Knowledge-Informed Semantic Alignment and Rule Interpretation for Automated Compliance Checking

Zhe Zheng[1], Yu-Cheng Zhou[1], Xin-Zheng Lu[1], Jia-Rui Lin[1,*]

1. Department of Civil Engineering, Tsinghua University, Beijing, 100084, China

*Corresponding author, E-mail: lin611@tsinghua.edu.cn

Abstract:

As an essential prodecure to improve design quality in the construction industry, automated rule checking (ARC) requires intelligent rule interpretation from regulatory texts and precise alignment of concepts from different sources. However, there still exists semantic gaps between design models and regulatory texts, hindering the exploitation of ARC. Thus, a knowledge-informed framework for improved ARC is proposed based on natural language processing. Within the framework, an ontology is first established to represent domain knowledge, including concepts, synonyms, relationships, constraints, etc. Then, semantic alignment and conflict resolution are introduced to enhance the rule interpretation process based on predefined domain knowledge and unsupervised learning techniques. Finally, an algorithm is developed to identify the proper SPARQL function for each rule, and then to generate SPARQL-based queries for model checking purposes, thereby making it possible to interpret complex rules where extra implicit data needs to be inferred. Experiments show that the proposed framework and methods successfully filled the semantic gaps between design models and regulatory texts with domain knowledge, which achieves a 90.1% accuracy and substantially outperforms the commonly used keyword matching method. In addition, the proposed rule interpretation method proves to be 5 times faster than the manual interpretation by domain experts. This research contributes to the body of knowledge of a novel framework and the corresponding methods to enhance automated rule checking with domain knowledge.

Keywords:

automated rule checking (ARC); automated compliance checking (ACC); rule interpretation; natural language processing (NLP); semantic alignment; knowledge modeling; building information modeling (BIM)

1 Introduction

Building regulatory documents such as design guidelines, codes, standards, and manuals of practice stipulate the safety, sustainability, and comfort of the entire lifecycle of a built environment (Nawari, 2018). For a long time, the regulation compliance of a building design was checked manually by domain experts. However, this process is not only immensely time-consuming but is also challenging for project participants (Lee et al., 2015) due to the complexity and ambiguity of rules and regulations (Lee et al., 2020). Thus, extensive human knowledge is needed. Additionally, subjective judgment is always inevitable in this process, making it impossible to handle massive and heterogeneous building information and regulations effectively and sustainably (Xu & Cai, 2021). Therefore, an automated compliance checking (ACC) method is critical for a more efficient and consistent checking procedure.

To promote the compliance checking process in the architecture, engineering, and construction (AEC) industry, automated rule checking (ARC, also known as ACC) has been extensively studied by many researchers, and several ARC systems have been established. (Eastman et al., 2009; Beach et al., 2020; Ismail et al., 2017; Fuchs, 2021). The rule checking process can be broadly structured into four stages: 1) rule interpretation, which translates rules represented in natural language into a computer-processable format, 2) building model preparation, which prepares the required information for the checking process, 3) rule execution, which implements the prepared model checking using computer-processable rules, and 4) reporting checking results (Eastman et al., 2009; Ismail et al., 2017). Among the four stages, rule interpretation is the most vital and complex stage that needs more investigation (Ismail et al., 2017).

Most of the existing ARC systems, including CORENET emerging from the first large effort of ARC in Singapore and the widely used Solibri Model Checker (SMC), are usually based on hard-coded or manual rule interpretation methods (Eastman et al., 2009). Sometimes, they are referred to as 'Black Box' approaches because it is difficult to maintain and modify hard-code rules efficiently (Dimyadi et al., 2016; Nawari, 2019). Therefore, semiautomated and automated methods for interpreting the regulation text into a computer-processable format have been proposed to achieve a flexible, transparent, and portable ARC process. In the beginning, semiautomated rule interpretation methods, such as the RASE methodology (Hjelseth & Nisbet, 2011) were introduced to help AEC experts analyze the semantic structure of regulatory requirements, and document markup techniques were used to annotate different components of a regulatory requirement. However, these methods just process text at a coarse granularity level and require considerable human effort to mark regulation documents and create queries or pseudocodes from them. For this reason, natural language processing (NLP), a widely used method for processing and understanding text-based human language (Fuchs, 2021), has been introduced to automate rule interpretation from regulatory documents. Generally, the NLP-based method for automated rule interpretation consists of two tasks: 1) information extraction, which extracts semantic information from textual building regulatory documents automatically (Zhang & El-Gohary, 2015; Zhou & El-Gohary, 2017), and 2) information transformation, which transforms the extracted information into logic clauses to support reasoning in compliance checking (Zhang & El-Gohary, 2016; Xu & Cai, 2021). Chaining the information extraction and information transformation tasks together has enormously facilitated the automated rule interpretation process. To enhance NLP-based methods for different tasks in rule interpretation, techniques, such as ontology, machine learning, and deep learning are introduced to achieve a more comprehensive understanding of regulatory texts (Fuchs, 2021).

Although many efforts have been made for automated rule interpretation, there are still some research gaps. First, as pointed out by some researchers (Dimyadi et al., 2016; Zhou & El-Gohary, 2021), to achieve fully automated rule checking, both the regulations and the BIM models should use the same concepts and relations to define and describe the same objects, i.e., the two representations should speak "the same language". However, different terms and modeling paradigms may be adopted when dealing with regulatory documents and BIM models. Therefore, an extra step called semantic alignment is needed to determine correspondences between different concepts and relations in regulations and BIM models. In previous studies, the methods for aligning the concepts and relations in computer-interpretable regulations with those in ontologies or BIMs were mainly based on keyword matching (Zhang & El-Gohary, 2015; Zhou & El-Gohary, 2017) or handcrafted mapping tables (Xu & Cai, 2021). Researchers may make a huge effort to construct mapping tables containing all aliases, abbreviations, alternate spellings, etc. For instance, designers may use the alias "*fire partition wall*" to represent the concept "*firewall*". If the alias of a concept is not contained in the mapping table, then the semantic alignment process will fail. Therefore, a more robust and flexible method for automated semantic alignment is required. Second, when mapping the extracted information into plain description logic clauses, such as the Horn clause (Zhang & El-Gohary, 2016), the B-Prolog representation (Zhang & El-Gohary, 2015; Zhou & El-Gohary, 2017), or the deontic logic (DL) clauses (Xu & Cai, 2021), hidden domain knowledge is required to infer implicit relationships, data types, and properties. Due to a lack of domain knowledge, many queries in the AEC domain are hampered by implicit required relationships and properties that are difficult to retrieve (Zhang et al., 2018). It is important to incorporate domain knowledge into the rule interpretation process. In addition, the widely used B-Prolog representation and DL clauses have difficulty dealing with implicitly defined quantity information. For example, information such as room A containing safe exit B and safe exit C is explicitly stored in a BIM model, while the number of safe exits contained in room A is implicit information that requires additional calculations. Therefore, the logical representation of complex rules or high-order information is needed.

To address the abovementioned problems, this research proposes a novel knowledge-informed semantic alignment and rule interpretation framework for ARC. Domain knowledge, such as equivalent concepts, constraints, and relationships are modeled based on ontology, which are further utilized in three folds to enhance the ARC processes: 1) automated alignment of regulations concepts with those in ontology based on semantic similarity, 2) improving the accuracy of rule interpretation by detecting and resolving potential conflicts of rules, and 3) automated generating complex rules with implicit information based on SPARQL.

The remainder of this paper is organized as follows: Section 2 reviews the related studies and highlights the potential research gaps. Section 3 describes the knowledge-informed framework for ARC proposed in this research. The proposed algorithms and methods are also explained in this section. Section 4 illustrates and analyzes the results of the experiments. Section 5 demonstrates the application of this framework. Section 6 discusses the advantages and contributions of this research, while also noting the limitations. Finally, Section 7 concludes this research.

2 Overview of related studies

2.1 NLP-based automated regulatory document interpretation

ARC methods and systems have been extensively studied in recent decades with the increasing maturity and adoption of BIM (Beach et al., 2015; Beach et al., 2020) since decision tables were first utilized to check structural design against building codes (Fenves, 1966). However, for the existing ARC systems, intensive manual efforts are still needed in the rule interpretation stages (Eastman et al., 2009). Therefore, to make the rule interpretation stage more efficient and transparent, numerous automated rule interpreting methods have been proposed.

In recent years, NLP algorithms that can achieve a comprehensive understanding of a text (Fuchs, 2021) have been utilized to develop automated rule interpreting methods. NLP algorithms can be mainly divided into two approaches: the handwritten rules approach and the statistical approach (Nadkarni et al., 2011). The former depends on symbolic human-defined pattern-matching rules, including Backus-Naur Form (BNF) notation, regular expression syntax, and Prolog language. The latter automatically builds probabilistic 'rules' from training datasets (e.g., large annotated bodies of text) similar to machine-learning algorithms, including support vector machines (SVMs), hidden Markov models (HMMs), conditional random fields (CRFs), and naïve Bayes (Nadkarni et al., 2011). In the AEC industry, Zhang & El-Gohary pointed out that domain-specific regulatory text is more suitable for automated NLP than general nontechnical text (e.g., news articles, general websites) because the regulatory text has fewer homonym conflicts and fewer coreference resolution problems. Additionally, the ontology for one specific domain is easier to develop as opposed to one that captures general knowledge (Zhang & El-Gohary, 2016). Domain ontology is able to provide a shared vocabulary among the parties by defining abstract concepts and relationships, including the taxonomy of concepts, equivalent and disjoint concepts, and enumerations of terminologies. As such, data is given explicit meaning, making it easier for machines to automatically process and integrate data through ontology (Xu & Cai, 2020). A domain ontology may enhance the automated interpretability and understandability of domain-specific text (Zhang & El-Gohary, 2016). Therefore, in the AEC industry, many studies in terms of NLP-based automated rule interpreting methods have adopted rule-based and ontology-based approaches (Fuchs, 2021). For example, Zhang & El-Gohary (Zhang & El-Gohary, 2015; Zhang & El-Gohary, 2016) proposed an automated rule interpretation method mainly consisting of three processing phases: 1) text classification, which recognizes relevant sentences in a regulatory text corpus, 2) information extraction, which extracts the words and phases in the relevant sentences based on pattern-matching-based information extraction rules and ontology. 3) information transformation, which transforms the extracted information into the Horn clauses or the B-Prolog representation using regular expression-based mapping rules. Zhou & El-Gohary (Zhou & El-Gohary, 2017) proposed a rule-based ontology-enhanced information extraction method for extracting building energy requirements from energy conservation codes, and then formatted the extracted information into a B-Prolog representation. Zhou et al. (Zhou et al., 2022) proposed a novel automated rule extraction method based on a deep learning model and a set of context-free grammars (CFGs), which can interpret regulatory rules into pseudocode formats without manual intervention, increasing generality, accuracy, and interpretability. Xu and Cai (Xu & Cai, 2021) proposed an ontology and rule-based NLP framework to automate the interpretation of utility regulations into deontic logic (DL) clauses, where pattern-matching rules included encoded information extraction. Pre-learned model and domain-specific hand-crafted mapping methods were adopted in semantic alignment between rule and ontology.

