default. More detailed information about the KGDSL syntax will be provided in future articles.

7.3 Logical Rule Programming

Predicate semantics are the key foundation for implementing the SPG logical rule programming. Through predicate semantics, SPG can be translated into a machine-understandable form and enable machine automatic reasoning capabilities. The definition of these capabilities includes the following layers: (1) **Built-in predicates:** These are predefined basic predicate capabilities. They do not possess any business semantics but can be referenced by higher-level rules. (2) **Logical rule knowledge:** This layer includes logical rules that exist in the form of properties/relations, and enable dynamic classification of entities based on these rules. (3) **Reasoning decision rules:** These rules are derived from subgraphs, structures, paths, and other forms. They support rule-based decisions, knowledge injection, and more. Figure 42 illustrates the overall layered structure. To balance the cost of rule management and computational complexity, it is specified that built-in predicates can only be used to define logical rule knowledge. The application of knowledge reasoning layer relies only on basic factual knowledge and logical rule knowledge.

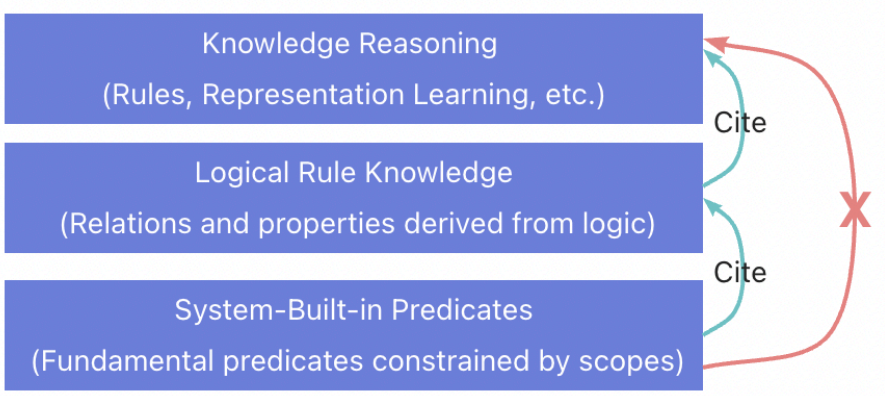

Figure 42: Dependency of the Reasoning Decision

Defining dependencies between knowledge using predicates and logical rules has been extensively explained in Chapter 4, and it will not be reiterated in this chapter. Logical rule programming mainly consists of two parts: defining knowledge dependencies through logical rules and generating logical derived properties/relations. Additionally, complex end-to-end rule decisions can be defined using DSL/GQL. An example of KGDSL decision is provided below:

```
Structure {
 (s:User)
 (e1:TradeEvent)-[ps1:std.subject]->(su1:User)
 (e1:TradeEvent)-[pp1:std.object]->(sp1:PID)
 (e2:TradeEvent)-[ps2: std.subject]->(su2:User)
 (e2:TradeEvent)-[pp2: std.object]->(sp2:PID)
 (su1)-[has]->(sp2)
```

```
  (su2)-[has]->(sp1)
  (e2)-[pb:belongTo]->(o:/TaxoOfTradeEvent/HighTransactionAmount)
}
Constraint {
  s.id == su1.id
  e1.ts < e2.ts and hour(current_time()) - hour(e1.ts) < 24
  group(s).count() > 10
}
Action {
  createEdgeInstance(
    src=s,
    dst=o:TaxoOfUser/TransactionRisk/MultipleRefundTransactions,
    type="belongTo",
    value= { time=now() }
  )
}
```

7.4 Knowledge Graph Representation Learning

The core problems addressed by the knowledge graph representation learning framework are graph feature extraction and subgraph extraction. It is designed to be compatible with mainstream deep learning frameworks such as TensorFlow/PyTorch and further convert them into tensor structures required by corresponding graph learning algorithms.

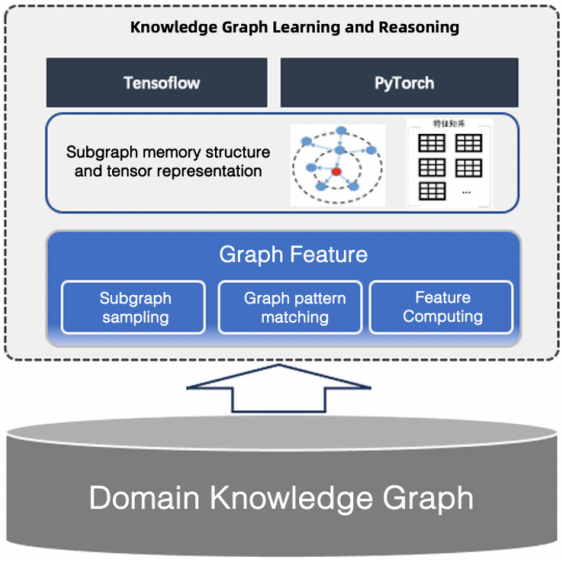

Figure 43: Graph Representation Learning Framework

Knowledge graph representation learning achieves the linkage and decoupling of the graph learning and the graph data through the sampling operator module. The main graph sampling operators currently available include: (1) Subgraph sampling: This is primarily used for multi-hop subgraph sampling in GCN models. It includes positive/negative sample generation and supports weighted sampling and time filtering in the SPG large-scale dynamic heterogeneous paradigm. (2) Structure extraction: This is used for structure-aware reasoning tasks such as symbol rule-guided graph learning and rule mining. (3) Feature computation: This mainly involves extracting subgraph features such as PageRank and degree centrality that can be

computed from the complex graph structure. The following pseudo-code represents the concept of multi-hop subgraph sampling in GCN algorithms. It demonstrates the efficient reading of the knowledge graph data using Python sampling operators:

```
# -*- coding: utf-8 -*-
import libkg_client
from kgrl.conf import KgrlConstants # noqa
from kgrl.data import KGExpression # noqa
from kgrl.data.sampler import KGStateCacheBaseSampler

in_degree = KGExpression.SourceNodeInDegreeKey()
out_degree = KGExpression.SourceNodeOutDegreeKey()
node_version = KGExpression.SourceNodeVersionKey()
edge_version = KGExpression.EdgeVersionKey()
v_begin = 30
v_end   = 40
def get_filters(v_begin, v_end):
   return {
      KgrlConstants.NEIGHBORHOOD_SAMPLING_FILTER_NAME: f"{edge_version}<{v_begin} and
{edge_version}>{v_end}",
      KgrlConstants.NODE_SAMPLING_FILTER_NAME: f"{node_version}==0",
      KgrlConstants.EDGE_SAMPLING_FILTER_NAME: f"{edge_version}<{v_begin} and {edge_version}>{v_end}",
   }
def get_weights(v_begin, v_end):
   return {
      KgrlConstants.NEIGHBORHOOD_SAMPLING_WEIGHT_NAME: f"abs({edge_version}-
{v_begin})*log2({edge_version}+{v_end})",
      KgrlConstants.NODE_SAMPLING_WEIGHT_NAME: f"({out_degree}+{in_degree})",
      KgrlConstants.EDGE_SAMPLING_WEIGHT_NAME: f"abs({edge_version}-
{v_begin})*log2({edge_version}+{v_end})",
   }
sampler_conf = {
    "client_conf": {...},
    "gen_data_conf": {
        "random": True, "fanouts": [50, 20], "buffer_size": 2, "filters": get_filters(10, 20), "weights": get_weights(10, 20),
   },
}
sampler = NodeSubGraphSampler.from_params(sampler_conf)
```

7.5 Summary

This chapter provides a summary introduction to the layered abstraction of the SDK programmable framework. The complete programmable framework is expected to be released in detail in the “SPG Semantic-enhanced Programmable Knowledge Graph Framework” whitepaper 2.0.

SPG employs a sampling operator module to integrate graph data with machine learning, supporting multi-hop and weighted sampling for efficient knowledge representation, while its layered SDK framework ensures scalability and flexibility.

Chapter 8 SPG-LLM Layer

At the beginning of 2023, Large Language Models (LLMs) have demonstrated their powerful capabilities, particularly in language understanding and dialogue generation. On the other hand, knowledge graphs excel in addressing factual “illusions” and complex reasoning problems that the LLMs struggle with. By effectively combining the strengths of knowledge graphs and LLMs, we can leverage their respective expertise to provide high-quality AI services and products.

Building upon SPG, the SPG+LLM dual-drive framework is formed by leveraging the structure, semantics, and logical understanding capabilities of LLMs. With SPG's strong schema, logical constraints, and symbolic expression capabilities, the efficiency of the domain knowledge construction and reasoning can be further improved. By integrating the intent understanding, intent diffusion, task construction, and controlled generation based on users' natural language expressions, SPG-LLM enables interactive knowledge querying and reasoning using natural language. This is an ongoing direction of exploration. Base on the strong schema, logical constraints, and symbolic expression capabilities of SPG, further efforts are made to improve the efficiency of the domain knowledge construction and reasoning, accelerate the industrial implementation of knowledge graph, and combine the intent understanding/intent diffusion, task construction, and controlled generation based on users' natural language expressions, to achieve interactive natural language querying and reasoning on the knowledge graph. This chapter provides a brief introduction to the architecture of LLM and SPG for natural language interaction. It also introduces knowledge extraction based on LLM, drawing from the practice of DaGuan Technology.

8.1 SPG-LLM Natural Language Interaction Architecture

Based on the overall architecture defined in Figure 24, the natural language interaction of the large language model (LLM) can be divided into four main parts: LLM Adapter Interface, SPG Constructor for automatic extraction and construction of the knowledge graph, SPG NL Query for natural language querying based on LLM, and SPG NL Reasoner for natural language reasoning based on LLM.

8.2 Automatic Extraction and Automated Construction of Knowledge Graphs

After using LLM, the process of constructing a knowledge graph is as shown in Figure 44.

