LLM-based Exploration of BIM Clash Information using Semantic Knowledge Graph

D. Shin[1], W. Lee[1], H. Lee[1], Y. Sung[1], W. Jeong[1], Y. Jeong[1], B. Koo[1*]

[1]Department of Civil Engineering, Seoul National University of Science and Technology, Republic of Korea
ehddnr567@seoultech.ac.kr, wonbok@seoultech.ac.kr, hyunwoo@seoultech.ac.kr, syj7593@seoultech.ac.kr, woosung@seoultech.ac.kr, yurim@seoultech.ac.kr, *bonsang@seoultech.ac.kr

Abstract –

BIM models are typically designed separately by discipline and integrated at later stages, a process that often results in numerous clashes. Although existing commercial BIM-based clash detection tools effectively identify physical interferences, they provide limited support for semantic interpretation and decision-making, often requiring practitioners to manually analyze large volumes of clash results.

To address these limitations, this study developed an LLMChain-based clash information exploration module that integrates a semantic knowledge graph with a Large Language Model (LLM). A clash-oriented semantic knowledge graph was constructed to explicitly represent element–space relationships and clash-related attributes, enabling semantic and spatial-level exploration of clash information.

Based on this graph, an LLMChain architecture was designed to decompose the reasoning process into user's query interpretation, Cypher generation, and response formulation, thereby improving logical consistency and mitigating hallucination issues.

Experimental validation showed that the proposed module achieved an executability of 0.92 and a semantic accuracy of 0.85, outperforming a single-prompt LLM by 0.22 and 0.19, respectively. These results indicate that combining graph-based semantic exploration with a structured LLM-based reasoning process effectively enhances the reliability of complex query processing and has strong potential to advance automation in BIM-based clash review.

Keywords –

BIM, Clash Information Exploration, Semantic Knowledge Graph, Large Language Model, Text-to-Cypher

1 Introduction

Building Information Modeling (BIM)-based design is typically conducted separately by discipline, such as architectural, structural, and mechanical, electrical, and plumbing. These discipline-specific models are integrated shortly before the construction phase, a process during which numerous clashes frequently occur. Such clashes are a major cause of design changes and schedule delays [1]. If these clashes are not adequately considered during the design stage, this can lead to deterioration in construction quality. Accordingly, BIM-based clash detection has been recognized as a critical process for identifying multidisciplinary clashes in advance and ensuring constructability during the design stage [2].

However, existing commercial BIM-based clash detection tools primarily identify physical clashes between elements, which limits their ability to provide semantic information such as clash types or severity levels. In real-world projects, thousands of clashes are often detected and presented at the same level, requiring practitioners to manually classify and assess a large volume of results [3]. Consequently, the level of automation in the clash review process remains low, making it difficult to ensure consistency and reproducibility in decision-making.

To address these limitations, recent studies have explored organizing clash-related information using semantic knowledge graphs. For example, [4] incorporated clash resolvability attributes into a semantic knowledge graph to enhance information utilization, while [5] proposed a framework for automatic clash classification by embedding criteria for clash types and severity into a semantic knowledge graph. Nevertheless, these studies mainly focused on representation and have not been sufficiently extended to support automatic exploration and utilization of clash information based on user-issued queries.

This study developed clash information exploration module that utilizes a Large Language Model (LLM) to translate user-issued natural language queries into a graph query language and automatically explore clash information stored in a knowledge graph. To this end, clash information was structured into a semantic knowledge graph, and an LLMChain architecture was designed by integrating a Text-to-Cypher (T2C) module

with a Question and Answer (QA) module.

The objective of this study is to minimize dependency on manual graph query formulation and to establish a framework that enables users to efficiently explore clash information using natural language inputs alone. The remainder of this paper is organized as follows. Section 2 reviews related studies. Section 3 describes the research methodology. Section 4 presents the development of the proposed semantic knowledge graph and exploration module. Section 5 validates the module through experiments.

2 Research Background

2.1 Current Status and Limitations of BIM-based Clash Review

BIM-based clash detection has been widely used as a core technique for improving design quality by identifying physical clashes between disciplines during the design stage. However, clashes encountered in practice differ in their impact on project quality and schedule, and some represent irrelevant clashes that do not require practical resolution.