2.2 Semantic alignment methods for ARC

Most of the studies focusing on information extraction assumed that the concepts and relations in the rules were consistent with the BIM models, but this was not the case. To achieve full automated rule checking, both the regulations and the BIMs and Ontologies should use the same concepts and relations to define and describe the same objects, i.e., the two representations should speak "the same language" (Dimyadi et al., 2016; Zhou & El-Gohary, 2021).

Thus, the semantic alignment in ARC aims to map the extracted named entities in the text to the corresponding concepts in the knowledge base (e.g., ontology, knowledge graph) to facilitate the understanding of the text (Shen et al., 2014; Karadeniz et al., 2019). In the existing studies of ARC, the methods for aligning and mapping the concepts and relations in computer-interpretable regulations to those in ontologies or BIMs are mainly rule-based approaches, such as exhaustive usage of keyword matching (Zhang & El-Gohary, 2015; Zhou & El-Gohary, 2017) and handcrafted mapping tables (Xu & Cai, 2021).

However, semantic alignment methods have been widely researched (Shen et al., 2014) in other domains and could be divided into three types of approaches: 1) rule-based methods, 2) supervised learning methods, and 3) unsupervised learning methods. Rule-based methods utilize dictionary look-up and string-matching algorithms (Morgan et al., 2008) or hand-written rules to measure the morphological similarity between extracted words and ontology concepts (Ghiasvand & Kate, 2014). For string-matching algorithms, large efforts are involved in the construction of a large name dictionary containing aliases, abbreviations, alternate spellings, etc. (Shen et al., 2014). If an alias of a concept is not contained in the mapping table, then the semantic alignment process will fail. For hand-written rules, human efforts are involved to define text patterns and pattern-matching rules for semantic alignment. The supervised learning methods learn the similarities between extracted words and ontology concept names from labeled training data. In recent years, some open datasets for model training have been developed; therefore, deep learning-based semantic alignment methods have been investigated (Mohan et al., 2021). While this type of approach eliminates human involvement in pattern definition, it still requires manual effort to prepare a training dataset (Xu & Cai, 2021) because there are few open datasets that contain domain-specific entities in the construction domain. Unsupervised methods, such as word embedding models, can learn distributed representations of words from large unlabeled corpora and are promising approaches for capturing semantic information (Karadeniz & Özgür, 2019). Unsupervised methods require less human effort in the data preparation stage than the former two types of methods.

2.3 Research gaps

Despite a large number of efforts in automated regulatory document interpretation, there are still some limitations in the following two main aspects.

First, in the construction domain, the methods for semantic alignment are mainly based on rule-based methods in the existing studies, which are also hard-coded manners and hard to generalize. In addition, constructing mapping tables containing all aliases, abbreviations, and alternate spellings requires a large amount of effort.

Second, the existing studies mainly focus on interpreting the rules expressed by natural language into pseudocode or plain description logic. The pseudocode formats are still difficult to directly reason by computers (Xu & Cai, 2021). The studies utilizing plain description logic mainly focus on the logical representation of some simple rules. Some complex rules that are hampered by implicit required relationships and properties that are difficult to retrieve are ignored. The typical example is "the number of stories in a factory should not exceed 2". When the number of stories is not stored in the BIM model, the number of stories should be derived through aggregation functions via counting the floor objects. However, the counting ability of plain description logic language (e.g., the first order logic) is very limited (Kuske & Schweikardt, 2017). It is difficult to derive the implicit information required for rule checking.

There are two main obstacles when interpreting complex rules. On the one hand, the representability of the widely used plain description logical clauses is limited for the complex rules with implicit attributes that need additional reasoning. On the other hand, it is difficult to automatically choose the proper information transformation methods for different regulatory texts. For example, this can be seen when identifying the texts that need counting functions automatically. Text classification is a typical task for identifying the characteristics of a text. Zhou & El-Gohary (Zhang & El-Gohary, 2015; Zhou & El-Gohary, 2017) adopted a text classification method to filter out irrelevant text. However, the existing text classification studies mainly focus on the intended scopes and applications at a coarse granularity level (e.g., relevant or irrelevant), so they cannot be used to choose the proper information transformation method to be utilized for each rule.

3 Methodology and framework

To address the abovementioned problems, a novel knowledge-informed framework for ARC based on NLP techniques is proposed, as illustrated in Fig. 1. The proposed framework consists of four parts: ontology-based knowledge modeling, model preparation, rule interpretation, and checking execution. Based on the framework, three new methods, including the unsupervised-based semantic alignment method, the conflict resolution method, and the code generation method that supports a wider range of rule interpretations, are proposed. The main contributions of this work are highlighted in Fig. 1. First, we introduce the four parts of the proposed framework.

1. Ontology-based knowledge modeling. This part is the basis of the proposed framework, and the domain-specific concepts, properties, relations, and descriptions related to regulatory documents are integrated into the ontology. In addition, a set of model semantic enrichment rules in model preparation and conflict resolution rules in rule interpretation can be established based on domain-specific knowledge.

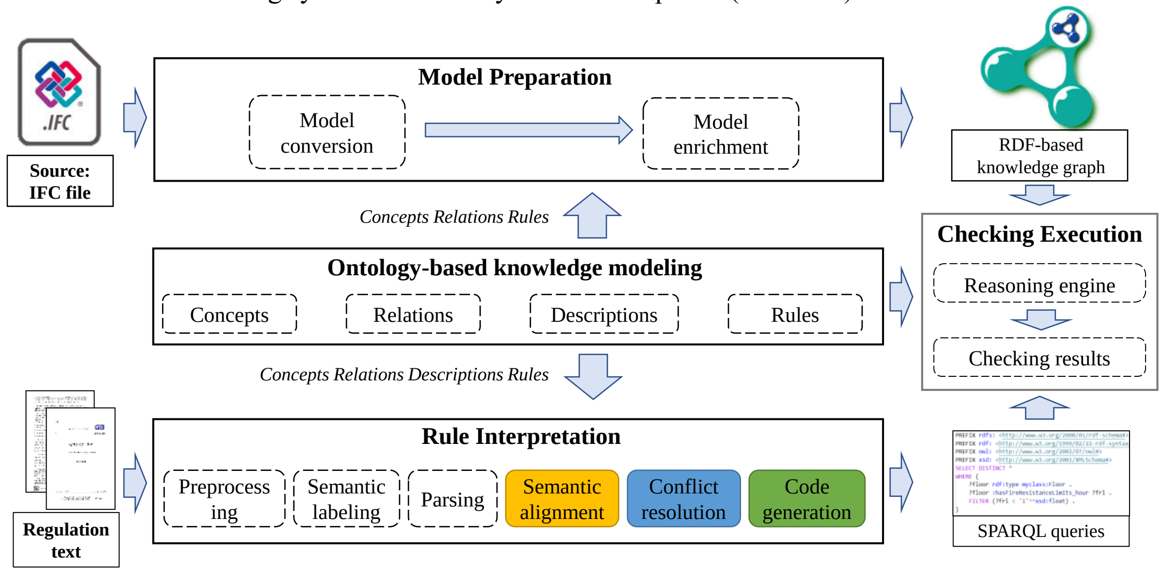
2. Model preparation. IFC (Industry Foundation Classes) is supported by a wide range of BIM software (IFCwiki, 2021), and thus, using IFC files for data exchange can improve the scalability of ARC systems. However, the IFC file cannot be directly parsed by the reasoning engine, thus requiring further model conversion and enrichment. This part aims to automatically convert the BIM model data stored in IFC format into the Turtle (Terse RDF Triple Language) format based on the proposed ontology,

the IFC parsing method, and the ontology mapping method. Moreover, a set of semantic model enrichment rules are adopted to infer the implicit information and supply missing information in the Turtle model before checking can begin. The model preparation pipeline consists of model conversion and model enrichment, as shown in Fig. 1. The details of the model preparation pipeline are illustrated in Appendix A.

3. Rule interpretation. In this part, the rules are automatically interpreted into a computer-executable format based on a pipeline of NLP-based methods, including preprocessing, semantic labeling, parsing, semantic alignment, conflict resolution, and code generation. First, preprocessing aims to preprocess these sentences by sentence splitting to facilitate later work. Semantic labeling aims to label words or phrases in a sentence. Subsequently, the parsing step parses the labeled sentence into a language-independent syntax tree. The semantic alignment automatically aligns the concepts in the syntax tree to those in the ontology based on unsupervised learning techniques, such as the word2vec technique. Then, a set of conflict resolution rules are utilized to revise the alignment result. Finally, the code generation step aims to transform the syntax tree after alignment into computer-executable code.

4. Checking execution. Finally, automated compliance checking is performed through a reasoning engine, such as protégé and GraphDB (Ontotext GraphDB, 2021).

To better illustrate the proposed knowledge-informed ARC framework, the regulatory texts from Section 3 in the *Chinese Code for fire protection design of buildings* (GB 50016-2014) are selected as an example. GB 50016-2014 (GB 50016-2014) is the Chinese national fire code that restricts the shape, properties, safe evacuation, and so on of the buildings. The regulatory texts of Section 3 in GB 50016-2014 (GB 50016-2014) are about the fire safety of factories. First, a new ontology and a set of rules are developed to integrate domain knowledge (Section 3.1). Then, the automated rule interpretation method is illustrated in Section 3.2. Note that the model preparation part is not the core of this work, so it will be briefly introduced in Appendix A. For the checking execution stage, this work implements an automated rule checking system based on Python and GraphDB (Section 5).



```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT DISTINCT *
WHERE {
    ?floor rdf:type myclass:Floor .
    ?floor :hasFireResistanceLimits_hour ?frl .
    FILTER (?frl < '1'^^xsd:float) .
}
```

Fig. 1 The proposed NLP-based and knowledge-informed automated rule checking framework

3.1 Ontology-based knowledge modeling

To integrate domain-specific knowledge and concepts, the fire protection of building ontology (FPBO) was developed. In addition, domain-range constraints and equivalent classes for conflict resolution and rules for semantic enrichment are also established based on the FPBO and domain knowledge.