Figure 44: The Construction Chain of Knowledge Graph

Business Understanding and Schema Design: The design and implementation of the knowledge graph schema require a deep understanding and abstraction of the knowledge in the domain. It also needs to consider the quality and accessibility of the data sources, as well as the requirements and limitations of the

application scenarios. This process is typically a collaborative effort involving multiple parties. The book "Knowledge Graph: Cognitive Intelligence Theory and Practice", provides a series of practical experiences and summarizes them as the "Six Taoist Methods" (as shown in Figure 44). In the schema design process, it is important to make full use of the content of SPG-Schema and further expand it by introducing standardized natural language annotations. The content related to natural language annotations of SPG-Schema will be introduced in future versions of the whitepaper. We can refer to the definitions within Ontology, which allows us to define concepts, properties, relations, constraints, and rules. It also supports inference and validation. Well-known ontology libraries such as schema.org, FIBO, GO, etc., can be referenced or reused to optimize the design of the schema. The goal is to ensure the standardization, consistency, and universality of the designed schema.

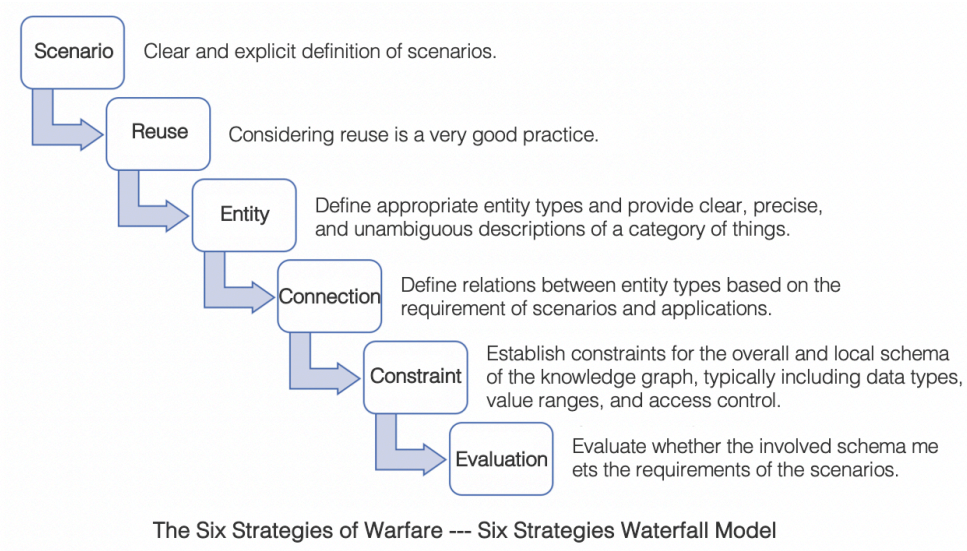

Figure 45: Chapter 2 of the book "Knowledge Graph: Cognitive Intelligence Theory and Practice" [6]

Manually curated Examples and Automatic/Human-Written Prompts: The process of prompt engineering is based on the designed schema to achieve automatic extraction of entities, relations, and properties, thereby constructing a knowledge graph. The engine for automating prompt generation can also be implemented with reference to the reasoning engine in the ontology. The generation of the prompts relies on the natural language annotations in the schema and the manually curated examples. In practice, the manually curated examples or the use of LLM-generated extraction examples can help with few-shot learning to improve the accuracy of LLM extraction.

SPG-LLM's role in refining AI-driven knowledge extraction, ensuring structured schema design, and improving accuracy through ontology-driven validation and human-AI collaboration.

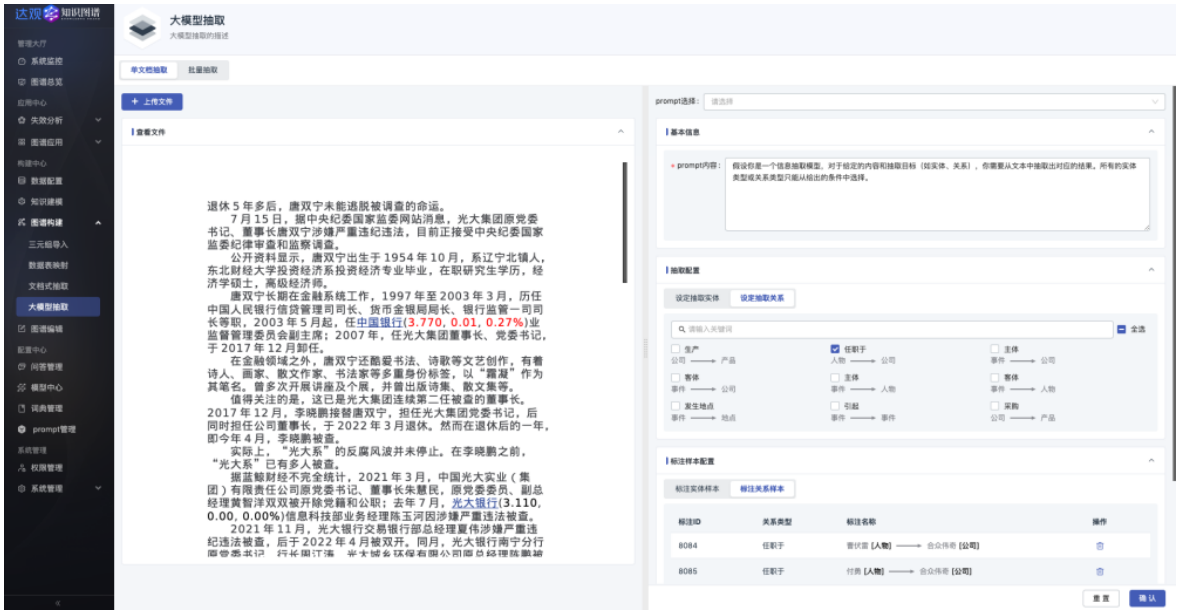

Figure 46: Example of LLMs Extraction

LLM Extraction and Optional Manual Review: Utilizing LLM to construct a knowledge graph, with the provision of manual review when necessary to ensure the accuracy of the constructed knowledge graph.

Figure 47: Examples of LLMs Extraction and Review

Knowledge Graph Construction: Integrating the results extracted by LLM into an existing knowledge graph, involves using LLM for entity extraction, relation extraction, and other methods to construct the knowledge graph from a large amount of text. The key lies in the specification of the entity types, relation types, property types, and other elements defined in the knowledge graph schema, particularly the relevant natural language annotations. This is strongly related to the specification of the SPG-Schema, which provides natural language annotations in the schema, aiding in their transformation into prompts for LLM extraction and interaction. There should be three levels of content in this regard:

- Natural language annotations for the entity types, relation types, and property types.

LLMs aid in constructing knowledge graphs by extracting entities, relations, and properties, integrating schema-based annotations, and improving accuracy through few-shot learning.

- Natural language annotations for the concept hierarchies, semantic correlations, logical rules, and other aspects.
- Standardization of the above two types of annotations, which helps in achieving a common library (engine) for automated prompt generation.

The natural language annotations in the schema serve two purposes: on one hand, they enable the automatic generation of the prompts, and on the other hand, when using LLMs to construct knowledge graphs, they can be used to generate samples for few-shot learning. In practice, few-shot learning is crucial in relation extraction. It is difficult to achieve good performance in zero-shot learning, but few-shot learning can significantly improve the effectiveness of relation extraction.

8.3 Domain Knowledge Completion with LLMs

Using LLMs for knowledge completion can help small and medium-sized institutions acquire richer domain knowledge. Compared to relying solely on internally accumulated knowledge, LLMs leverage vast amounts of text data to obtain comprehensive common knowledge and domain-specific knowledge. By extracting and saving knowledge from LLMs into a knowledge graph using specialized methods, organizations can benefit from more efficient knowledge accumulation and utilization. In contrast to traditional knowledge mining processes that aim to make implicit knowledge explicit in existing knowledge graphs, knowledge completion with LLMs focuses more on extracting domain-specific knowledge from LLMs and integrating them into the knowledge graph, providing knowledge that does not exist in the knowledge graph. This whitepaper only introduces the concept of "knowledge completion" using LLMs, and further versions will provide more implementation methods, examples, significances, and other details.

8.4 Natural Language Knowledge Querying and Intelligent Question Answering

Traditional knowledge graphs have limited natural language understanding capabilities. LLMs can help address this deficiency by leveraging their language understanding and generation capabilities, which are developed through training on billions of parameters. By combining the two, knowledge graphs can understand user queries in natural language and provide accurate answers using their inherent knowledge. The LLM is responsible for semantic analysis, while the knowledge graph provides structured knowledge for answer retrieval, complementing each other. This combination leverages the advantages of structured knowledge in knowledge graphs and the language understanding capabilities of LLMs, thereby providing more user-friendly question answering services. The LLM analyzes the real intent of the query, while the knowledge graph offers rich background knowledge to assist in retrieving more accurate and relevant search results. In dialogue systems, knowledge graphs provide a source of contextual knowledge, making conversations more intelligent and human-like. The LLM handles natural language interaction, while the knowledge graph supplements relevant knowledge, enabling the robot to have stronger contextual awareness.

LLMs enhance domain knowledge completion and natural language question answering by extracting knowledge, analyzing intent, and working alongside structured knowledge graphs to provide accurate, context-aware responses.

By combining the powerful capabilities of LLMs and vector search, integrating natural language interaction with knowledge graphs can create controllable, trustworthy, and reliable question answering systems. This approach helps address the "hallucination" problem faced by LLMs and facilitates the implementation of industrial applications, enabling the realization of the "last mile". Please refer to Figure 48 for illustration.