Existing BIM-based clash detection tools are limited in their ability to distinguish the semantic importance or impact level of clashes. As a result, many clashes are presented at the same level, which reduces review efficiency and undermines the reliability of decision-making. Accordingly, recent studies have focused on classifying clashes based on their impact.

2.1.1 Clash Type and Severity Classification

Studies on clash type and severity classification aim to extend clash review beyond simple detection toward decision support. For example, [6] predicted clash relevance using attributes such as clash volume, location, and multidisciplinary relationships, while [7] analyzed penetrability based on clash images to identify irrelevant clashes that could be resolved through sleeve installation.

Other studies have focused on classifying clash severity by considering impact scope and resolution requirements. Specifically, [8] evaluated clash impact using discipline combinations and rule-based reasoning, and [1] categorized clash severity based on the need for design coordination and the feasibility of resolution during construction. These studies enable practitioners to better understand clash types and severity, thereby extending clash review from simple detection to decision-support-oriented processes.

2.1.2 Semantic Knowledge Graph–based Clash Representation

Semantic knowledge graphs have been adopted to represent clash information for structured management and reuse. By modeling elements, spaces, and clash relationships as nodes and edges, knowledge graphs support relationship-oriented storage and semantic exploration of clash information.

For example, [4] developed a semantic knowledge graph, referred to as ‘BIMClash’, by extracting element and clash information from BIM models. In BIMClash, node attributes represent element and space related information, including building type, system type, element type, discipline type, and bounding-box information. Edge attributes represent adjacency- and clash-related information, including clash distance, clearance, topology, clash type, severity, penetrability, and movability. Clash severity was classified into three levels (‘Major’, ‘Medium’, and ‘Minor’) based on design coordination requirements, element usability, and installation constraints.

Although such representations improve information organization, BIMClash primarily focused on element-to-element clash relationships, which limited support for spatial context-based exploration. In addition, effective use of the graph required manual formulation of graph query languages, constraining practical usability for non-expert users.

2.2 LLM-based Semantic Information Retrieval

Knowledge graph–based approaches have enabled structured management of clash information in BIM environments. However, extracting and interpreting relevant content from knowledge graphs in response to user queries remains challenging. To address this issue, recent studies have explored the use of LLMs for semantic information retrieval.

BIM data involve complex hierarchical structures and multiple relationships, which limit the ability of standalone LLMs to perform stable semantic interpretation and logical reasoning. Accordingly, there is a growing need for exploration frameworks that integrate LLMs with knowledge graphs.

2.2.1 Graph Retrieval Augmented Generation

Retrieval-Augmented Generation (RAG) improves the accuracy of LLM-generated responses by retrieving relevant information from external databases [9]. However, conventional RAG approaches primarily retrieve unstructured text, limiting their ability to support relational or hierarchical reasoning.

GraphRAG extends retrieval to graph databases, enabling semantic exploration and reasoning based on relational data represented by nodes and edges [10]. This makes GraphRAG well suited to the BIM domain, which involves complex relationships and hierarchical structures. For example, [11] transformed IFC data into graph structures and established Cypher query patterns.

[12] demonstrated that supplying subgraph information enables LLMs to reason about semantic relationships among BIM elements.

2.2.2 Text-to-Cypher

Querying graph databases requires explicit formulation of Cypher queries, the declarative query language used by Neo4j to specify graph patterns and relationships, which poses a significant entry barrier for non-expert users. To address this issue, T2C techniques have been proposed to automatically translate natural language queries into Cypher, with prior studies focusing on improving translation accuracy and executability.

[13] improved syntactic correctness using paired natural language–Cypher datasets, while [14] enhanced translation performance through a generation–verification framework that detects schema inconsistencies and query errors.

However, existing T2C studies largely focus on single-object queries or limited relational structures, which restrict their applicability to BIM-based clash review involving complex relationships and spatial context. In particular, applications of T2C techniques to BIM clash review and systematic performance validation remain limited.

Accordingly, this study proposed a clash information exploration module based on a semantic knowledge graph and an LLMChain architecture. Its applicability was evaluated using queries of complexity.