3.1.1 Classes

For the building information domain, some concepts and hierarchical relations in ifcOWL (O'Sullivan et al., 2004) and Building Topology Ontology (Rasmussen et al., 2017) are referred to in the construction of the FPBO. For the fire protection domain, the concepts and relations are extracted and summarized from *GB 50016-2014* (GB 50016-2014). A semiautomated ontology construction method proposed by Qiu et al. (Qiu et al., 2018) was adopted. Based on the regulatory corpus, the concepts were first extracted using statistics-based methods (e.g., term frequency inverse document frequency), and the top 500 words with the highest score were extracted. Then, the semantic clustering method is used to merge the domain words. If the similarity (i.e., word2vec-based similarity) of two clusters is higher than a certain value (0.6 in this work), then merge the two clusters. A total of 183 clusters are obtained. Then, the subsumption method is used to build the taxonomy structure in each clustering. The basic idea is that concept c1 is considered more specific than concept c2 if the document set in which c1 occurs is the subset of the document set in which c2 occurs (Qiu et al., 2018). After the subsumption method, the typical taxonomy structure is that the first level includes "building" and the second level includes "plant", "warehouse", "residence", etc. At last, the concepts and hierarchy of the ontology were manually adjusted and supplemented. The semiautomated method reduces the manual effort in keywords searching and clustering. Finally, the FPBO was created in protégé (Musen et al., 2015). Fig. 2 illustrates part of the structure of the FPBO. The classes are represented by blue boxes in Fig. 2. The classes of *BuildingSpatialElement* and *BuildingComponentElement* are the central concepts of the FPBO. *BuildingSpatialElement,* representing the general space that has a 3D spatial extent related to the building domain, is further categorized into four subclasses: *BuildingSite*, *BuildingRegion*, *BuildingStorey*, and *BuildingSpace*. *BuildingComponentElement* represents the essential elements in architecture and structural engineering, such as beams, columns, walls, and doors.

3.1.2 Properties and their constraints

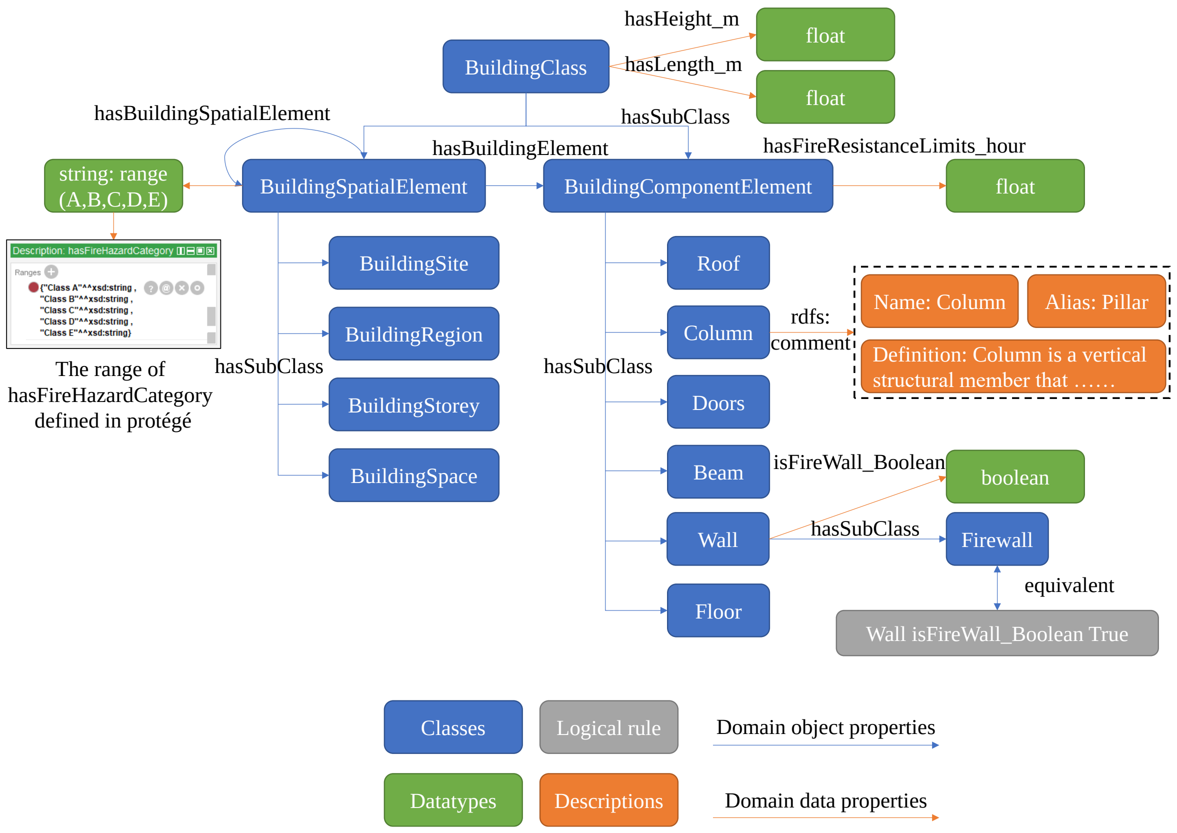
Object properties (e.g., *hasBuildingSpatialElement, hasBuildingElement)* are also introduced in the FPBO to establish the relationships between *BuildingSpatialElement* and *BuildingComponentElement*. The object properties are represented by blue arrows in Fig. 2. Key terms in the fire protection domain are defined as data properties, such as *hasFireResistanceLimit_hour* representing fire resistance rating. The data properties are represented by orange arrows in Fig. 2. In addition, the domains and ranges of the data properties and object properties have been defined, which could be used for conflict resolution when generating rules from texts. Part of the domains and ranges of the properties are shown in Fig. 2. For example, the domain of the data property '*isFireWall_Boolean*' is class '*Wall*', and its range is a bool value. Other situations are not allowed. For another example, the range of the data property '*hasFireHazardCategory*' only includes the following values: Class A, B, C, D, and E. These values also uniquely correspond to the data property '*hasFireHazardCategory*'. The ranges of the properties are defined using the protégé's function "data range expression", and the result of the data property '*hasFireHazardCategory*' is shown in Fig. 2. The range of the data property '*hasNumberOfFloors*' is also defined using the same method, and the range values are shown in Table 1.

3.1.3 Descriptions and equivalent classes

Different vocabularies in natural language may be used by various regulatory documents to describe the same concept. For instance, different terms, such as "fire partition wall" and "firewall" are often used to refer to the same concept '*firewall*'. Therefore, we define three types of descriptions, including names, aliases, and definitions for each concept via annotations. Each concept has one name, one definition which contains one or two short sentences, and zero to two aliases. A total of 287 descriptions are defined for 138 concepts. Since the aliases of the concepts are difficult to summarize completely, the definitions and semantic similarity matching methods are utilized in concept semantic alignment between rules and the FPBO. A typical example (e.g., the description of the column) of a description is shown in the orange box in Fig. 2. In addition, the equivalent classes are also considered in the FPBO for describing the same concept. The equivalent classes are defined by some logic rules, shown in the gray box in Fig.2. For example, the class: '*FireWall*' = class: '*Wall*' + data property: '*isFireWall_Boolean*' + value: '*True*'.

3.1.4 Semantic enrichment rules

Missing, incomplete, implicit, and incorrect information are major obstacles to automated code compliance checking in the construction industry (Bloch & Sacks, 2020). Thus, semantic enrichment should be performed to infer the implicit information and supply missing information before checking can begin. In this work, since most of the attributes have been supplemented via the ifc conversion process and the constructed ontology, only a few topological relations between the objects need to be inferred. Therefore, a series of semantic enrichment SWRL rules are defined based on the FPBO. The defined SWRL rules aim to derive implicit information regarding which building spaces and building components are contained in a building. The detailed SWRL rules are illustrated in Appendix A.

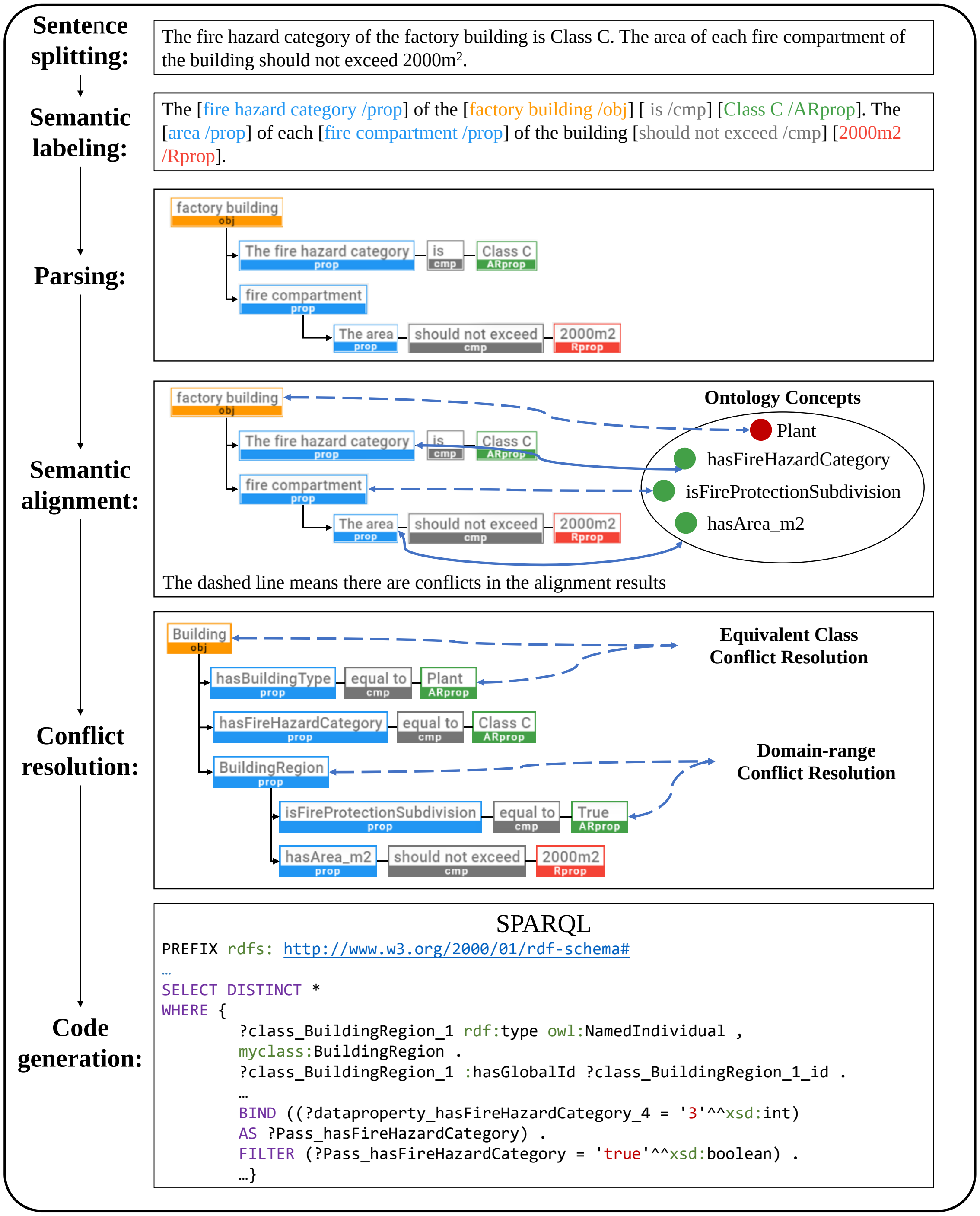


```
{"Class A"^^xsd:string ,
"Class B"^^xsd:string ,
"Class C"^^xsd:string ,
"Class D"^^xsd:string ,
"Class E"^^xsd:string}
```