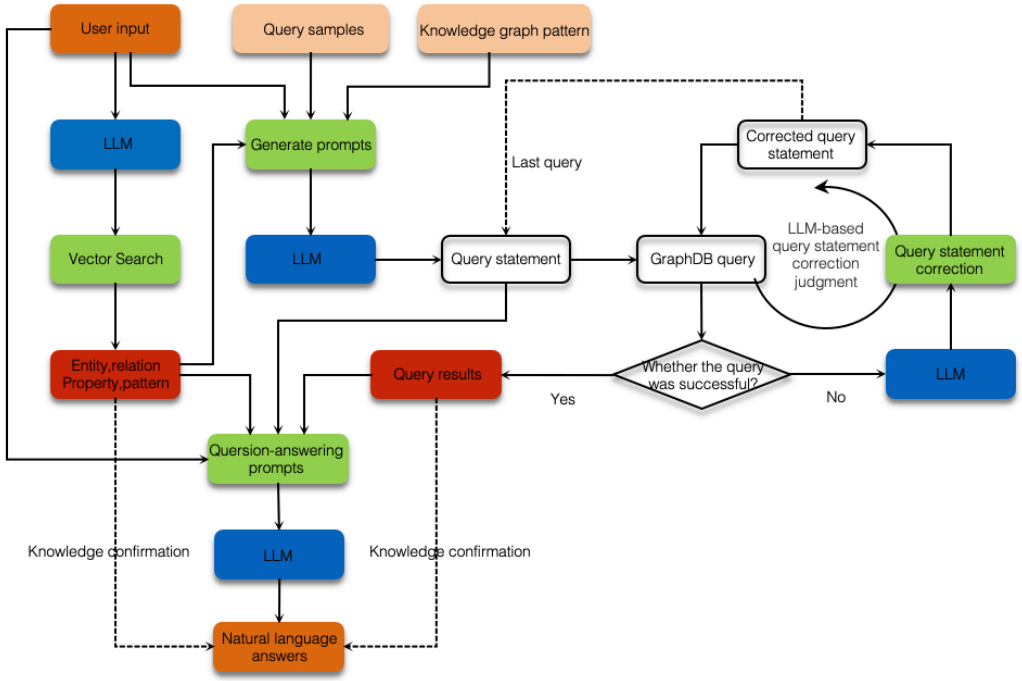

Figure 48: Architecture for controllable, trustworthy, and reliable question answering with the combination of LLMs and knowledge graphs (early draft)

NL2GQL/NL2KGDSL utilizes human-annotated datasets, which contains tens of thousands of natural language - GQL/KGDSL pairs, to perform Semantic Feature-based Training (SFT) on LLM, enabling natural language knowledge querying and intelligent question answering. The focus of this whitepaper release is primarily on theoretical exploration.

In future versions, as GQL or KGDSL mature, related datasets will be released, along with specifications, code repositories, and models for SFT based on open-source large language models. Please stay tuned for further updates.

8.5 Summary

This chapter provides a summary introduction to the basic principles and framework of the SPG-LLM layer. It also introduces the concept of "knowledge completion". The future release of the "Semantic-enhanced Programmable Knowledge Graph Framework (SPG)" whitepaper is expected to focus on providing a comprehensive overview of the SPG-LLM layer.

Integrating LLMs with knowledge graphs and vector search enhances question-answering reliability, mitigates hallucinations, and enables natural language-based intelligent querying through Semantic Feature-based Training (SFT).

Chapter 9 New Generation Cognitive Application Cases Driven by SPG

In Chapter 2, we summarized and analyzed the problems in the construction and application of LPG-based knowledge graph in enterprise causal and risk control. This chapter combines the problems raised in Chapter 2 to explain how they are solved based on SPG, and provides an overall solution.

9.1 Enterprise Causal Knowledge Graph Driven by SPG

In this chapter, we take the example of the 2019 dam collapse incident at the Vale of Brazil, mentioned in section 2.3. The incident caused an increase in the price of iron ore, resulting in a rise in steel production costs for downstream companies. In the entire chain of events, companies belonging to the same industry as Vale (i.e., competitors) benefited from the incident and saw an increase in profits. However, this had a negative impact on the downstream of the industry chain, as the rising cost of raw materials led to a decrease in profits for these companies. In response to the issues raised in section 2.3 regarding the application of the enterprise causal knowledge graph based on property graph, we propose a solution using SPG.

1. Addressing the Requirement for Dynamic Event Classification through Derivable Concepts

The enterprise causal knowledge graph involves the evolution and reasoning of events at the conceptual level. Therefore, the first step is to map event instances to their respective event concepts, using the "belongTo" predicate to connect event instances to the corresponding concepts. Since there are numerous event types that are difficult to define in advance, SPG supports the derivation of new concepts through specific combination rules using concepts. This enables dynamic classification of event instances. An example is shown in Figure 49.

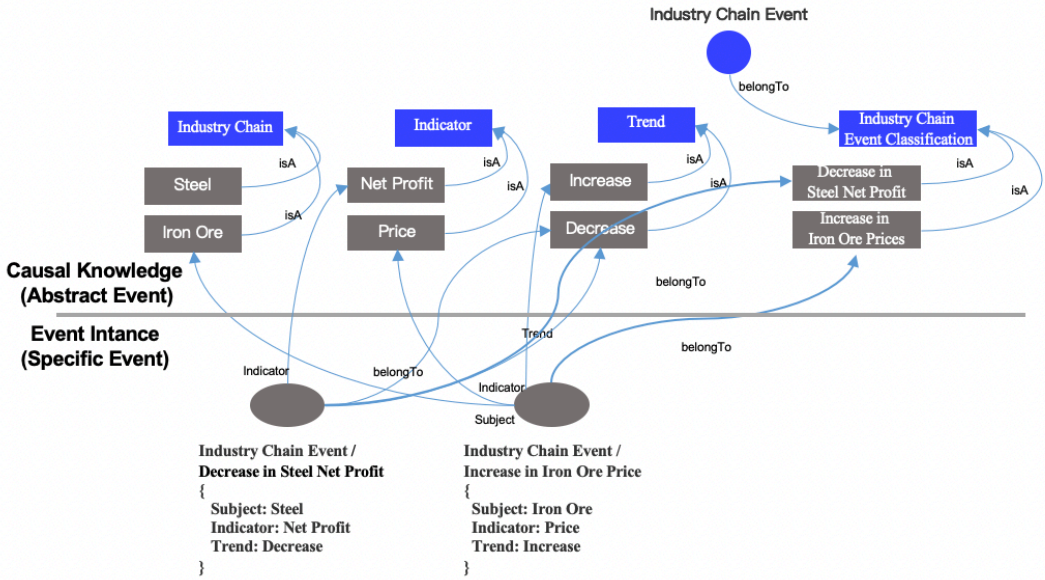

Figure 49: Combination of Concepts

SPG enables causal knowledge graph modeling for event-driven insights, dynamically classifying new events through derivable concepts, as demonstrated with the 2019 Vale iron ore incident.

In this example, there are two event instances: “Decrease in Steel Net Profit” and “Increase in Iron Ore Prices”. The former belongs to the event classification of the industry chain, and the concept of “Decrease in Net Profit” can be derived through the combination of the “indicators: Net Profit” and the “trend: Decrease”. Similarly, the latter also belongs to the event classification of the industry chain, and the concept of "Increase in Iron Ore Prices" can be derived through the combination of the “industry chain: Iron Ore”, the “indicator: Prices”, and the “trend: Increase”. It is worth noting that SPG does not immediately derive all possible combinations of concepts but rather derives the corresponding event concepts based on the actual occurrence of event instances, in order to avoid meaningless or logically inconsistent concept systems.

2. Solving the Inability to Express the Entire Event Context through Conceptual Causal Modeling

Building upon the solution to the previous problem, SPG can establish causal relations within the conceptual classification system to express the context of events, as shown in Figure 50. The red dashed lines represent the event propagation automatically generated in the event instance layer after the activation of the “leadTo” predicate in the causal knowledge layer.

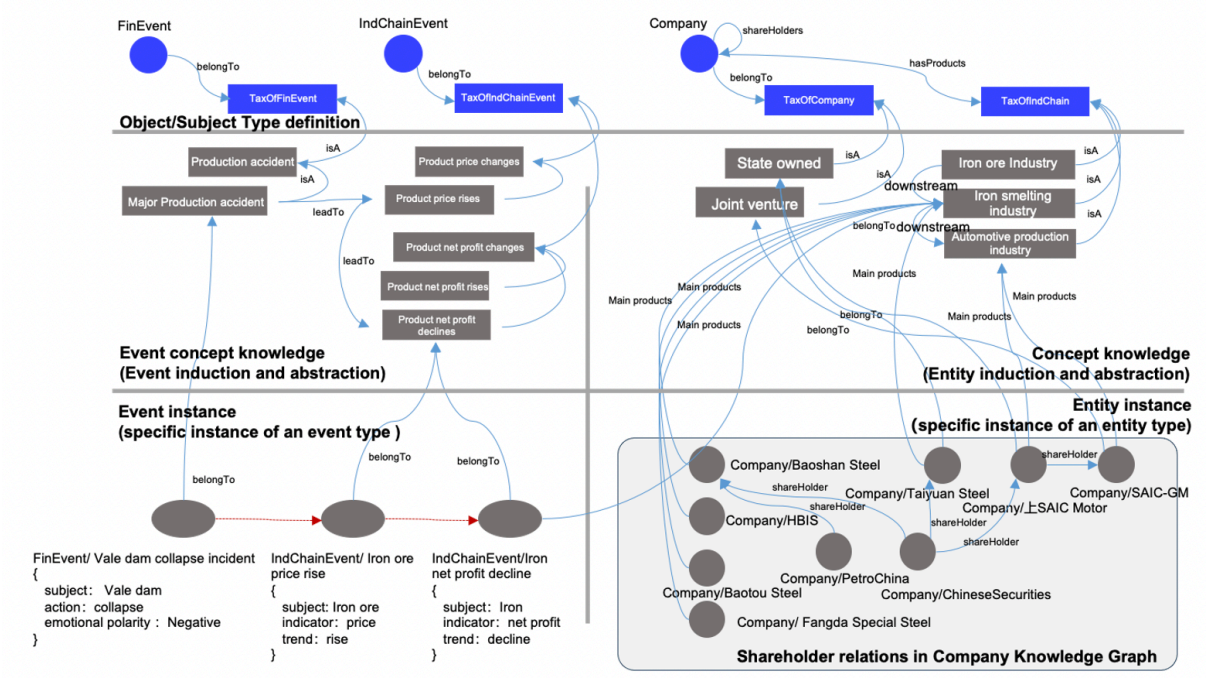

Figure 50: Causal Reasoning

In Figure 50, the diagram is divided into two layers from bottom to top: the event instance layer and the conceptual knowledge layer. The event instance layer represents specific event instances, such as the "Brazilian dam collapse incident," which can be mapped to the conceptual knowledge layer using the method described in 1). The conceptual knowledge layer has the ability to express causal relations at the conceptual level. In this example, the following causal relations have been defined in the conceptual knowledge layer: a “major production accident” in a company leads to either an “increase in iron ore prices” or a “decrease in steel net profit” in the industry chain to which the company belongs. When the Brazilian dam collapse incident occurs and is classified as a “major production accident” concept, the event inference transitions