3 Research Methodology

This study developed an LLMChain-based clash information exploration module by integrating a semantic knowledge graph with an LLM to improve the efficiency of BIM-based clash review. As illustrated in Figure 1, the verall research process consists of three phases.

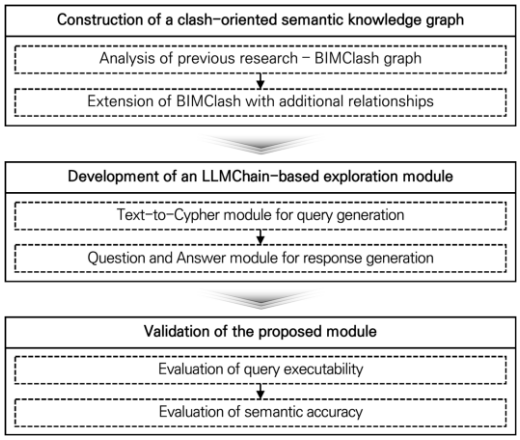

Figure 1. Research process

Using a multidisciplinary office-building IFC model, the proposed methodology was implemented through the following three phases: 1) construction of a clash-oriented semantic knowledge graph, 2) development of an LLMChain-based exploration module for natural language–driven clash information exploration, and 3) validation of the proposed module using a query complexity-based evaluation dataset.

4 Clash Information Exploration Module

4.1 Construction of a Clash-Oriented Semantic Knowledge Graph

Effective clash review requires consideration of not only element attributes and clash types but also the spatial context of clashes. Previous studies have incorporated spatial information into semantic knowledge graphs by prioritizing clashes based on spatial units and distribution [15]. However, such approaches often modeled element–space containment and clash relationships (‘PotentialClash’) [4], limiting the identification of space-level clash locations.

To address this limitation, this study constructed a clash-oriented semantic knowledge graph based on an office-building IFC model. Element nodes were instantiated from discipline-specific IFC entities, whereas space nodes were instantiated from *IfcSpace* entities. Element–space containment was represented through ‘SpatialContainment’, while clash interactions between elements were represented through ‘PotentialClash’. The containing space of each clash was additionally recorded as an attribute of ‘PotentialClash’ to enable space-level tracking.The graph was constructed through a four-step pipeline: (1) ontology schema definition in Protégé and implementation in Neo4j; (2) extraction of IFC entities and GUIDs using IfcOpenShell to instantiate nodes; (3) Axis-Aligned Bounding Box (AABB)-based geometric processing to derive inter-object relationships, where discipline-specific proximity rules were applied to define ‘PotentialClash’ and ‘SpatialContainment’ edges; and (4) integration of clash-related attributes through GUID-based matching with clash-detection outputs.

Clash data were obtained using Autodesk Navisworks, where hard clashes were defined as direct geometric intersections and soft clashes were identified based on discipline-specific clearance thresholds. A zero-tolerance criterion was applied to structure–MEP clashes, whereas a 10 mm tolerance was applied to other discipline combinations. Additional attributes, including offset, topology, and clash volume, were derived through Python-based geometric analysis.

As shown in Figures 2 and 3, this structure separates containment and clash relationships while enabling

spatially aware clash representation.