Fig. 2 Structure of the fire protection of building ontology

3.2 Rule interpretation

This step aims to automatically interpret the domain regulatory texts expressed in natural language into computer-processable code based on a pipeline of NLP techniques, including preprocessing (see Section 3.2.1), semantic labeling and parsing (see Section 3.2.2), semantic alignment (see Section 3.2.3), conflict resolution (see Section 3.2.4), and code generation (see Section 3.2.5). Fig. 3 shows an example of the proposed automated rule interpretation method.



The fire hazard category of the factory building is Class C. The area of each fire compartment of the building should not exceed 2000m^2.

The [fire hazard category /prop] of the [factory building /obj] [is /cmp] [Class C /ARprop]. The [area /prop] of each [fire compartment /prop] of the building [should not exceed /cmp] [2000m2 /Rprop].

The dashed line means there are conflicts in the alignment results

```
PREFIX rdfs: http://www.w3.org/2000/01/rdf-schema#
…
SELECT DISTINCT *
WHERE {
        ?class_BuildingRegion_1 rdf:type owl:NamedIndividual ,
        myclass:BuildingRegion .
        ?class_BuildingRegion_1 :hasGlobalId ?class_BuildingRegion_1_id .
        …
        BIND ((?dataproperty_hasFireHazardCategory_4 = '3'^^xsd:int)
        AS ?Pass_hasFireHazardCategory) .
        FILTER (?Pass_hasFireHazardCategory = 'true'^^xsd:boolean) .
        …}
```

Fig. 3 Example illustrating the processing of automated rule interpretation

3.2.1 Preprocessing

Preprocessing aims to preprocess these sentences by sentence splitting to facilitate later work. Sentence splitting is a rule-based, deterministic consequence of tokenization (Jurasky et al., 2020). A sentence ends when sentence-ending punctuation (. , !, or ?) occurs. Thus, a regulatory document can be split into single sentences by identifying these sentence boundary indicators.

3.2.2 Semantic labeling and parsing

Sentence labeling and parsing aim to extract and parse the internal structure of a text represented in natural language and transform it into a syntax tree.

Semantic labeling is the process of assigning semantic labels to words or phrases in a sentence. Zhang et al. and Zhou et al. (Zhang et al., 2019; Zhou et al., 2022) pointed out that accurate and complete annotation and information extraction involve a deep level of semantic information analysis that requires a complete understanding of building code requirement sentences. Some requirement sentences are rich and complex in semantics and syntactic, which are challenging for conventional methods (e.g., POS tagging, gazetteer lookup). To better extract the features of the sentences, the authors (Zhou et al., 2022) defined seven semantic labels, including **obj**, **sobj**, **prop**, **cmp**, **Rprop**, **ARprop**, and **Robj**, based on the hierarchical structure of the objects and attributes in the BIM model. The **sobj** (short for super object), **obj** (short for object), and **prop** (short for property) labels are used to locate and identify checked elements, but the differences are their levels. The **sobj** is the parent node of **obj**, while the **prop** is the child node of **obj**. The **cmp** (short for compare) means the comparative/existential relation between a **prop** and a requirement. The **Rprop** is the restriction requirement applied to a **prop** that shall be satisfied. The **ARprop** is the applicability applied to a **prop**, and the rule checking will be performed if the applicability is satisfied. **Robj** is the parent or refereed element of a **Rprop** or **ARprop**. (Zhou et al., 2022)

Then, the authors used the state-of-the-art DNN pretrained model BERT (Devlin et al., 2018) for automated semantic annotation to better capture the deeper semantics of the sentences. The meanings of each semantic label and training process of the DNN model have been depicted in detail in previous work (Zhou et al., 2022). In this work, the predefined labels and the well-trained model are still utilized.

Parsing is the process of analyzing the structure of a labeled sentence and parsing it into the syntax tree. The domain-specific regulation text is more regular (e.g., less ambiguity and fewer homonym conflicts) than general nontechnical text; thus, it is more suitable for parsing with semantic labels (Zhang & El-Gohary, 2016). There are two commonly used grammars according to the Chomsky hierarchy: CFG (type-2) and regular grammar (type-3) (Chomsky, 1956). The authors (Zhou et al., 2022) adopted CFG because CFG has higher expressiveness and can be obtained by simpler regular expressions. The authors used the ANTLR4 package (Parr, 2013) and the Python language to implement the parsing process. The proposed method achieves state-of-the-art high accuracies for parsing simple and complex sentences. In this work, the parsing method is still utilized. Note that the syntax tree is still not in computer-executable format. Thus, further steps, including semantic alignment, conflict resolution, and code generation, are required to achieve the whole interpretation.

3.2.3 Semantic alignment

Semantic alignment aims to map the labeled elements in the text to the corresponding concepts in the knowledge base (e.g., ontology, knowledge graph) to facilitate the understanding of the text (Shen et al., 2014; Karadeniz et al., 2019). In this study, semantic alignment maps the extracted semantic elements in each node of the aforementioned syntax tree to the concepts in the FPBO, i.e., classes and data properties defined in Section 3.1. An example of semantic alignment is shown in Fig. 3, where the classes are represented in red circles in the ontology concepts and the data properties are in green circles. The dashed line means there are conflicts in the alignment results, and the downstream task conflict resolution should be performed on them.

The methods of semantic alignment include rule-based methods, supervised learning methods, and unsupervised learning methods. There are few open datasets that contain domain-specific entities in the construction domain. Therefore, large efforts are required to manually annotate a sufficiently large amount of training data for supervised learning methods. Analogous to supervised learning methods, rule-based methods also require large efforts in the construction of a large and comprehensive name dictionary containing aliases, abbreviations, alternate spellings, etc. (Shen et al., 2014). Given that huge efforts are needed to build datasets and dictionaries for the construction domain, this study adopts an unsupervised learning-based method, which only needs less effort than the other types of methods.

The unsupervised learning-based approach adopted in this work is mainly based on the assumption that semantically similar words have similar vector spaces. The workflow of the semantic similarity measurement between a named entity and a concept is shown in Fig. 4. For each named entity in the syntax tree, the semantic similarity between the named entity and all the ontology concepts is measured, and then the ontology concept with the highest similarity score is assigned as the normalized concept to the named entity. As each concept has several descriptions, the similarity between the named entity and each description of the concept is calculated first, and then the average similarity value of all descriptions is taken as the semantic similarity between the concept and the target named entity. To measure the semantic similarity between one description and the target named entity, the multiword named entity and description are preprocessed, including tokenizing, word splitting, and stopping word removal, before two composing wordlists are generated. Each word in the wordlist is represented as a real-valued vector using a pretrained unsupervised learning model illustrated later. For the multiword named entity and description, the vector representations are computed by weighted averaging the vectors of their composing words. Widely used weighted methods include the average weighted method and the TD-IDF (Term Frequency-Inverse Document Frequency) weighted method (Lilleberg, et al., 2015). Finally, a cosine similarity score is assigned as the semantic similarity between the entity and the description.