SPG enables conceptual causal modeling by linking event instances to higher-level concepts, dynamically classifying events through propagation mechanisms and predefined rules.

from the event instance layer to the conceptual knowledge layer. Based on the defined conceptual knowledge, the event propagation mechanism generates a new industry chain event instance at the event instance layer. This new event instance is then classified as the concept of "increase in iron ore prices", making it an "increase in iron ore prices event". The conceptual knowledge can be derived from event induction or generalized from causal patterns, guiding the classification and propagation of specific event instances. For example, a "major production accident" leads to the occurrence of an "industry chain event", which belongs to either the concept of "increase in iron ore prices" or the concept of "decrease in steel net profit". The rule template is as follows:

RULE1: Major Production Accident -leadTo-> Industry Chain Event

RULE2: Industry Chain Event -belongTo-> Increase in Iron Ore Prices

RULE3: Industry Chain Event -belongTo-> Decrease in Steel Net Profit

At this point, events can be classified into derived concepts, allowing the derived concepts that satisfy the rules to form causal relations and ultimately form the causal relations in the conceptual knowledge layer. This expression guides the classification and propagation of the event instances from top to bottom, enabling the expression of the entire event context in the conceptual knowledge layer and the event instance layer. In the example above, the conceptual knowledge layer expresses that accidents lead to a decrease in profits for the downstream companies in the industry to which the entities involved in the accident belong. When concretized at the event instance layer, this means that the "Brazilian dam collapse incident" leds to a decrease in profits for several steel companies.

It is important to note that SPG provides a framework for causal description, and the causal relations between concepts still need to be created by users based on their business characteristics. More detailed examples of concept induction methods will be published in future articles of the SPG, and addressing specific practices.

3. Built-in Logical Expressions to Address Data Unconstrained by Reasoning Logic

SPG expresses conceptual classification logic using the logical constraints. Taking the dam collapse incident as an example, the following rules can be defined to classify this event:

```
Define (s:FinancialEvent)-[p:belongTo]->(o:`FinancialEventTaxonomy/MajorIndustrialAccidents`) {
  Structure {
    (s)-[:std.subject]->(company:Company) //Associated company instances
  }
  Constraint {
    R1("Subject experienced a Industrial accident"): s.behavior == 'Industrial accident'
    R2("Subject company has a market share exceeding x% with significant impact"): company.marketShare > x%
  }
}
```

The above rules can be interpreted as follows: when a safety accident occurs and the market share of the company exceeds x%, then the event is classified as a major production accident in the corresponding

SPG uses logical constraints to classify unconstrained data, ensuring accurate categorization of financial and industrial events through structured reasoning rules.

industry. Similarly, the “leadTo” relation between concepts can also be logically expressed. When event “e1” occurs, it will generate and activate another event “e2”.

```
Define (s:`FinancialEventTaxonomy/MajorIndustrialAccidents`)-[p:leadTo]->(o:`IndustryChainEventTaxonomy/
priceIncrease `) {
   Structure {
      (s)-[:std.subject]->(company:Company)-[:industry]->[I:Industry)
   }
   Constraint {
   }
   Action {
      createNode(
         type=IndustryChainEvent
         value={
            subject=I.name
            index='price'
            trend='increase'
         }
      )
   }
}
```

After a new event instance is generated through event propagation, the new event instance can trigger the classification of the event again. The classification of the event can be done using combination concepts. Here is an example that uses the combination concepts of “IndustryChain”, “Index”, and “Trend” to define the classification of the industry chain event mentioned earlier.

```
Define (s: IndustryChainEvent)-[p:belongTo]->(o: `IndustryChainEventTaxonomy`/`IndustryChain` + `Index` + `Trend`) {
   Structure {
   }
   Constraint {
      o = s.subject + s.index + s.trend
   }
}
```

4. Logical Properties and Relations to Address External Dependency on Auxiliary Data

In the example mentioned above, the Company entity has a property called “marketShare”, which represents the market share of the company. In practice, this data may come from other systems and may not exist in the knowledge graph. In such cases, the logical rules can be used to define the source of the data. An example is as follow:

```
Define (s:Compnay)-[p:marketShare]->(o:Float) {
  Structure {
    (s)
  }
 Constraint {
    o = callForMarketShares(s.id, 'marketShare')
  }
}
```

SPG enables dynamic event classification using conceptual event indicators and retrieves external data without storing redundant auxiliary information, ensuring logical consistency.

In line 6, the code "callForMarketShares" refers to a user-defined operator that can retrieve the market share information from other systems. Unlike the previous approach of importing all data into the knowledge graph, this method does not require additional copying of data from other systems. It ensures logical consistency of the data. This approach addresses the dependency on external data for decision-making in the context of enterprise causal knowledge graph scenarios.

5. SPG Addresses the Lack of Interpretability in Reasoning Conclusions

In this example, SPG decouples the propagation issue between the conceptual level and the instance level through the definition of event concepts. It also ensures the consistency between the logical rules and data. Following the four-quadrant approach, SPG ensures interpretability in the following four aspects:

- **Interpretability of the generalization and derived logic of the event concept ontology**

At the level of causal modeling, a structured representation scheme is defined for various types of events and entity types. At the same time, a top-down approach is used to define the concepts of entity ontology (such as product classification) and event ontology in a hierarchical system. Each more granular event concept is realized through filling in the event slot values, which concretizes the higher-level event concept (for example, in Figure 50, the "Product Price Change Event" is a specialization concept of the "Industry Chain Event", and fills the slot of "Indicator" with the specific value of "Price". On the other hand, the "Increase in Net Profit" provides further constraints on the concept of "Net Profit Change" in the slot of "Trend". By defining slots and concretizing slot values in a top-down manner, the interpretability of concept semantics is achieved through the combination of slot value properties. Clear generalization and derived logic exist between upper and lower concepts.

- **Interpretability of the logical relations in causal knowledge**

By defining the RULE patterns, the logical relations of causality, succession, and spatiotemporal relations between conceptual events are defined. For example, "an increase in product price" does not necessarily lead to "a decrease in product profit". Through the summary of the domain expert knowledge or analysis of numerous actual cases, rules such as "an increase in the price of upstream raw materials leads to a decrease in net profit of downstream products" can be obtained due to the impact of supply and demand relations in the industry chain. Furthermore, Based on the relations between upstream and downstream in the industry chain, specific causal logic, such as price increase-profit decrease, can be derived and generated in batches. For example, "increase in iron ore price -> decrease in steel profit" and "increase in steel price -> decrease in automobile profit". By defining and applying rules for generating multiple causal knowledge and combining them, an interpretable causal knowledge system can be generated for specific industries and scenarios.

- **Traceability of the factual causality and spatiotemporal succession between event instances**

The event subject, occurrence time, location of each sub-event composing the factual chain, as well as the factual associations, semantic associations, spatiotemporal co-occurrence or succession relations between these elements, provide the basis for the establishment and attribution traceability of the relations

SPG enhances interpretability in causal knowledge reasoning by structuring event ontologies hierarchically, defining logical causal relationships, and ensuring traceability in event classification.

SPG enables the interpretation of causal knowledge through factual relations.

between event instances. In Figure 51, the dam collapse incident at Vale on January 25th, 2019, and the subsequent rise in iron ore prices to a high level in July of the same year, along with the decrease in net profit of Baosteel and Fangda Special Steel, are supported by news reports and financial disclosures. Through the "belongTo" relation between the event instance and the event concept, the "leadTo" relation in the causal knowledge layer, and the industry chain relation between iron ore, steel, and automobiles, it can be explained clearly that these events are not independent but rather form a factual chain that can be explained by the causal relations within the industry chain.

- **Interpretability of the inductive and deduction between factual relations and causal knowledge**

SPG decouples factual relations from causal logic, ensuring accuracy in reasoning.

At the level of causal knowledge, the logical relations and common sense relations between abstract and specific ontological concepts are defined. In the level of factual instances, the structured and semantically standardized representation of instance knowledge and the factual associations between events are defined. The representation of factual relations and causal knowledge is decoupled, while using standard predicates in SPG, such as "belongTo", "isA", and "isInstanceOf", providing a unified representation method for the deduction from conceptual events to specific facts and the induction from specific factual relations to causal logic. This association and logical interpretation between abstract concepts and specific facts, can help validate the correctness of causal relations using existing factual relation samples in specific scenarios, and assist in the exploration of hidden causal and succession relations between events using causal logic. For example, by using the generated causal pattern "increase in steel price - decrease in automobile net profit" and integrating the equity penetration relations of the automobile companies in the enterprise knowledge graph, automobile companies with profit decline risks can be discovered.