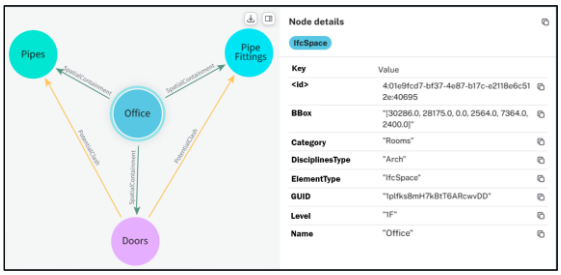

Figure 2. Result sample of node attributes

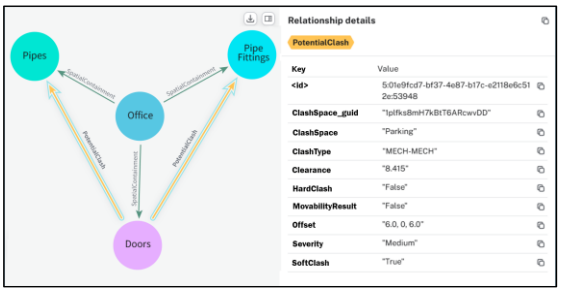

Figure 3. Result sample of edge attributes

4.2 LLMChain Architecture for Clash Information Exploration

This study employs an LLMChain-based exploration module to efficiently retrieve clash information from BIM models using a semantic knowledge graph. To mitigate the logical complexity and hallucination issues commonly observed in single-prompt LLM approaches, the exploration workflow is explicitly decomposed into sequential stages for user's query interpretation, Cypher generation, and response formulation.

The proposed LLMChain architecture consists of two functional sub-modules (T2C module and QA module) and three LLM components, each assigned a clearly defined role (Figure 4). The overall workflow comprises the following stages: 1) T2C module, and 2) QA module.

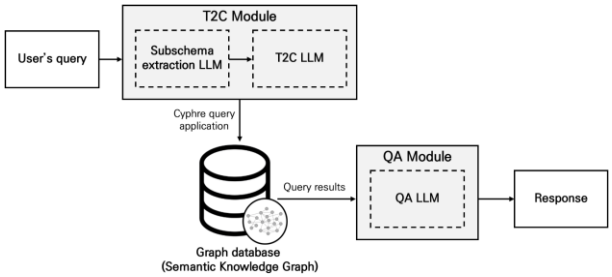

Figure 4. LLMChain-based clash information exploration module process

1) T2C module

The T2C module is responsible for translating a user's natural language query into an executable Cypher query. Within this module, a schema extraction LLM first identifies a relevant subschema by extracting key domain concepts and attribute constraints from the query. Based on the extracted subschema, the T2C LLM then generates a Cypher query that accurately reflects the structure of the semantic knowledge graph.

2) QA module

The QA module receives the results returned from the graph query and generates a natural-language response by summarizing and interpreting the retrieved clash information in alignment with the user's query intent.

By structuring the exploration workflow into these two sequential modules, the proposed LLMChain architecture maintains semantic consistency throughout the reasoning process, reduces hallucination effects, and improves the coherence and reliability of the generated responses.

4.3 Implementation of the Clash Information Exploration Module

4.3.1 T2C Module

The T2C module was developed to reduce the entry barrier associated with manual graph query formulation and to enable efficient clash information exploration from natural language queries. The module was implemented using OpenAI GPT-4o within the LangChain framework as a two-stage process separating subschema extraction and Cypher generation (Figure 4).

In the subschema extraction stage, a schema extraction LLM identifies key domain concepts and attribute constraints from the user's natural language query and selectively extracts relevant nodes, relationships, and attributes from the complete knowledge graph schema. This step limits unnecessary schema references and reduces the risk of syntactic errors and schema mismatches.

In the Cypher generation stage, the T2C LLM generates an executable Cypher query based on the extracted subschema. Explicit mapping rules are embedded in the prompt to ensure consistent alignment between query terms and graph schema elements, while restricting query construction to subschema components only. Query results are returned in JSON format to support seamless downstream processing.

LLMs may fail to fully interpret user intent or omit certain conditions when handling complex queries composed of multiple constraints [16]. To address this limitation, this study focused on composite queries combining discipline, severity, and spatial conditions and applied few-shot learning to improve both Cypher generation and response performance.

4.3.2 QA Module

The QA module was developed to transform graph query results into intuitive natural-language responses. Operating within the same LLMChain framework, it receives query results in JSON format and generates responses aligned with the user's intent.

To support decision-making in clash review, the QA module is designed to interpret semantic relationships among clash attributes rather than merely listing retrieved values. Definitions and interpretation rules for key domain attributes are explicitly embedded in the prompt to guide coherent reasoning. In particular, movability and penetrability are defined as core criteria for assessing clash resolvability, with a clash considered resolvable if at least one of these conditions is satisfied.

To mitigate hallucination risks, response formulation is constrained to the retrieved results, the user query, and the extracted subschema, preventing inference beyond available information.