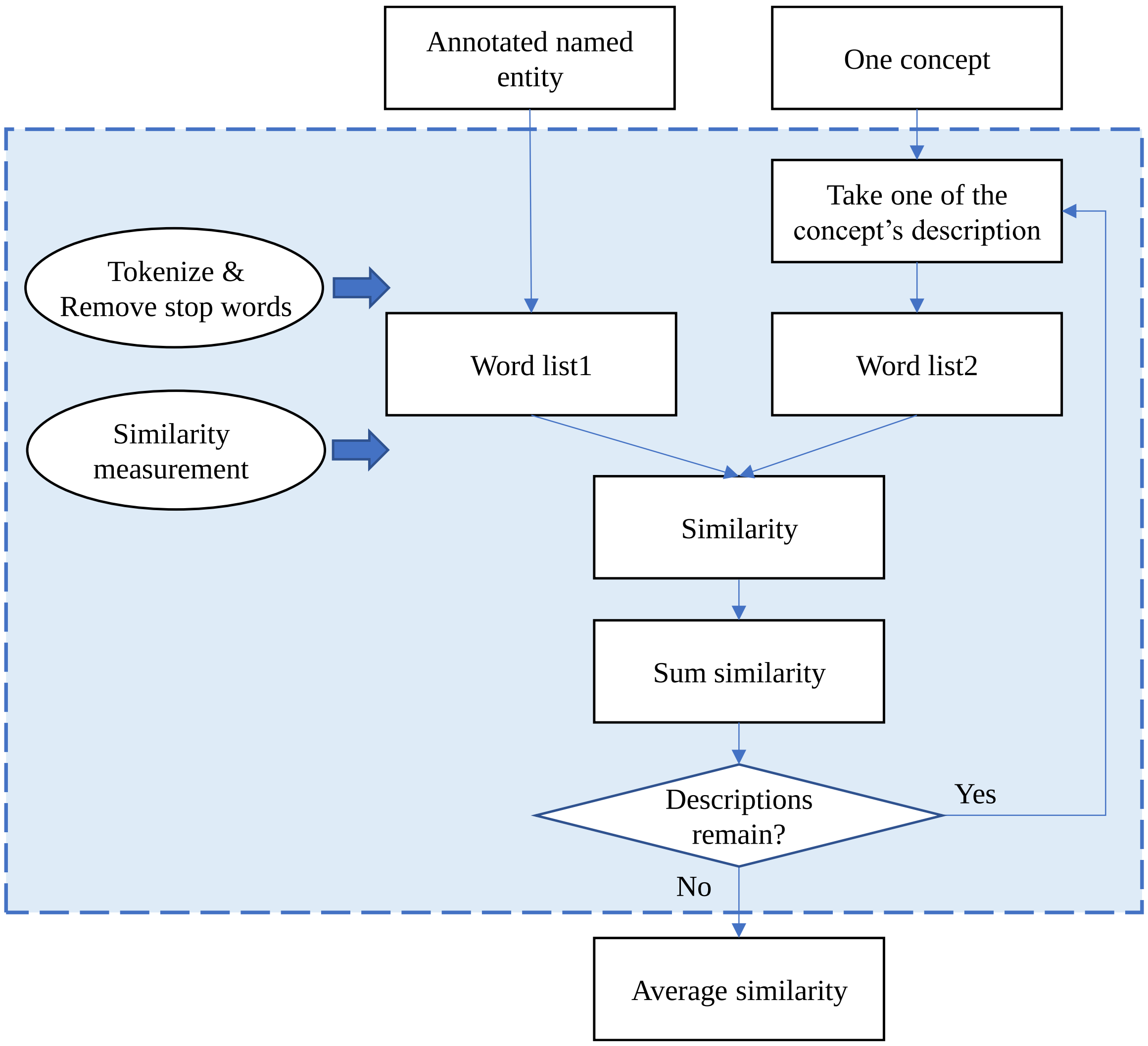


Fig. 4 Flow chart of semantic similarity measurement between a named entity and a concept

In this work, the adopted unsupervised learning model is the word embedding model (Mikolov et al., 2013). The open-source toolkit gensim (Rehurek et al., 2010) is used to train the word embedding model with dimensions of 400 according to previous research (Godin et al., 2015; Savigny & Purwarianti, 2017). The training corpus is composed of the Chinese corpus from Wikimedia (Wikimedia, 2021) and 1560 Chinese code corpora from the website of the Chinese codes (Soujianzhu, 2021). On the one hand, the corpus of the domain-related code corpus is too small to cover some frequently used words. On the other hand, the Chinese corpus from Wikimedia has few domain-related words. The performance of the models trained using two corpora is better than only using one (Zheng et al., 2022). Therefore, the model has been trained using the two corpora above.

In addition, we also adopt two dictionary-based keyword matching methods as the baseline, including the keyword matching method (KW) and the weighted keyword matching method (KW-Weighted). For KW, if a named entity completely matches one of the descriptions of an ontology concept, it is assigned as the normalized concept. For KW-Weighted, in the descriptions of an ontology concept, the shorter the length of the description, the more information it contains, and the higher the correlation with the term. Based on the above ideas, Formula (1) can measure the similarity between a named entity and a concept as follows:

$$similarity = \frac{1}{n} * (\sum_{n=i}^{N} \frac{similarity_i}{log\ (len(term_{description_i}))}) \quad (1)$$

where *similarity*$_i$ indicates the similarity between the named entity and the number *i* description of a concept. If the named entity is the same as the number *i* description, the value of *similarity*$_i$ equals 1; if the named entity is the substring of the number *i* description, the value of *similarity*$_i$ equals 0.6, and the rest equals 0. The denominator is the log value of the string length of description *i*. The similarity between the named entity and the concept is the average of the similarity values of the concept's descriptions.

3.2.4 Conflict resolution

Conflict resolution refers to modifying the semantic alignment results according to the domain knowledge of civil engineering. It is necessary to ensure that the regulations and the models use the same concepts and relations to define and describe the same objects. As a result, two types of conflict resolution methods, the domain-range conflict resolution method and the equivalent class conflict resolution method, are proposed based on domain knowledge.

For the domain-range conflict resolution method, the domains and ranges of the data properties and object properties are defined in Section 3.1. For example, the range of a concept whose type is a data property cannot be an object. Otherwise, the wrong situation where an attribute value has an object may occur, which violates the definition of the BIM model. Another situation is when the range of some data properties must only be certain values. For example, the range of the data property '*hasFireHazardCategory*' only includes the following values: Class A, B, C, D, and E. These values also uniquely correspond to the data property '*hasFireHazardCategory*'. This means that we can identify the data property according to its range if the data property is missing in the text. The results of semantic alignment can be identified and checked according to ranges of data properties and object properties, and are further modified via the domain-range conflict resolution dictionary (Table 1). The domain-range conflict resolution dictionary was extracted from FPBO for a better index. If the wrong semantic alignment results are identified according to the range of a concept, then the information of correct concepts can be determined by keyword matching with the wrong concept as the key; afterward, the syntax tree structure can be modified. The specific case is shown in Fig. 5(a). The root entity '*fire compartment*' is aligned to '*isFireProtectionSubdivision_Boolean*', which is a data property. The range of the data property cannot have an object '*Column*'. Therefore, the original semantic alignment result is wrong. Then, the syntax tree is modified according to Table 1. If the text that relates to the data property is missing in the original text, we can identify it and complement the missing value according to the range of the data property. The specific case is shown in Fig. 5(b). The missing data property '*hasFireHazardCategory*' is supplemented via the range value '*Class A*'.

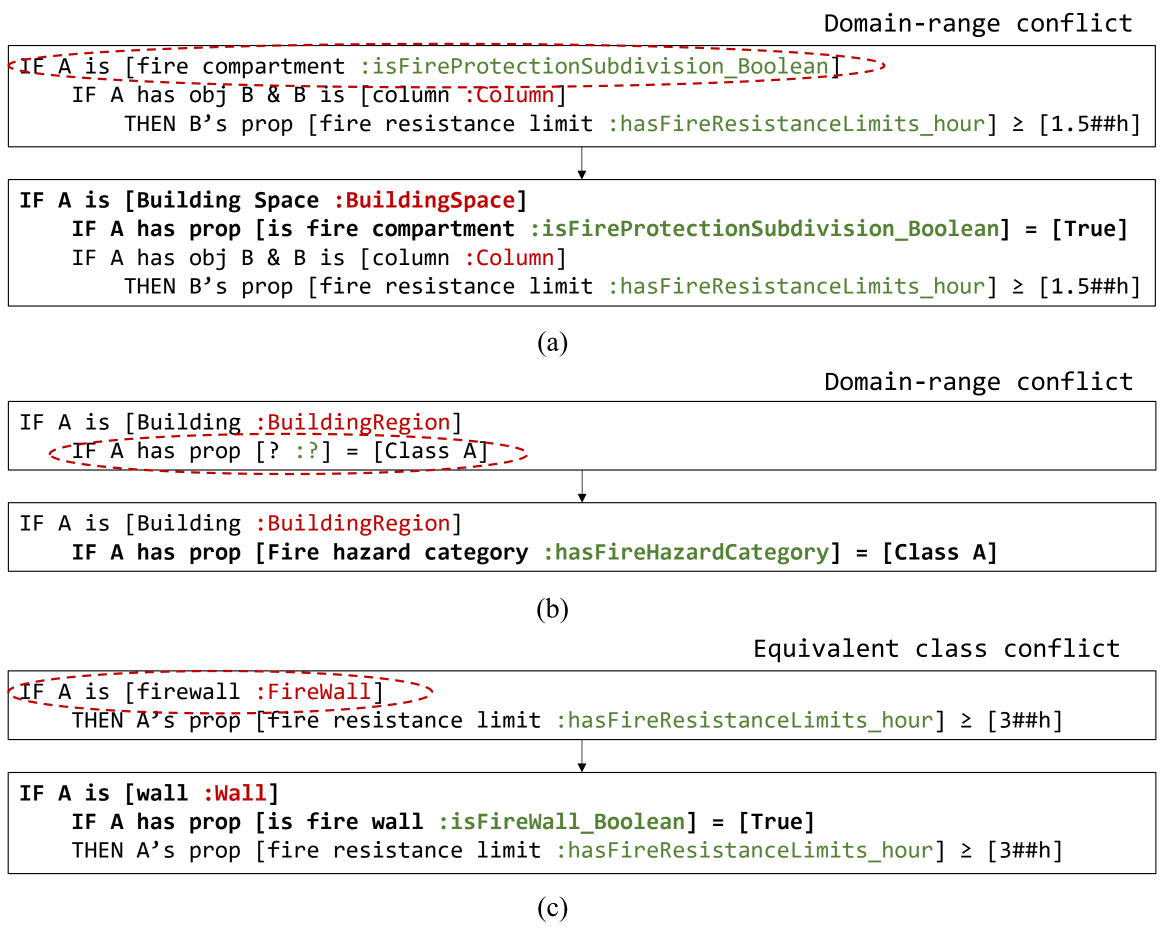
For the equivalent class conflict resolution method, equivalent classes are also defined. Since the IFC model contains the most basic concepts, we propose an equivalent concept conflict resolution method based on the decomposition of concepts using equivalent classes. Based on domain knowledge, some complex concepts in the regulations can be defined as the sum of basic concepts. For example: class: '*FireWall'* = class: '*Wall'* + data property: '*isFireWall_Boolean'* + value: '*True*'. To identify this kind of conflict, an equivalent concept conflict resolution dictionary is established according to the FPBO equivalent classes, as shown in Table 2. If the complex concepts are identified according to the key in Table 2, then the information of a series of simple concepts can be determined by keyword matching with the key; afterward, the syntax tree structure can also be modified. The specific case is shown in Fig. 5(c). The complex class '*FireWall'* is decomposed into the simple class '*Wall'* and the data property '*isFireWall_Boolean'*.