9.2 Comparison between SPG and LPG in the context of Enterprise Causal Knowledge Graph

Table 13: Comparison of capabilities between SPG and LPG in the context of enterprise causal knowledge graph

Application Phase	Issues in Application	LPG Solutions	SPG Solutions
Causal Definition and Propagation	Causal Definition	Generally replaced by business systems	Supports causal definition based on derived concepts
	Propagation of events between causal layers		Uses specific inference predicates like "leadTo" to describe causal definitions
Event Definition	How events are related to entity graph	There are two main ways: 1) Define events based on subgraphs, which require modeling and data preparation 2) Directly retrieve events from the business system, which requires business system development	Supports native event type definition, simplifying modeling and data preparation without the need for business system development
Event Classification	Single and multiple event classifications	Usually establishing a business system to classify and label events, or not classifying them and directly using analysis tasks instead of classification conclusions.	Supports dynamic classification of events to concepts
	The problem of non-exhaustive event classification		Derives new concepts (abstract events) using combination concepts
Event Propagation	How to describe the impact of an event	Usually customized processing is done in the form of tasks in the business system	Event propagation is achieved based on the linkage between events and concepts.

Comparison of SPG vs. LPG in Enterprise Causal Knowledge Graphs.

SPG improves causal reasoning by decoupling factual relations from causal logic, enabling validation of real-world event dependencies and offering superior interpretability over LPG.

As shown in the table above, SPG provides a framework for causal expression. In comparison with LPG, it effectively represents the propagation chain of the events, providing a new practical approach for rapid analysis and response to the impacts of financial events.

9.3 SPG-Driven Risk Mining Knowledge Graph

In Chapter 2, the challenges of applying the risk mining knowledge graph were discussed, focusing on the maintenance and management of data as well as the high cost of understanding. The entities and relations mentioned in Chapter 2, can be classified into two categories based on their data generation methods: (1) Basic data, which comes from original table data. Entities such as “Person”, “Phone”, “Cert”, “Device”, “App”, etc., and relations such as “Person-has->Phone” and “Person-has->Cert”, can be directly derived from the original tables. (2) Derived data, which are generated from basic data or other derived data. For example, relations such as “Person-samePhone->Person” and “Person-developed-App” are derived logically. Below, we will discuss in detail how the SPG solution can be applied to the risk mining knowledge graph, and addressing the challenges previously mentioned.

1. To address the issue of data inflation and increased costs after converting raw data into graph data. The original table has significant differences compared to the basic data. For example, the original table only provides the User table and the Application table, without the Device, Certificate, or Phone tables. These pieces of are usually present as fields in the User table and the Application table. Building a knowledge graph using LPG typically requires users to perform additional data transformation or provide mapping operations. SPG, on the other hand, offers standardized property capabilities to simplify user data modeling and reduce data cleaning costs. Here is an example of how to represent phones, devices, and certificates as standard properties:

```
CREATE TYPE (std.Phone {
        value STRING REGEX '^1([38]\d|5[0-35-9]|7[3678])\d{8}$'
});
CREATE TYPE (std.Cert {
        value STRING REGEX '^[a-f0-9]{32}$'
});
CREATE TYPE (std.Device {
        value STRING REGEX '^([0-9A-Fa-f]{2}[:-]){5}([0-9A-Fa-f]{2})$'
});
```

To further define other entities:

```
CREATE NODE ( User {
  id STRING,                  //User primary key
  name STRING,                //User name
  type STRING,                //User type, individual or legal entity
  hasPhoneNum std.Phone,      //Using standard property here
  hasCert std.Cert,           //Using standard property here
  hasDevice std.Device        //Using standard property here
});
```

SPG enhances risk mining in financial knowledge graphs by standardizing data properties, reducing data transformation costs, and enabling efficient modeling compared to LPG.

```
CREATE NODE ( App {
  id STRING,
  riskType STRING,              // Risk flag
  hasCert std.Cert,             // Using standard property here
  installDevice std.Device      // Using standard property here
});
```

Due to the use of the standard property to replace relation modeling, there is no explicit definition of relations. Only the import of the user information table and the application information table is required. It can be observed that using the modeling capabilities of SPG, simplifies the modeling cost of entities such as Device and Certificate, and also reduces the cost of the data cleaning.

2. Resolve the issue of duplicate data preparation caused by different business characteristics and support the reuse of knowledge graphs across different businesses. The risk mining knowledge graph requires the use of transfer data and equity data. SPG provides knowledge fusion capabilities, the entities and relations from other knowledge graphs can be referenced and integrated into the current knowledge graph by utilizing self-defined entity resolution operators, to accommodate the specific requirements of the business scenario. These entities and relations can be referenced from the funding knowledge graph and equity knowledge graph. For example, as shown in Figure 51, the fusion of the funding knowledge graph and the risk mining knowledge graph is depicted, where the textual structure of the instance is represented as: Type/PropertyName = PropertyValue.

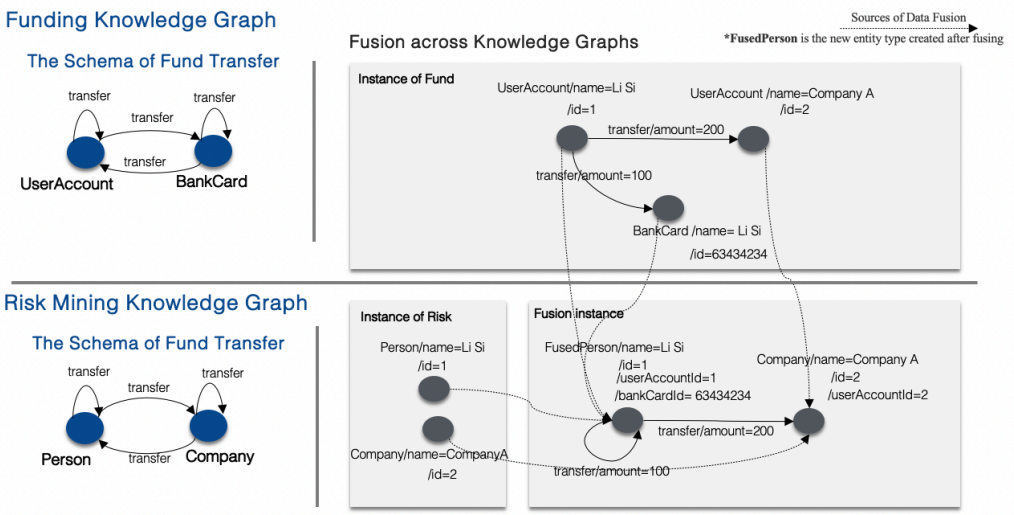

Figure 51: Knowledge Fusion across Knowledge Graphs

"FusedPerson" is derived from the entity linking and resolution operators of "UserAccount" from the funding knowledge graph and "Person" from the risk mining knowledge graph. To achieve this unified relation, two stages need to be defined: entity linking and entity resolution. It is worth noting that "FusedPerson" is only a declaration of the type and the association of the operators, and it does not generate actual fusion instances, which greatly reduces computation and storage costs.

1) **Entity linking:** This stage primarily defines how the instances of "UserAccount" and "Person" are corresponded using the entity linking operators. It can be a one-to-one correspondence or a many-to-one

SPG simplifies knowledge graph integration by replacing relation modeling with standard properties and supporting entity linking across different business knowledge graphs.

SPG resolves inconsistencies using rule-based entity resolution.

correspondence. In this example, it is a many-to-one correspondence. Entity linking across knowledge graphs can be achieved by applying rule-based linking operators. The pseudocode for the operator interface definition is as follows:

```
@BaseOp.register("FusedPersonLinkOp", bind_to="FusedPerson", is_api_iface=True)
class FusedPersonLinkOp(EntityLinkingOp):
  def eval(self, record: Vertex) -> EvalResult[List[Vertex]]:
      pass
```

2) Entity resolution: In this example, entity resolution is achieved using the resolution operators. Based on the conditional expression rules, the properties, relations can be filtered and processed for the successfully mapped instances of “UserAccount” and “Person”. The pseudocode for the operator interface definition is as follows:

```
@BaseOp.register("PersonFuseOp", bind_to="FusedPerson", is_api_iface=True)
class FusedPersonFuseOp(EntityFuseOp):
   def eval(
      self, source_vertex: Vertex, target_vertexes: List[Vertex]
   ) -> EvalResult[List[Vertex]]:
      pass
```

3) Addressing the issue of inconsistency in relation data due to logical dependencies. SPG provides capabilities for derived relations and derived data. Let's take the example of “same phone number” and “same person” as the logical dependencies.

```
Define (s:Person)-[p:samePhone]->(o:Person) {
   Structure {
      (s)-[:hasPhoneNum]->(w:std.Phone]<-[:hasPhoneNum]-(o)
   }
   Constraint {   }
}
```

The pre-defined relations types can be reused in the logical rules of KGDSL. For example, the “same person” relation can be defined as two individuals having the same phone number and the same device simultaneously.

```
Define (s:Person)-[p:sameUser]->(o:Person) {
   Structure {
      (s)-[:hasPhoneNum]->(o),  (s)-[:hasDevice]->(o)
   }
   Constraint {   }
}
```

The complex corporate control relations can also be defined through transitive, as shown in Figure 52, where the textual structure of the instance is represented as: Type/PropertyName = PropertyValue.

SPG resolves data inconsistencies by enabling logical dependencies, rule-based entity resolution, and transitive relations for advanced corporate data modeling.