4.3.3 Module Integration and Exploration Workflow

The developed module operates in a sequential manner, as illustrated in Figure 5.

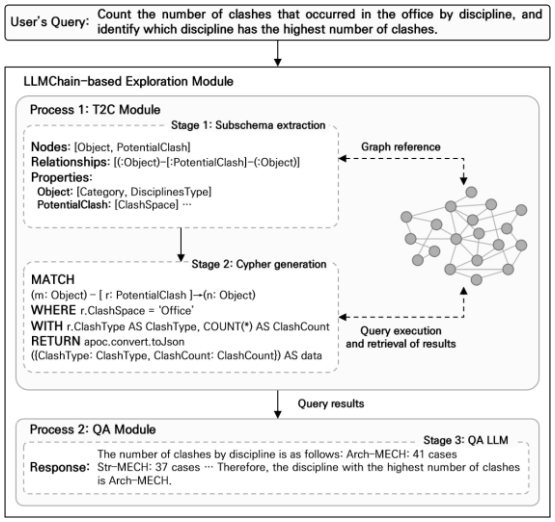

Figure 5. Example of workflow of the clash information exploration module

When a user submits a natural language query, such as "Count the number of clashes that occurred in the office by discipline and identify which discipline has the highest number of clashes," the module first interprets the semantic intent of the query and then performs stepwise exploration of clash information based on the semantic knowledge graph.

The T2C module extracts key semantic elements from the user query, including spatial information, elements, and clash-related attributes, and maps them to the semantic knowledge graph schema to identify the corresponding subschema. Based on the identified subschema, an executable Cypher query is automatically generated and executed to explore the graph, and the query results are returned in JSON format. The retrieved results are then passed to the QA module, which generates a natural-language response by semantically interpreting clash severity and resolvability.

5 Validation and Performance Evaluation

5.1 Validation Dataset and Evaluation Metrics

To validate the performance of the proposed LLMChain-based exploration module, a single-prompt LLM was used as the baseline. The validation dataset was constructed from the clash-oriented semantic knowledge graph and included ground-truth answers for each query.

The dataset comprised 40 natural language queries reflecting common BIM-based clash review tasks, including retrieving clash counts, identifying discipline-specific clashes, filtering by spatial locations, and assessing severity and resolvability. The queries were formulated through interviews with BIM practitioners to ensure practical relevance and were evaluated using clash data extracted from an IFC-based BIM model.

To reflect increasing reasoning complexity, the queries were categorized into three types. Simple queries involve one to two attribute constraints. Composite queries require logical combinations of multiple conditions using operators such as AND/OR. Aggregate queries involve attribute aggregation or statistical operations (e.g., COUNT, MAX). Based on this categorization, the dataset included 15 simple queries, 15 composite queries, and 10 aggregate queries. Table 1 presents representative queries for each query type.

Table 1. Representative query examples by query type

Query type	Representative query
Simple	Find clashes classified as Major in the second-floor office (room 202).
Composite	Summarize clashes related to the structural discipline within the second-floor office (room 202) that are either classified as Major or are non-movable.
Aggregate	Count the number of clashes that occurred in the office by discipline and identify which discipline has the highest number of clashes.

Performance was evaluated using two metrics: executability and semantic accuracy.

1) Executability

Query executability was defined as the proportion of generated Cypher queries that could be executed in the Neo4j graph database without errors. For each query, execution success was evaluated using a binary criterion

(success = 1, failure = 0), and average executability scores were calculated for each query type.

2) Semantic Accuracy

Semantic accuracy was defined as the consistency between generated responses and ground-truth answers. Given the limitations of conventional metrics such as BLEU [17] and ROUGE [18] in semantically complex domains, this study adopted an LLM-as-a-Judge approach [19].

Evaluation based solely on query execution is insufficient to capture response-level errors (e.g., omitted conditions, semantic misinterpretation, incorrect aggregation). Accordingly, GPT-4o was used to perform binary judgments based on three inputs: (1) the user query, (2) the ground-truth answer, and (3) the model-generated answer.