Table 1 domain-range conflict resolution dictionary

Key (Original concept of the node)	Range of the concept	Value ([New concept of the node, new word of the node, concept of its sub-node, word of its sub-node, the value of its sub node])
isFireProtectionSubdivision_Boolean	True/False	[BuildingSpace, Building Space, isFireProtectionSubdivision_Boolean, is fire compartment, True]
IsSecurityExits_Boolean	True/False	[Doors, door, IsSecurityExits_Boolean, is Safe exit, True]
isFireWall_Boolean	True/False	[Wall, wall, isFireWall_Boolean, is fire wall, True]
hasFireHazardCategory	Class A, Class B, Class C, Class D, Class E	/
hasNumberOfFloors	Single-story, multiple-story	/

Table 2 equivalent concept conflict resolution dictionary

Key (Original concept of the node)	Value ([New concept of the node, new word of the node, concept of its sub-node, word of its sub-node, the value of its sub node])
firewall	[Wall, wall, isFireWall_Boolean, is fire wall, True]
Plant	[BuildingRegion, Building, hasBuildingType, has the building type of, Plant]



Domain-range conflict

```
IF A is [fire compartment :isFireProtectionSubdivision_Boolean]
    IF A has obj B & B is [column :Column]
        THEN B’s prop [fire resistance limit :hasFireResistanceLimits_hour] ≥ [1.5##h]
```

```
IF A is [Building Space :BuildingSpace]
    IF A has prop [is fire compartment :isFireProtectionSubdivision_Boolean] = [True]
    IF A has obj B & B is [column :Column]
        THEN B’s prop [fire resistance limit :hasFireResistanceLimits_hour] ≥ [1.5##h]
```

(a)

Domain-range conflict

```
IF A is [Building :BuildingRegion]
    IF A has prop [? :?] = [Class A]
```

```
IF A is [Building :BuildingRegion]
    IF A has prop [Fire hazard category :hasFireHazardCategory] = [Class A]
```

(b)

Equivalent class conflict

```
IF A is [firewall :FireWall]
    THEN A’s prop [fire resistance limit :hasFireResistanceLimits_hour] ≥ [3##h]
```

```
IF A is [wall :Wall]
    IF A has prop [is fire wall :isFireWall_Boolean] = [True]
    THEN A’s prop [fire resistance limit :hasFireResistanceLimits_hour] ≥ [3##h]
```

(c)

Fig. 5 Example depicting the processing of conflict resolution (in the square brackets, the plain words are in black font, the classes in FPBO are in red font, and the data properties in FPBO are in green font). The red dashed circles show the conflicts in the original sentences. The conflict resolution results are bolded.)

3.2.5 Code generation

After semantic alignment and conflict resolution, the code generation step is conducted to transform the syntax tree into a computer-processable format. To support a wider application range for checking rules covering implicit information, a new code generation method consisting of text classification is proposed. The text classification step aims to identify the proper SPARQL function for each rule. The SPARQL (Prud'hommeaux et al., 2008; Harris et al., 2013) language is selected for its ability to reason implicit information.

Text classification aims to classify sentences into different categories based on semantics and is an important preparation for text analysis (Solihin et al., 2015; Salama et al., 2016). Salama (Salama et al., 2016) classifies regulation documents into 14 predefined categories, including security, safety, health, etc., depending on the objective of the document. Solihin et al. (Solihin et al., 2015) classified the regulatory text into four categories according to the complexity and suitability of the tools or techniques required, as shown in Table 3. Following the above two classification works, this study proposes new classification categories that can consider the computability of the text. The categories and their definitions are shown in Table 3. Different categories of rules are subsequently processed using different code generation techniques. For example, the rules of Class 2 may require additional reasoning functions, such as the aggregated algebra provided by SPARQL.

To date, several methods have been proposed for automated text classification. For example, Caldas et al. (Caldas et al., 2002) developed an automated text classification system based on traditional machine learning methods, such as the Rocchio algorithm, naïve Bayes, k-nearest neighbors, and SVMs. Salama (Salama et al., 2016) classified documents based on machine learning methods, such as naïve Bayes, maximum entropy, and SVMs. This study adopts a classification method based on keyword matching and semantic ranking, as text classification techniques are not the focus of this paper and machine learning methods depend on large datasets that require very expensive manual labeling. In this work, we first define a table of keywords for classification. Part of the keywords is shown in Table 4. A regulatory text is classified into a specific category if it contains a keyword of the category. A text may contain more than one keyword, resulting in overlapping classifications. Therefore, a classification priority for semantic ranking is defined as follows: Class 3> Class 2.2 > Class 2.1 > Class 1 > Class 4. The results of the automated text classification are shown in Section 4.

Table 3 Rule categories

Solihin et al., 2015	Ours	
Class 1: Rules that require a single or small number of explicit data	Class 1: Direct attribute related requirement (Description: Rules that require only explicit data in BIM model)	
Class 2: Rules that require simple derived attribute values	Class 2: Indirect attribute related requirement (Description: Rules that require simple derived values from explicit data in BIM model)	Class 2.1: Quantity attribute related requirement
		Class 2.2 Other complex indirect attribute related requirement
Class 3: Rules that require an extended data structure	Class 3: Spatial geometry related requirement (Description: Rules that require extended data structure and complex derivation, such as geometric computing)	
Class 4: Rules that require a “proof of solution”	Class 4: Others (Description: Rules which is hard to be automated interpreted)	

Table 4 Keywords for rule categories

Class	Keywords (in Chinese)
Class 1: Direct attribute related requirement	Length, width, height, thickness, depth, span, precision…
Class 2.1: Quantity attribute related requirement	Number, times…
Class 2.2 Other complex indirect attribute related requirement	Area, volume…
Class 3: Spatial geometry related requirement	Distance…
Class 4: Others	

After the proper SPARQL functions for the rules are identified by the text classification method, the syntax trees are transformed into SPARQL by pattern rules. According to the SPARQL syntax (Prud'hommeaux et al., 2008; Harris et al., 2013), During the transformation, the following three aspects should be noted, as shown in Fig. 6.

1. Prefix. The prefix is the SPARQL instruction for a declaration of a namespace prefix which defines where the adopted function comes from. The prefix should be written at the beginning of a SPARQL file. It allows us to write prefixed names in queries instead of having to use full URIs everywhere. Therefore, it is a syntax convenience mechanism for shorter, easier-to-read (and write) queries.

2. Aggregate Algebra. Aggregate algebra is a set of predefined function provided by SPARQL, which supports counting and other aggregate functions. According to the text classification, except for the rules of Class 1 direct attribute-related requirements, the data required for checking other categories of rules cannot be directly obtained from the Turtle file, and additional derivations are needed. In this work, for the rules of Class 2 indirect attribute-related requirements, the aggregate algebra provided by SPARQL can be used for some implicit data via simple derivation from explicit data in the BIM model. For example, the explicit data in the BIM model are that fire compartment A has safety exits A, B, and C. The indirect attribute related requirement: "The number of safety exits for each fire compartment is not less than 2" can be checked using the COUNT provided by SPARQL. The code "`(COUNT(DISTINCT? safety_exit) AS? Safety_exit_num)`" can derive and store the number of safety exits, which is implicit data, in '*?Safety_exit_num*', to realize automated rule checking.

3. Object property mapping. Object property mapping is to establish the relationship between the two objects queried. For example, space A contains element B. The 'contains' is expressed through the object property 'hasBuildingElement'. If there is no object property, the relationship between space A and element B can not be reasoned, resulting in the query failing. The object property between the two classes can be uniquely determined by a HashMap with two keys, where one key is the class name of the domain class and the other key is that of the range class.