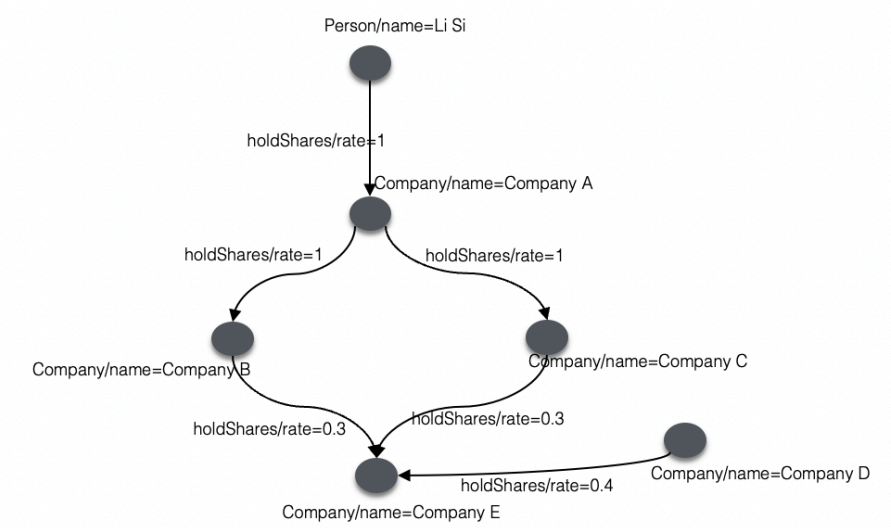

Figure 52: Example of holdShare Relations

```
// Define the holding proportion first, the transitive predicate requires that the types in the GraphStructure must be the same.
Define transitive (s:Company)-[p:holdShares]->(o:Company) {
    Structure {
        // The graph structure must follow the following pattern to express transitivity
        (s)-[p1:holdShares]->(c:Company), (c)-[p2:holdShares]->(o)
    }
    Constraint {
        // Group and aggregate by entity s and o to obtain all actual equity.
        real_rate("The actual holding proportion") = group(s,o).sum(p1.shares*p2.shares)
        p.shares = real_rate  // assigning the actual shares
    }
}
Define (s:Person)-[p: indirectHolding]->(o:Company) {
    Structure {
        (s)-[p1:holdShares]->(c:Company)-[p2:holdShares]->(o)    // Indirect control of company o through company c
    }
    Constraint {
        R1("Direct controlling equity ratio must be greater than 50%"): p1.shares > 0.5
        R2("Indirect controlling equity ratio must be greater than 50%"): p2.shares > 0.5
    }
}
```

By extension, all relations can be completed based on expert rules.

4) Overcoming the problem of continuously expanding and ultimately unmaintainable schema and data in the face of iterative business evolution.

In traditional LPG models, data and schema are tightly coupled. If there are changes in the business, the schema needs to be modified accordingly, resulting in high costs. However, SPG provides dynamic classification capabilities based on concepts, allowing for business extensions at the conceptual level, as shown in Figure 53.

SPG enables flexible schema evolution through conceptual extensions.

SPG models corporate ownership through transitive relations, enables flexible schema evolution, and supports dynamic classification for evolving business structures.

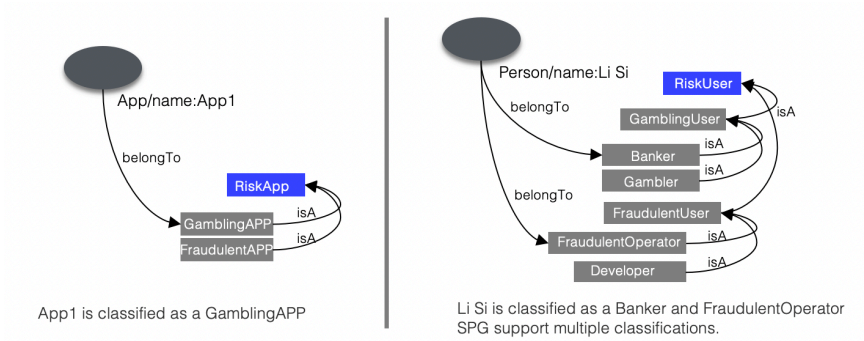

Figure 53: Concept-based dynamic entity classification in the risk mining knowledge graph

We introduce a new reasoning predicate called “belongTo”, which links the entities and concepts that meet the rule requirements. For example, when an App is confirmed as a fraudulent App, its developer can be classified as a suspicious fraudster. Taking "Fraudster" as an example, the rule for dynamic classification can be defined as follows:

```
Define (s:Person)-[p:belongTo]->(o:TaxonomyOfRiskUser/Fraudster) {
   Structure {
      // The person developing fraudulent applications is a fraudster.
      (A:App)-[:developer]->(s)
   }
   Constraint {
      R1("It is a fraudulent application"): A.type == 'fraud'
   }
}
```

The association between concepts and entities can be described using expert rules, which helps address the problem of tight coupling between business and actual data. This approach avoids the need to modify underlying data when there are changes in the business requirements and reduces the direct perception of the changes in underlying data at the business application layer. By strongly binding concepts with the business, concepts can be used as types for the business users. Taking the example mentioned above, let's consider using "Fraudster" as a entity type. Here's an example of how it can be implemented:

```
MATCH
   (u:TaxonomyOfRiskUser/Fraudster)
RETURN
   u
```

Other scenarios for SPG-based risk mining applications.

1. Knowledge Query.

When a certain app is identified as a gambling application (possibly from user complaints or other security events), we can find the organization behind the app through the following query statement.

```
MATCH
    (a:App)-[:developer|boss]->(u:Person)
WHERE
   a.id = 'gambling application 1'
```

SPG enables adaptive risk detection through concept-driven classification, fraud detection using business logic rules, and risk-based knowledge queries.

SPG’s ability to model complex business relationships dynamically while ensuring scalability and adaptability for risk analysis and evolving enterprise needs.

Fusion Learning of Neural Symbols & Graph-Based Reasoning

```
RETURN
  u
```

2. Fusion learning of neural symbols

The fusion of deep learning and rules has always been a hot and challenging research topic. Deep learning can solve many representation learning problems, such as image classification tasks, while rules (symbolic logic) can handle many explicit reasoning problems. There are two main application directions for existing neural-symbol integration in reasoning.

1) Fusion methods of rules and neural networks.

From the perspective of rule priors, these methods can be classified into two categories:

First category: constraining model structure with rules. Typical methods include DeepProbLog [23], neuro-symbolic forward reasoning (NSFS) [24], Logical Neural Networks (LNN) [25], and LogicMP [26].

Second category: constraining objectives with rules. The rules are treated as prior knowledge and are incorporated into the objective function as a penalty term. Typical methods include SemanticLoss [27] and NCLF [28].

Whether it is the first category or the second category, the first-order predicate forms are required as the input form for rules. Various rules mentioned in this paper can be converted to and from the first-order predicate forms, for example.

```
Define (s:Person)-[p:belongTo]->(o:Fraudster) {
  Structure {
    (A:App)-[:developer]->(s)
  }
  Constraint {
    R1("It is a fraudulent application"): A.type == 'fraud'
  }
}
```

The conversion of first-order predicates is as follows:

```
forall s: exits a:  developer(a, s) & type(a) == 'fraud' -> belongTo(s) == Fraudster
```

In addition, the expertise of business experts, which belongs to hard rules, can easily lead to low recall rates. In LogicMP, specific rule content can be softened to improve rule coverage.

2) Representing relations between symbols as a graph structure and performing reasoning through graph algorithms.

In this example, the graphical form generated by logical rule definitions can be input into the graph algorithms for training. This approach decouples neural and symbolic methods through the form of a graph, ensuring scalability and flexibility.

SPG integrates deep learning and symbolic reasoning using neural-symbolic fusion techniques, enabling graph-based reasoning and improved scalability.

GCN (Graph Convolutional Network) is a deep learning algorithm designed to operate on graph-structured data. It is a type of neural network architecture that generalizes convolutional operations to graphs, enabling feature learning and representation learning from graph-based data.

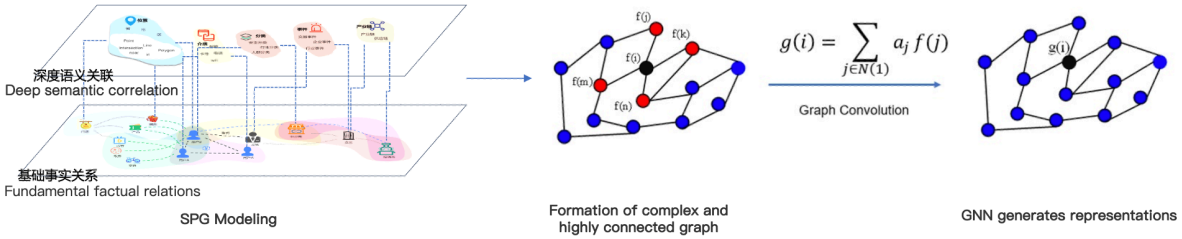

Figure 54: Combine the SPG with the GCN algorithm

9.4 Comparison between SPG and LPG in the Risk Mining Knowledge Graph

From the perspectives of knowledge construction, knowledge application, and knowledge evolution, we can compare the advantages and disadvantages of SPG and LPG.

Table 14: Comparison of abilities between SPG and LPG in the Risk Mining Knowledge Graph

Application Phase	Application Approach	LPG Solution	SPG Solution
Knowledge Construction	Import of vertices, edges, and properties	Provides mapping solutions, where mapping needs to be specified for vertices, edges, and properties	Uses semantic modeling instead of mapping, supports Table2SPG; uses logical properties/relations instead of importing derived data
	Data Consistency Maintenance	Defines schema constraints, typical solution is PG-Key	Maintains data consistency using logical dependencies, effectively reducing the risk of data inconsistency
	Graph Data Fusion	Not supported, can only import one graph or re-import	Supports cross-graph fusion, generates the required data for the current knowledge graph without additional data processing costs
Knowledge Application	Graph Analysis	Provides graph query language, but users need to write analysis language based on the underlying data	Provides graph query language, allows users to use logical relations and properties to shield complex underlying data, users only need to focus on the required information, reducing user learning curve and easier maintenance
Knowledge Evolution	Adding, deleting, or modifying the entity/relation classifications	Requires schema changes and data re-import, which incurs high costs for modifications	Based on the expressive power of concepts, changing concepts can achieve entity classification goals without the need for schema changes and data re-import.