A response was considered correct if it was semantically consistent with the ground truth and satisfied the query intent. For aggregate queries, correctness required exact agreement with the ground-truth results (e.g., counts, grouped outputs, and dominant categories). Responses were marked incorrect if any required information was omitted, contradicted, or numerically inconsistent, including partially correct outputs.

To reduce variability in LLM-based evaluation, each response was evaluated three times and the average score was used as the final semantic accuracy. The evaluation prompt enforced strict logical consistency and avoided lenient interpretation, thereby mitigating potential bias in LLM-based judgment. A standardized evaluation prompt was used to ensure consistent judgment criteria across all evaluations and is available upon request for reproducibility. This protocol ensures consistent handling of borderline cases, particularly for partially correct or aggregation-based responses.

5.2 Results and Analysis

This section analyzes the performance of the proposed module in comparison with the single-prompt LLM (GPT-4o). The evaluation was conducted using the executability and semantic accuracy metrics defined in Section 5.1.

5.2.1 Quantitative Performance Comparison

The validation results showed that the proposed module consistently outperformed the single-prompt LLM across all query types. Table 2 presents the comparison results for executability.

Table 2. Comparison of query executability

Query type	Single prompt LLM (baseline)	LLMChain module	Variation (LLMChain module – baseline)
Simple	0.80	1.00	▲ 0.20
Composite	0.60	0.87	▲ 0.27
Aggregate	0.70	0.90	▲ 0.20
Average	**0.70**	**0.92**	**▲ 0.22**

The LLMChain module achieved an average executability of 0.92, representing an improvement of 0.22 over the single-prompt LLM baseline, which recorded a score of 0.70. For simple queries, all Cypher queries were executed without errors, resulting in a perfect executability score of 1.00. This performance was attributed to the subschema extraction stage of the T2C module, which effectively suppressed unnecessary entity references and syntactic errors.

For composite and aggregate queries, executability slightly decreased to 0.87 and 0.90, respectively, due to increased query complexity. Nevertheless, these values still represent improvements of 0.27 and 0.20 over the baseline. These results suggest that the LLMChain architecture mitigates error propagation by separating user query interpretation and Cypher generation into sequential stages. In addition, few-shot learning helped maintain consistent Cypher generation patterns.

Table 3 presents the comparison results for semantic accuracy across different query types. The LLMChain module achieved an average semantic accuracy of 0.85, representing an improvement of 0.19 over the baseline. The most substantial improvement was observed for composite queries, which can be attributed to the explicit incorporation of domain-specific attribute interpretation criteria in the QA module prompt. This design enabled the LLM to interpret semantic relationships among attributes more reliably during response generation.

Table 3. Comparison of semantic accuracy

Query type	Single prompt LLM (baseline)	LLMChain module	Variation (LLMChain module – baseline)
Simple	0.80	0.93	▲ 0.13
Composite	0.56	0.80	▲ 0.24
Aggregate	0.62	0.81	▲ 0.19
Average	**0.66**	**0.85**	**▲ 0.19**

A runtime comparison was conducted to evaluate the computational cost of the baseline and proposed modules. In the implemented setting, the baseline used a single LLM call per query, whereas the proposed LLMChain module required three sequential LLM calls for subschema extraction, Cypher generation, and response generation. End-to-end latency, including graph-query execution and LLM inference, was measured from user-query submission to final response generation. The baseline achieved an average latency of 4.8 s/query, whereas the proposed module required 11.6 s/query.

These results indicate that the staged exploration

structure increases runtime due to multiple LLM calls. However, this overhead should be interpreted together with the improvements in executability and semantic accuracy. In practical BIM coordination workflows, such response times remain acceptable for interactive query-based exploration, where reliability is prioritized over real-time response.

5.2.2 Error Analysis of Composite Queries

To further analyze the causes of the observed performance improvements, a representative composite query was selected, and the Cypher queries and final responses generated by the single-prompt LLM and the proposed LLMChain module were compared. The selected query involved multiple logically combined conditions, such as "summarize clashes related to the structural discipline within the second-floor office that are either classified as Major or are non-movable," as illustrated in Figure 6.