SPG focuses on addressing the cost issue for users of the knowledge graph in the field of risk control. It effectively improves user efficiency and reduces usage costs in all stages of knowledge construction, knowledge application, and knowledge evolution.

9.5 Summary

This chapter primarily discussed the applications of SPG in the enterprise causal knowledge graph and the risk mining knowledge graph. In the enterprise causal knowledge graph, SPG provides an event expression framework that effectively describes the impact propagation relations of events, resulting in timely and effective conclusions, which complements the limitations of LPG in causal knowledge graph applications. In the risk mining knowledge graph, SPG primarily addresses the issues of user usage costs and efficiency, solving problems such as the difficulty of ensuring data consistency, high evolution costs, and high comprehension costs in practical applications of LPG.

SPG enhances risk mining efficiency by reducing costs, improving structured event reasoning, and overcoming the data consistency challenges of LPG.

SPG's strengths in combining symbolic and neural reasoning for scalable AI models and its advantages over LPG in cost-effective risk analysis.

Chapter 10 SPG Embracing the New Era of Cognitive Intelligence

The future trend of enterprise digitalization and intelligent upgrading is to build knowledge based on the massive business data accumulated in the enterprise's big data system, promoting data knowledgeization. By integrating business data with AI systems, business intelligence can be achieved. Property graphs have the advantage of compatibility with big data systems, and SPG, based on property graph, aims to accelerate the data knowledgeization and the organic integration of knowledge and AI systems. This chapter combines the core capabilities and two case studies introduced in previous chapters to summarize and analyze the strengths, limitations, opportunities, and challenges of SPG.

10.1 SWOT Analysis of SPG Compared to Property Graphs

Based on the previous descriptions, we can analyze the strengths, weaknesses, opportunities, and threats of SPG from the four quadrants of SWOT.

- **Strengths of SPG.** (1) SPG has a low cost and is compatible with big data architecture, allowing for the rapid construction of domain knowledge graphs based on structured data accumulated in enterprise-level applications. (2) The hierarchical semantic model of SPG supports the continuous evolution of incomplete knowledge graphs, meeting the requirements of rapid business deployment, continuous data accumulation and improvement, and gradual technological application in industrial applications. (3) SPG overcomes the semantic limitations of LPG and effectively connects big data with LPG node/edge structures, enhancing semantic connections and better integrating AI technologies.
- **Weaknesses of SPG.** SPG is still in the growth stage, and there are some compromises and weaknesses in the design of its capabilities, which we need to continuously overcome in the later stage. (1) How does dynamic classification achieve inheritance and extension? Currently, the dynamic classification model effectively addresses the granularity issue of types but has limitations in application, making it difficult to extend properties under new subclasses. (2) Continuous improvement of the built-in semantic structures of the entity is necessary. The current built-in semantic structures in the entity model are not sufficiently enriched, with definitions and constraints limited to event subjects/objects/times and hierarchical concepts. To meet the demands of controlled generation and interpretable reasoning, clear expression of built-in semantic structures within entities requires ongoing improvement in collaboration with downstream applications. (3) Instance-concept linkage reasoning model. While SPG possesses some inductive reasoning capabilities from instances to concepts, there is still significant room for improvement in the co-conduction of concepts and instances and the deductive reasoning from concepts to instances. This requires continuous optimization through the application and refinement of various causal knowledge graphs.

- **Opportunities for SPG.** (1) Filling the gap in semantic frameworks for enterprise knowledge graph applications. RDF/OWL, due to their complexity, have not effectively landed in enterprises. Establishing enterprise-level standardization to facilitate cross-entity knowledge semantic alignment, and to promote the circulation, interoperability, exchange, and sharing of knowledge more conveniently. (2) Driving the development of a universal engine architecture for building knowledge graphs. This promotes the democratization and accessibility of the knowledge graph technology. Large-scale applications in various technical fields rely on standardization and framework development, as seen in search engines, deep learning, and cloud computing. (3) Bridging the gap between knowledge graphs and LLMs. Enterprises can quickly incubate/fine-tune new pre-training models based on Transformer or open-source LLMs. Using the standardized symbols of SPG, efficient knowledge injection, associative prompts, knowledge queries, and other tasks can be achieved during the pre-training, SFT/RLHF, and inference stages, and forming a stable paradigm for the interaction between knowledge graphs and LLMs. Additionally, by data knowledgeization, we aim to construct a symbolic world domain knowledge system that complements and is equivalent to the neural network-based LLM knowledge system.
- **Challenges for SPG.** (1) Performance challenges in scaling applications, particularly during the knowledge construction phase. The performance overhead of extraction models and entity linking can significantly impact the efficiency of large-scale knowledge graph construction. (2) Further refinement of system capabilities. The capabilities of the SPG system need continuous optimization in conjunction with more business scenario and applications. (3) Cultivating semantic understanding in users' mental models. On one hand, there is a need to continuously improve users' understanding of semantics, and on the other hand, efforts should be made to reduce users' perception of semantics.

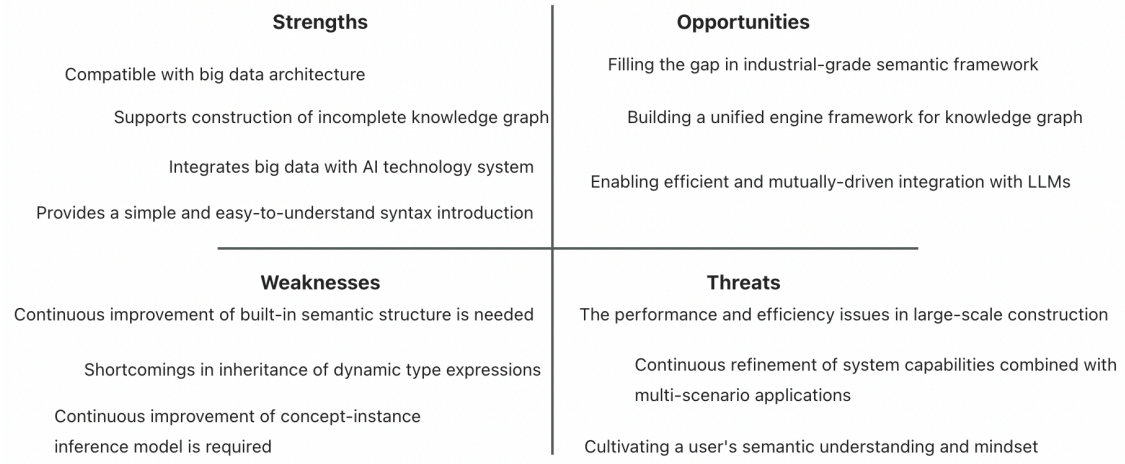

Figure 55: SWOT Analysis of SPG

10.2 Problem Resolution and Outstanding Issues from Chapter 2

Table 15: Fundamental Problems of Knowledge Management Based on LPG and the Resolution Status by SPG

Typical Problems of LPG		SPG Solutions	SPG Status
Type Management	Semantic deficiency in subjects	Event, entity, and concept classification	Supported
	Type granularity issue	Dynamic types	Supported
	Difficulty in property/relation selection	Standard properties, concept types	Supported
	Entities of the same type with different names	Knowledge fusion	Supported
Incompleteness Knowledge Graph Construction	Entity construction from multiple data sources	Entity linking, entity resolution	Partial support
	Knowledge reuse across multiple knowledge graphs	Knowledge fusion	Supported
Logical Dependencies	Semantic deficiency in logical predicates	Semantic predicates with rule conditions	Partial support
	Inconsistency due to separation of definition and logic	Logical Derivation	Supported
	Window-type property/relation explosion	Window standard types, variable parameter properties	Supported
	Inconsistent property logic	Logical properties	Supported
	Inconsistent relation logic	Logical relations	Supported
Logical reasoning of causal	Obstruction in event logic propagation	Activating new event instances during causal deduction	Supported
	Event Classification	Dynamic types	Supported
	Semantic Composition of Concepts	Dynamic composition based on event and entity facts	Supported

It is important to note that Table 15 primarily lists the fundamental problems of knowledge management based on LPG and the resolution status by SPG. It mainly focuses on the semantic aspects of entities and logical predicate semantics. The programmable framework and complex knowledge reasoning are built upon a knowledge management framework in a virtuous cycle. They are not included in the basic capabilities of knowledge management and are not listed in this table. However, they will be further described in the future release plan in Chapter 11.

Chapter 11 Outlook on the Future of SPG

This whitepaper has addressed the challenges faced by enterprise-level knowledge management and discussed the higher requirements for knowledge semantic representation and engine frameworks due to changes in demand paradigms in enterprise-level knowledge graph applications. In Chapter 1, we summarized some of the key problems that still exist in the development of knowledge graph technology:

- Lack of unified semantic representation. Currently, strong semantic knowledge graphs have not achieved industrial implementation based on RDF/OWL, while weak semantic property graphs (LPG) are widely used in industrial-grade knowledge graphs.
- Multiple tools but lack of standardization. The development of customized extraction algorithms/entity linking algorithms for each data set, graph database-backed graph storage, representation learning tools, fuzzy retrieval tools, knowledge query tools, and other tools have led to significant dispersion and inconvenience in the application of knowledge graph technology.

In order to achieve large-scale industrial application of any complex technology, it is necessary to have a unified technical framework that shields complex technical details and supports rapid deployment of new businesses. It also requires a modular architecture that allows for layering and decoupling of domain models and core engines, enabling fast migration to new domains. The same applies to knowledge graphs. The development of knowledge graph technology needs to keep up with the times. SPG defines an industrial-grade, user-friendly knowledge semantic framework for strong semantic knowledge graphs. It helps enterprises accelerate the knowledgeization of massive amounts of data. Through the unified technical framework and engine architecture provided by the SPG knowledge engine, the technology can be standardized, democratized, and made accessible to a wide range of users.