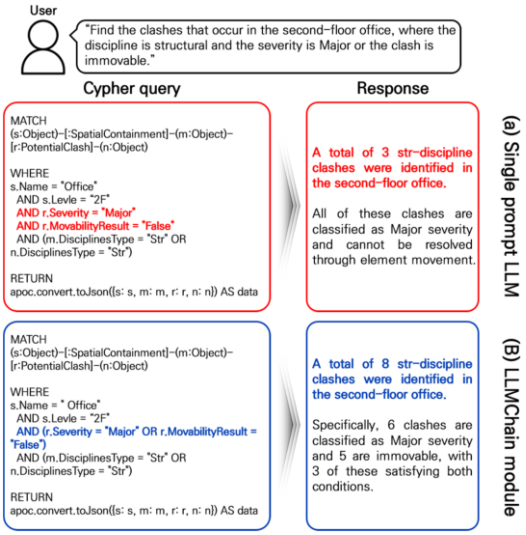

Figure 6. Example of Cypher generation

As shown in Figure 6(a), the single-prompt LLM generated a semantically incorrect Cypher query. Because query interpretation and Cypher generation were performed simultaneously within a single step, individual conditions were recognized, whereas their logical relationships were not explicitly structured. Consequently, the condition "clashes with Major severity or non-movable elements" was incorrectly constrained using an AND operator instead of an OR operator, resulting in the expression *r.Severity = "Major" AND r.MovabilityResult = "False"*. This incorrect constraint excluded valid clashes that satisfied only one of the specified conditions, leading to an underestimation of the total number of relevant clashes.

In contrast, as shown in Figure 6(b), the proposed module correctly handled logical relationships through structured reasoning based on the semantic knowledge graph. During the subschema extraction stage, semantic elements contained in the user query were systematically identified, and only the relevant nodes, relationships, and attributes defined in the semantic knowledge graph schema were selected. Based on this structured interpretation, the Cypher generation stage explicitly reflected logical operators, resulting in a query formulated as *r.Severity = "Major" OR r.MovabilityResult = "False"*.

Through this process, logical relationships among composite conditions were accurately reflected in the exploration results, ensuring consistency with the relational structure of the semantic knowledge graph. Consequently, the T2C module generated results aligned with the user's intent, and the QA module produced responses with high semantic accuracy. These findings demonstrate that the proposed architecture effectively mitigates logical distortions and condition omissions in complex queries, thereby improving the reliability and semantic accuracy of generated responses.

6 Conclusion

This study proposed an LLMChain-based clash information exploration module that integrates a semantic knowledge graph to enable efficient exploration of BIM clash information. By decomposing the reasoning process into sequential stages—query interpretation, Cypher generation, and response formulation—the architecture mitigates issues of logical complexity and hallucination observed in single-prompt LLM approaches.

The validation results showed that the proposed module outperformed a single-prompt LLM in terms of query executability and semantic accuracy. In particular, robust performance was observed for composite and aggregate queries, highlighting the effectiveness of combining a semantic knowledge graph with a structured LLM-based reasoning process.The results confirm the feasibility of supporting space-level exploration and complex clash queries through natural-language interaction using a semantic knowledge graph that incorporates element–space relationships and clash-related attributes. This capability is difficult to achieve with conventional BIM-based clash review tools and enables intuitive exploration without requiring manual graph query formulation.

However, the generalizability of the proposed module should be interpreted with appropriate caution. The validation was conducted using an office-building IFC model and a single LLM (GPT-4o), and therefore reflects feasibility within this specific setting rather than universal applicability.

Nevertheless, the proposed framework is not

inherently restricted to a specific building type or model configuration, as it is based on an IFC-driven semantic knowledge graph and a modular, schema-guided workflow. This design enables adaptation to other building types and LLMs through reconfiguration of graph-construction rules, domain taxonomies, and prompting strategies, providing a transferable framework rather than a model-specific solution.

Future work will focus on validating the proposed module across diverse BIM contexts and LLM settings. In addition, the LLMChain architecture will be enhanced by expanding few-shot examples and incorporating a query-validation stage to improve reasoning reliability. The approach will also be extended toward an agentic AI-based collaborative framework.