Looking towards the future, knowledge graphs have vast application potential. On one hand, as the best modeling practice for structured data, knowledge graphs can unify data modeling from various perspectives such as machines, algorithms, engineering, business, and operations. They can build next-generation data architectures in line with the concept of data fabric, accelerating the knowledgeization of massive enterprise data, connecting data silos, discovering implicit relations, unlocking the full value of data, and reducing the cost of finding and using data, ultimately bringing greater growth opportunities for businesses. On the other hand, knowledge graphs complement LLMs perfectly. Knowledge graphs have characteristics such as strong facts, weak generalization, strong interpretability, low computational cost, and high construction cost. In contrast, LLMs have weak facts, strong generalization, poor interpretability, high computational cost, and strong semantic understanding. In the future, the goal is to achieve efficient cooperation and complementarity between unified knowledge symbol representation and engine architecture and LLMs. Through the advancement of LLM technology, the cost of knowledge graph construction can be further reduced, accelerating data knowledgeization and providing additional domain knowledge for controlled generation based on LLMs. The construction of massive common-sense domain knowledge bases can also accelerate the progress of general artificial intelligence. Realizing the integration and complementarity of

the knowledge graphs and LLMs strongly relies on a comprehensive knowledge graph and LLM technology stack. Currently, LLM technology has matured, and with the strong semantic knowledge graph framework defined by SPG, it is expected to form a seamless application framework that can be seamlessly integrated with LLMs. This framework will enable industrial-level, generalizable, highly robust, and interpretable comprehensive artificial intelligence technologies based on knowledge graphs and LLMs.

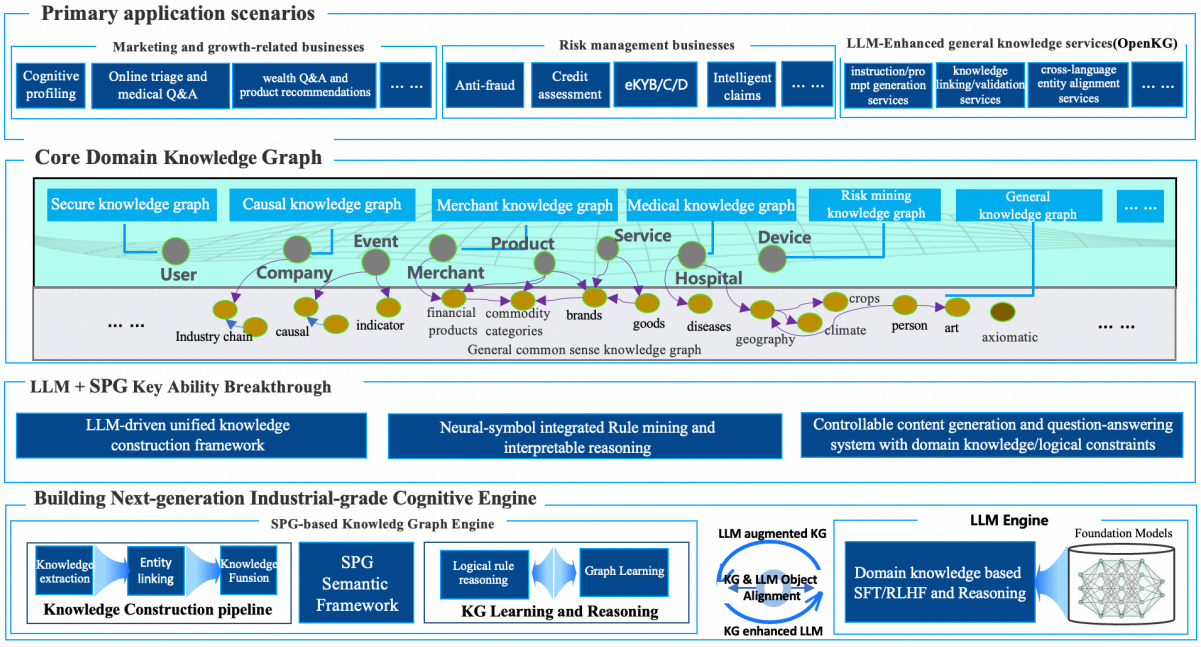

Figure 56: Future Outlook on the Dual-Drive Technical Paradigm of SPG and LLM

The fusion of symbolic logic and neural networks has always been a research hotspot in the industry. One common approach is to use neural networks to learn the rules and relations in symbolic logic, enabling them to better handle complex logical problems. Another approach is to use symbolic logic to guide the learning process of neural networks, improving their accuracy and interpretability. Knowledge graphs, as a typical representative of symbolic logic, have unique advantages in structural representation, semantic characterization, and knowledge association. The unified semantic framework provided by SPG can provide them with stronger vitality. Currently, the fusion of neural networks and symbolic logic mainly occurs in the knowledge reasoning stage. With the emergence of LLMs, new ideas are provided for the integration of symbolic logic and neural networks. On one hand, knowledge graphs, as the underlying support for semantic representation of symbolic logic and knowledge data management, can leverage the powerful semantic understanding capabilities of LLMs and the strong structure and semantics of knowledge graphs to automate prompts and sample construction. This can help the knowledge graphs form a unified knowledge extraction framework and accelerate the knowledgeization of data. On the other hand, in the content generation stage, applying domain knowledge data with strong semantic constraints can effectively avoid the problem of hallucinations and nonsense in LLMs. These issues are expected to be accelerated addressed in the SPG + LLM paradigm. We will continue to improve the expressive capabilities of SPG through industrial practices and enhance LLM through SPG to achieve alignment with objective facts, effectively avoiding/reducing model hallucinations. At the same time, LLM will also enhance SPG to improve the conversion efficiency

of data knowledgeization. We are committed to building a next-generation artificial intelligence engine driven and enhanced by both SPG and LLM.

Table 16: Future Release Plan for SPG

Stage	Release Content	Release Sub-items
Stage 1	"Semantic-Enhanced Programmable Knowledge Graph (SPG) White Paper" V1.0 ---- Next-generation industrial-grade knowledge semantic framework. Introduce the background, specifications, framework, and case studies of SPG knowledge semantic framework.	1.Overall framework of SPG 2.Two case studies with rich content 3.SPG-Schema Core 4.Basic requirements of SPG-Engine 5.Basic requirements of SPG-controller 6.Basic framework of SPG-Programming
	Establish SPG community and welcome its establishment	1.Establishment of community operation mechanism
	Open-source SPG Engine v1.0	1.Build an open-source engine that meets the requirements of White Paper 1.0 2.Create an open-source community based on the engine 3.Default adaptation to Tugraph technology stack, support vendor extensions
	Recommendation of standardization for SPG [Industry, National, International]	1.Contact various standardization organizations to promote standardization projects
Stage 2	"Semantic-Enhanced Programmable Knowledge Graph (SPG) White Paper" V2.0 ---- Next-generation industrial-grade knowledge semantic framework. Introduce SPG programmable framework Operator, SDK, case studies, etc., open up a complete knowledge construction framework.	1.Improved SPG-schema 2.Complete definition of operators and framework in SPG programmable framework 3.Support mapping-based [programmable] knowledge graph construction, support search engines, vector retrieval, model services, etc. 4.Basic inference solution, including basic query language for construction, query, and inference 5.Complete storage solution [standalone] 6.Basic requirements of SPG+LLM and exploration in knowledge extraction/NL2KGDSL
	Release a query language that meets semantic and programmable requirements	1.Release GQL or KGDSL that meets the basic requirements of SPG applications 2.CRUD operations in SPG 3.Support basic natural language queries
	Open-source framework V2.0	1.Complete open-source implementation of SPG framework including White Paper 2.0 2.Query language implementation module [GQL or KGDSL] 3.Programmable SDK
	Publish SPG industry and international standards	1.Finalize the standard draft 2.Publicize the standard draft
Stage 3	"Semantic-Enhanced Programmable Knowledge Graph (SPG) White Paper" V3.0 ---- Next-generation industrial-grade knowledge semantic framework. Introduce SPG inference engine, graph learning, rule-guided interpretable inference, etc., open up complete knowledge inference capabilities.	1.Programmable module to accumulate reusable model hubs, layer hubs, etc., supporting out-of-the-box use 2.SPG industry extensions for domains such as space-time, healthcare, etc. 3.Comprehensive syntax set for rule-based inference engine 4.SPG and AGL linkage to support knowledge inference, interpretable inference, etc.
	Open-source framework V3.0	1.Fully match the final standard draft and SPG V3.0 2.Extension modules
	Knowledge construction and open-source solutions based on common sense knowledge using SPG	Build an open-source, crowdsourced, service community for the healthy growth of common sense knowledge based on OpenKG
Stage 4	"Semantic-Enhanced Programmable Knowledge Graph (SPG) White Paper" V4.0 ---- Next-generation industrial-grade knowledge semantic framework. Introduce the paradigm of SPG + LLM double-driven and construct a controllable LLM technology system.	1.Complete paradigm and system framework of SPG + LLM 2.Basic natural language interface for construction, query, and inference 3.Natural language interaction module [based on LLM] 4.Supports extractive construction (based on LLM)
	Knowledge services for common sense knowledge based on SPG.	Build an enhanced knowledge services with LLMs, such as instruction/prompt generation services, entity linking/validation services, etc., to promote knowledge element exchange and growth.

In the future, we will continue to upgrade SPG. Table 16 represents our planned release content, and the release timeline will be updated on the SPG official account: “Semantic-Enhanced Programmable Knowledge Graph Framework”. We welcome your attention and interaction, and together, we can explore the industrial-grade knowledge graph architecture paradigm.