7 Acknowledgement

This research was supported by the "Research of Construction Codes Library and Ontology for Digital Transformation of Construction Codes" project, part of the "Research on the Development of Construction Standards Operated by the Korea Construction Standards Center" project of the Korea Institute of Civil Engineering and Building Technology (KICT).

References

[1] W. B. Lee. A study on automated clash type classification of building information modeling based on artificial intelligence. Master's thesis, Seoul National University of Science and Technology, 2024.

[2] R. Chahrour, M. A. Hafeez, A. M. Ahmad, H. I. Sulieman, H. Dawood, S. Rodriguez-Trejo, M. Kassem, K. K. Naji, and N. Dawood. Cost-benefit analysis of BIM-enabled design clash detection and resolution. Construction Management and Economics, 39(1):55–72, 2021.

[3] B. Akhmetzhanova, A. Nadeem, M. A. Hossain, and J. R. Kim. Clash detection using building information modeling (BIM) technology in the Republic of Kazakhstan. Buildings, 12(2):102, 2022.

[4] Y. S. Yu. Employing semantic knowledge graphs and artificial intelligence models for automated clash detection and severity classification in multidisciplinary BIM models. Ph.D. dissertation, Seoul National University of Science and Technology, 2024.

[5] D. U. Shin, Y. S. Yu, W. B. Lee, H. W. Lee, and B. Koo. Semantic knowledge graph construction and utilization for clash type and severity classification in federated BIM models. KIBIM Magazine, 15(3):44–56, 2025.

[6] Y. Hu and D. Castro-Lacouture. Clash relevance prediction based on machine learning. Journal of Computing in Civil Engineering, 33(2):04018060, 2019.

[7] H. W. Lee, Y. S. Yu, W. B. Lee, and B. Koo. Comparative study on the performance of multi-view image learning-based penetrability analysis model for classifying irrelevant clashes in BIM model. Korean Journal of Construction Engineering and Management, 26(5):26–39, 2025.

[8] Y. C. Lin, S. H. Chou, and I. C. Wu. Conflict impact assessment between objects in a BIM system. In Proceedings of the 30th International Symposium on Automation and Robotics in Construction (ISARC), pages 337–344, Montréal, Canada, 2013.

[9] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-T. Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems, 33:9459–9474, 2020.

[10] L. Bahr, C. Wehner, J. Wewerka, J. Bittencourt, U. Schmid, and R. Daub. Knowledge graph enhanced retrieval-augmented generation for failure mode and effects analysis. Journal of Industrial Information Integration, 45:100807, 2025.

[11] J. Zhu, N. Nisbet, M. Yin, R. Wei, and I. Brilakis. Cypher4BIM: releasing the power of graph for building knowledge discovery. arXiv preprint arXiv:2405.16345, 2024.

[12] S. Iranmanesh, H. Saadany, and E. Vakaj. LLM-assisted Graph-RAG information extraction from IFC data. arXiv preprint arXiv:2504.16813, 2025.

[13] M. G. Ozsoy, L. Messallem, J. Besga, and G. Minneci. Text2Cypher: bridging natural language and graph databases. arXiv preprint arXiv:2412.10064, 2024.

[14] A. Tiwari, S. K. Reddy, V. Yadav, M. Hashemi, and S. T. Madhusudhan. Auto-Cypher: improving LLMs on Cypher generation via LLM-supervised generation-verification framework. arXiv preprint arXiv:2412.12612, 2024.

[15] Y. Hu and D. Castro-Lacouture. Intelligent clash detection in building information modeling. In Research Companion to Building Information Modeling, pages 230–247. Edward Elgar Publishing, 2022.

[16] Y. Hao, H. Yu, and J. You. Beyond facts: evaluating intent hallucination in large language models. arXiv preprint arXiv:2506.06539, 2025.

[17] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: a method for automatic evaluation of machine translation. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pages 311–318, Philadelphia, Pennsylvania, USA, 2002.

[18] C.-Y. Lin. ROUGE: a package for automatic evaluation of summaries. In Text Summarization Branches Out, pages 74–81, Barcelona, Spain, 2004.

[19] L. Zheng et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. arXiv preprint arXiv:2306.05685, 2023.