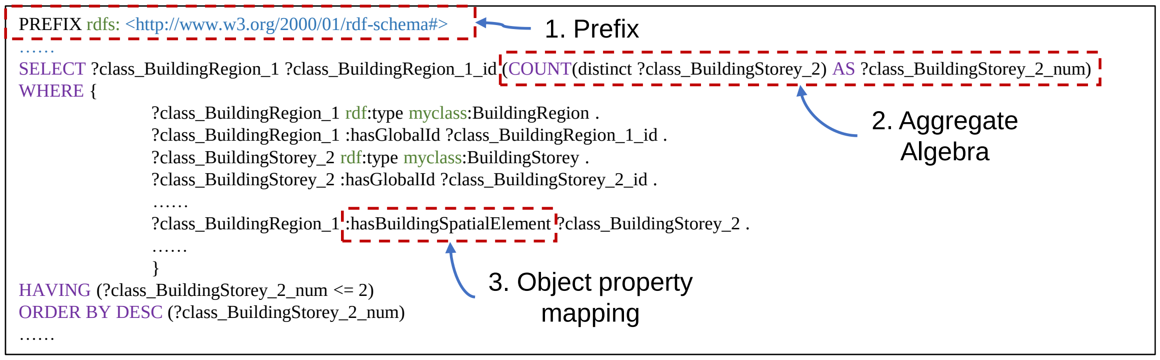

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
……
SELECT ?class_BuildingRegion_1 ?class_BuildingRegion_1_id (COUNT(distinct ?class_BuildingStorey_2) AS ?class_BuildingStorey_2_num)
WHERE {
                ?class_BuildingRegion_1 rdf:type myclass:BuildingRegion .
                ?class_BuildingRegion_1 :hasGlobalId ?class_BuildingRegion_1_id .
                ?class_BuildingStorey_2 rdf:type myclass:BuildingStorey .
                ?class_BuildingStorey_2 :hasGlobalId ?class_BuildingStorey_2_id .
                ……
                ?class_BuildingRegion_1 :hasBuildingSpatialElement ?class_BuildingStorey_2 .
                ……
                }
HAVING (?class_BuildingStorey_2_num <= 2)
ORDER BY DESC (?class_BuildingStorey_2_num)
……
```

Fig. 6 SPARQL syntax to be noted during the transformation process

4 Experiments and Results

This section describes the experiment conducted to validate the proposed method. First, several compulsory regulation texts are selected from *GB 50016-2014* (GB 50016-2014). The gold standards of concept alignment and text classification are manually developed. Second, various automated concept alignment methods are compared and evaluated based on the gold standard. Third, the results of automated text classification are compared with the gold standard. Finally, the time consumed by the proposed automated rule interpretation method is compared with the time consumed by the experts' manual rule interpretation method.

4.1 Dataset development

The regulatory texts from Section 3 in the *Chinese Code for fire protection design of buildings* (GB 50016-2014) are selected as the data source. First, the rules represented in figures are transformed into natural language. Then, the regulatory text on building codes is split into single sentences by semicolons, periods, and newlines. Next, these sentences are classified by experts according to the rule categories in Table 3 to form the gold standard for the automated text classification method. Then, the semantic alignments of the selected sentences are manually developed according to the FPBO to form the gold standard for the automated semantic alignment method. Finally, 27 types of semantic alignment tags are established, and the dataset contains 99 sentences and 468 semantic alignment labels. A repository containing the dataset is established on GitHub and can be found at https://github.com/SkydustZ/auto-rule-transform.

4.2 Semantic alignment result

To evaluate the performance of automated semantic alignment methods, the model predictions are compared with those from the gold standard. Accuracy and running time are employed as evaluation indices, and the results are shown in Table 5. Table 5 shows that the baseline algorithm KW achieves the lowest accuracy at 55.8%, and the KW-Weighted algorithm achieves an accuracy of 65.2%, which is just higher than the KW. The accuracy of methods based on semantic similarity (e.g., W2V-avg, W-tfidf, and WMD) is higher. The proposed semantic similarity and conflict resolution algorithm achieve the best accuracy of 90.1%. The KW-based algorithms only consider the morphological similarity between the named entities and the FPBO concepts, but they do not consider the semantic information of concepts represented by concept descriptions. In addition, different vocabularies in natural language may be used by various regulatory documents to describe the same concept. For instance, different terms, such as "fire partition wall" and "firewall" are often used to refer to the same concept of a '*firewall*'. Once the synonyms and aliases of a concept are not defined in the ontology, the KW-based methods cannot align the concept well. However, semantic similarity-based methods can measure the semantic similarity of two concepts, even if they differ greatly in morphology. The analysis of the misclassifications by semantic similarity-based methods suggests two main reasons for this. First, the named entity composed of more than one concept is difficult to distinguish. For example, the named entity '*fire protection zone*' is mistakenly aligned to the data property '*isFireProtectionSubdivision_Boolean*' in the FPBO. However, it should be aligned to the class '*BuildingSpace*' with the data property '*isFireProtectionSubdivision_Boolean*', and the value of the data property should be '*True*'. Second, it's hard to recognize the abbreviations or implicit concepts in the regulatory text. For example, “two-story house” is the abbreviation of “the number of floors in the house is 2”. The “two” in the phrase “two-story” should be the value of the data property ‘*hasNumberOfFloors*’. The semantic similarity-based methods cannot align the named entity “two-story” with the data property ‘*hasNumberOfFloors*’ with a value of 2. Another example is that "Plant in Class A" is the abbreviation of "the plant with the fire hazard category in Class A". "Class A" is the value of the data property ‘*hasFireHazardCategory*’. The semantic similarity-based methods cannot align "Class A" with the data property ‘*hasFireHazardCategory*’ with a value of ‘*Class A*’. Both the first and second types of misclassifications can be identified and corrected based on the conflict resolution methods proposed in Section 3.2.4. The proposed semantic similarity and conflict resolution method not only considers the semantics of a single word but also considers the information of the whole sentence structure and domain knowledge. Therefore, misclassifications are found and corrected according to the established conflict resolution dictionary, achieving an accuracy of 90.1%. There are two reasons why the accuracy cannot reach 100%. 1) Syntax tree error. When the sentence is too complicated, the syntax tree will fail to be constructed. Therefore, the sentence structure information cannot be considered. In the validation dataset, 3 complicated sentences fail to be constructed, and the other 96 sentences are correctly constructed. Therefore, 30% of alignment errors are caused by this type. 2) Conflict resolution error. When the semantic alignment results are far from the ground truth, they cannot be corrected by the proposed conflict resolution method. Because the defined correction dictionary does not contain such errors. For example, when the 'bearing wall' is wrongly predicted as 'RoofSheathing' in the process of semantic alignment, then the conflict resolution dictionary can not identify the error. 70% of alignment errors are caused by this type.

Because the structures of syntax trees are considered in the proposed semantic similarity and conflict resolution method, the time consumed by syntax tree construction is added to the time of the proposed method. Although the computational complexity of the proposed method is higher than that of other algorithms, the running time on the validation datasets including 27 concepts and 99 sentences is still acceptable. The average time consumed per sentence does not exceed 1 second.

Table 5 Performance of different semantic alignment methods

Method	Accuracy	Time
KW	55.8%	6.83 s
KW-Weighted	65.2%	**5.10 s**
W2V-avg	77.6%	25.8 s
W2V-tfidf	77.0%	33.9 s
WMD	73.0%	44.9 s
W2V-avg+conflict resolution	**90.1%**	75.0 s

Table 6 shows the interpretation performance at the sentence level. Our method achieves a 60.6% accuracy for semantically aligning the elements in whole sentences, which is higher than the results of other methods. Sentence-level accuracy is more suitable to indicate the performance of a semantic alignment method because this stricter criterion also considers the applicability of a method, such as supporting the downstream tasks, such as code generation and rule execution (Zhou et al., 2022). In the entire ARC process, each result of rule interpretation is executed for rule checking; therefore, the total correctness is related to the sentence-level accuracy of rule interpretation. The result of high sentence-level aligning accuracy also demonstrates the high interpretation performance of our method.

Table 6. Automated interpretation performance of the proposed method at the sentence level.

Method	Sentence number	Correctly interpreted sentences	Accuracy
KW-Weighted	99	14	14.1%
W2V-avg	99	20	20.2%
W2V-avg + conflict resolution	99	60	60.6%

4.3 Text classification result

To measure the result, the model predictions are compared with the gold standard, and the precision (P), recall (R), and F1-score (F1) are calculated for each semantic label as follows:

$$P = N_{correct}/N_{labeled} \tag{2}$$

$$R = N_{correct}/N_{true} \tag{3}$$

$$F_1 = 2PR/(P + R) \tag{4}$$

where $N_{\{correct,labeled,true\}}$ denotes the number of {model correctly labeled, model labeled, true} elements for a label. Finally, the weighted (i.e., micro) average F1-score is calculated to represent the overall performance (n_i denotes the number of elements of the i-th semantic label) as follows:

$$\text{Weighted } F_1 = \left(\sum_i n_i F_{1,i}\right) / \sum_i n_i \tag{5}$$

Table 7 shows that the adopted classification method based on keyword matching and semantic ranking obtains a promising 83.4% overall F1-score on the validation dataset. As a proof of concept in an early stage, the adopted sentence classification method shows promising results because it can classify different categories for long and complex sentences with relatively high accuracy, which reduces the human effort and supports the information transformation stage. However, it was found that the F1-score of some categories, such as Class 2.1 and Class 4, is low, which is due to the imperfect classification priority for semantic ranking or the incomplete keywords for rule categories. In future research, we will develop a larger corpus and train deep learning models that can better understand the meaning of the text to improve the effect of automated text classification.

Table 7 Performance of text classification method

Tag	Number	Precision	Recall	F1-score
Class 1	43	93.5%	100%	96.6%
Class 2.1	2	12.5%	100%	22.2%
Class 2.2	36	96.4%	75%	84.4%
Class 3	10	100%	100%	100%
Class 4	10	100%	10.0%	18.2%

Total	101	94.2%	82.2%	83.4%

4.4 Performance of automated rule interpretation

Table 8 shows the rule categories that can be automatically interpreted by our methods and previous methods. The SPARQL language provides predefined counting functions which is more suitable and concise for rules with implicit attributes that need additional reasoning than the widely used plain description. Therefore, our method can interpret the rules of Classes 2.1 & 2.2, which were difficult in previous studies.

Table 8 Rule categories can be interpreted

Rule Category	Previous studies	Our methods
Class 1	√	√
Class 2.1 & 2.2	×	√

To prove the effectiveness of the proposed automated rule interpretation method, the time consumed by the proposed automated rule interpretation method is compared with the time consumed by the experts' manual rule interpretation method. Two experts who participated in the experiment both major in both civil engineering and computers and are skilled in writing SPARQL. It should be noted that the keyword mapping method or handcrafted mapping tables are used in previous works, resulting in low sentence-level alignment accuracy and difficulty in generating SPARQL codes, as illustrated in Table 6. Therefore, the performance of our method is only compared with the manual results.

In this experiment, 4 direct attribute-related requirements (Class 1) and 8 indirect attribute-related requirements (Classes 2.1 & 2.2) are selected. The time consumed to interpret rules from natural language into SPARQL, which can be executed by the reasoning engine correctly, is recorded. It should be noted that the automated interpretation method cannot guarantee that the interpretation result is 100% correct, so manual correction is required. Therefore, the time consumed by the manual correction is also recorded as a part of the automated interpretation time. The results in Table 9 show that the time consumed by automated rule interpretation is 20.7% of that of the manual interpretation for direct attribute-related requirements and 17.2% for indirect attribute-related requirements, even if the manual correction time is considered. This means that the proposed rule interpretation method greatly reduces the time and cost.

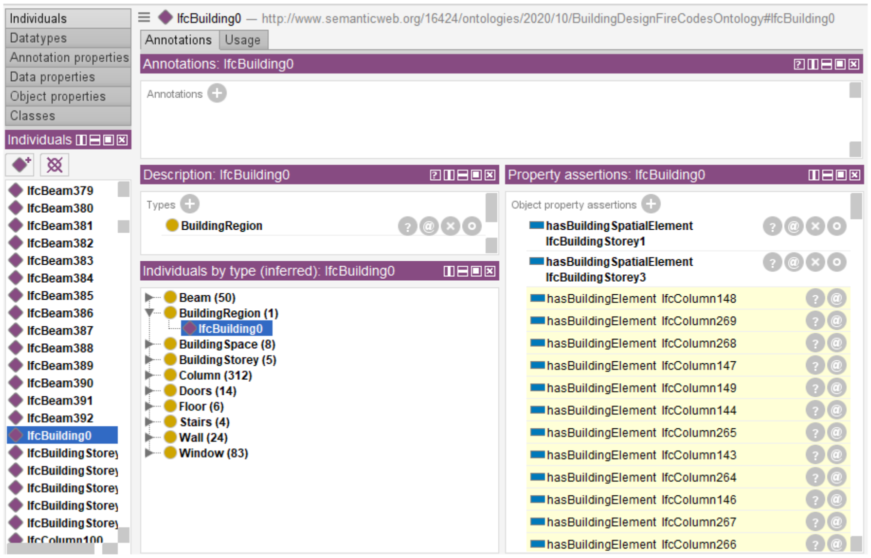
Table 9 Time consumed by the automated rule interpretation method

Method	Rule Class	Parsing (s)	Semantic labeling & Postprocessing (s)	Manual revised (s)	Total time (s)
Expert	Class 1	99	1288	/	1387
	Class 2.1 2.2	146	2344	/	2490
Algorithm	Class 1	0.43	65.39	190.75	286.57
	Class 2.1 2.2	0.56	128.64	298.69	427.89

Note: Manual correction refers to the time required to modify the automatically generated SPARQL code so that it can be executed correctly. Not all generated SPARQL codes need to be corrected manually.

5 Application

To demonstrate the real application of our method, this section implements rule checking of an actual plant building. Fig. 2 shows the workflow of our automated rule checking system. After the model is established, according to the model preparation method proposed in Appendix A, the IFC file exported from the Revit BIM model is converted into Turtle format. IFC objects related to fire protection that design compliance checking are converted into 507 instances of the FPBO. Then, semantic enrichment is conducted using the SWRL rules defined in Appendix A and the Pellet reasoner. The original file has 3817 axioms, while the inferred file has 10824 axioms, which means that some implicit information is inferred. The logical consistency and data consistency of the axioms are verified by reasons and experts, respectively. For example, the implicit information of the instance '*IfcBuilding0*' has been reasoned out, as shown in the content with a yellow background in Fig. 7.



IfcBuilding0 — http://www.semanticweb.org/16424/ontologies/2020/10/BuildingDesignFireCodesOntology#IfcBuilding0

Annotations: IfcBuilding0

Individuals: IfcBeam379, IfcBeam380, IfcBeam381, IfcBeam382, IfcBeam383, IfcBeam384, IfcBeam385, IfcBeam386, IfcBeam387, IfcBeam388, IfcBeam389, IfcBeam390, IfcBeam391, IfcBeam392, IfcBuilding0, IfcBuildingStorey, IfcBuildingStorey, IfcBuildingStorey, IfcBuildingStorey, IfcBuildingStorey, IfcColumn100

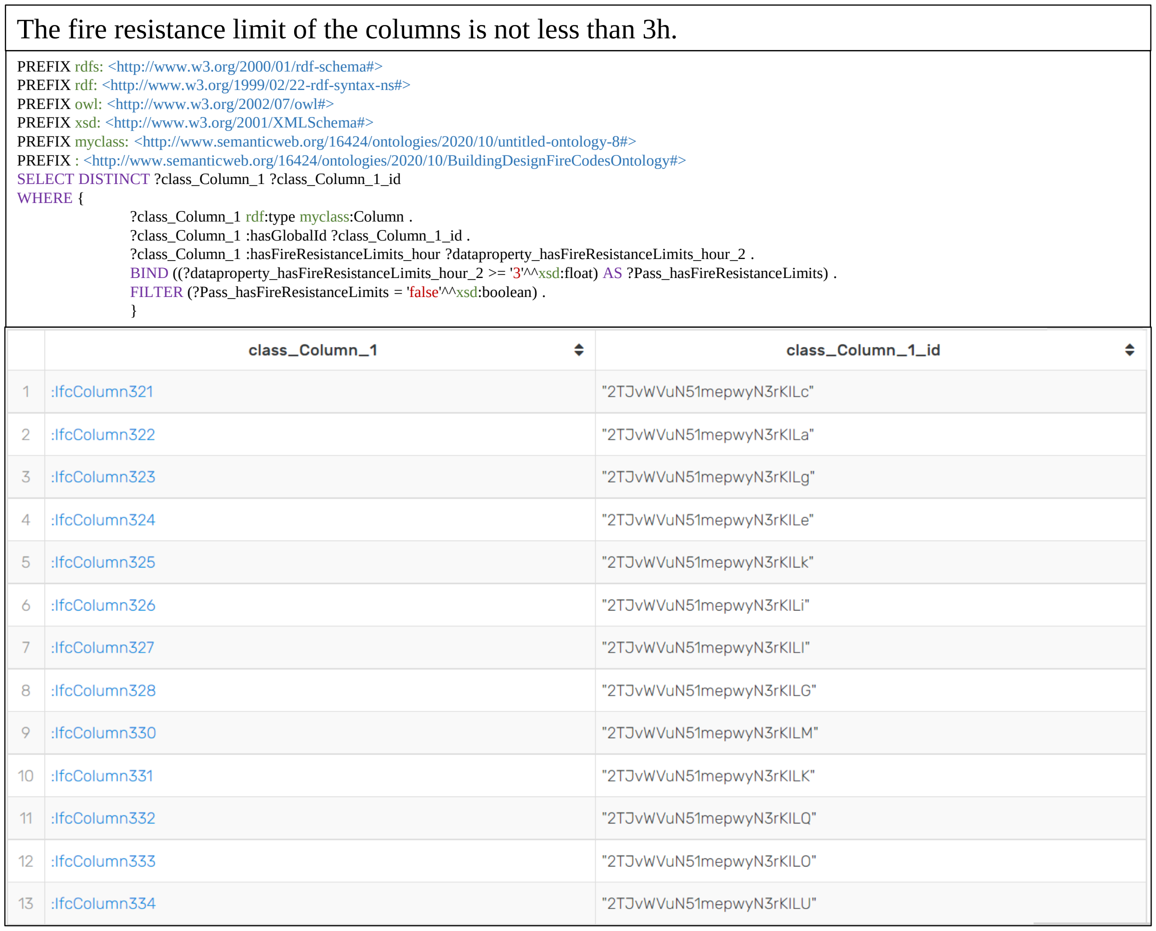
Description: IfcBuilding0
Types: BuildingRegion

Individuals by type (inferred): IfcBuilding0
- Beam (50)
- BuildingRegion (1)
  - IfcBuilding0
- BuildingSpace (8)
- BuildingStorey (5)
- Column (312)
- Doors (14)
- Floor (6)
- Stairs (4)
- Wall (24)
- Window (83)

Property assertions: IfcBuilding0
- hasBuildingSpatialElement IfcBuildingStorey1
- hasBuildingSpatialElement IfcBuildingStorey3
- hasBuildingElement IfcColumn148
- hasBuildingElement IfcColumn269
- hasBuildingElement IfcColumn268
- hasBuildingElement IfcColumn147
- hasBuildingElement IfcColumn149
- hasBuildingElement IfcColumn144
- hasBuildingElement IfcColumn265
- hasBuildingElement IfcColumn143
- hasBuildingElement IfcColumn264
- hasBuildingElement IfcColumn146
- hasBuildingElement IfcColumn267
- hasBuildingElement IfcColumn266

Fig. 7 Application of the semantic enrichment

After that, three example regulation rules are selected, including one direct attribute-related requirement (i.e., rule 1) and two indirect attribute-related requirements (i.e., rules 2 & 3). Utilizing the automated rule interpretation method in Section 3.2, the above rules are interpreted into SPARQL codes, which can be used for the reasoning engine, as shown in Fig. 8. Finally, these codes are executed to check the model using GraphDB. Fig. 8 also shows a rule checking application in a BIM model using the three example rules. According to rule 1, columns with a fire resistance limit of 2.0 h, which is less than the requirement of 3 h, were selected. A total of 13 column instances are screened from 302 column instances. The fire protection zone with 2 safety exits that meet the requirement of rule 2 is selected out. Because the plant has 2 fire protection zones, the other one fails to pass the second check. For rule 3, the building is a two-story plant with a fire hazard category of class C and a fire resistance grade of class three. It meets the requirement of the code. The model checker provides the global ID of the selected elements, and the users can locate the elements according to the global ID, as shown in Fig. 8. Thus, this rule checking identifies 13 column instances and 1 fire protection zone instance that failed to meet the requirements. The results are the same as the manual checking results. The automated rule checking successfully detects the issues and can greatly improve the efficiency of the design.

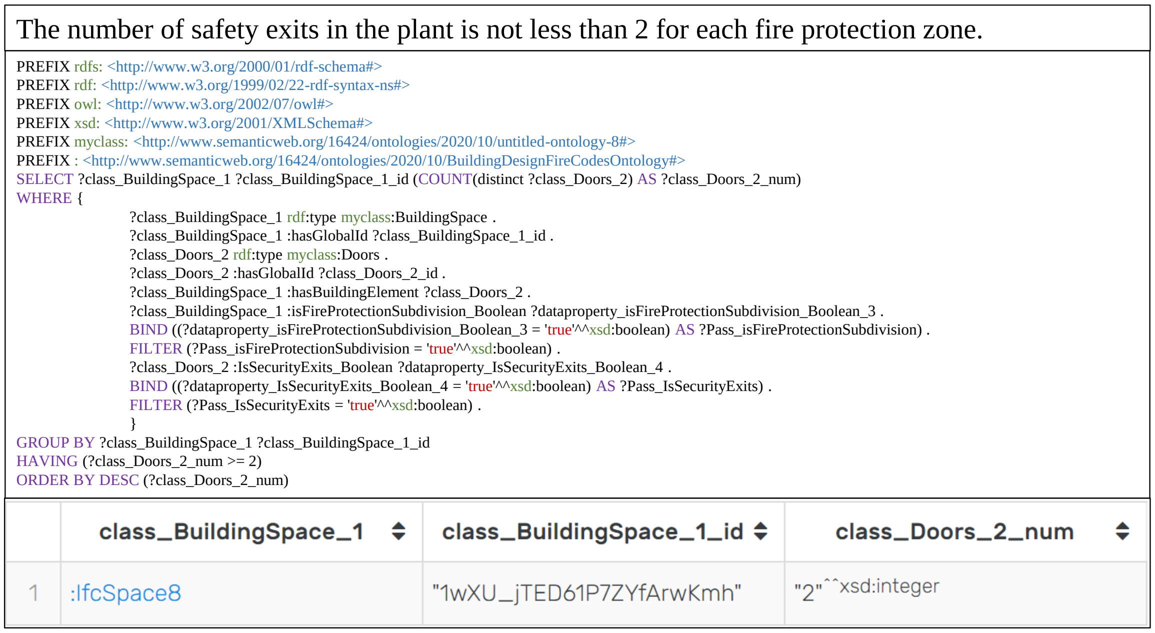


The fire resistance limit of the columns is not less than 3h.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX myclass: <http://www.semanticweb.org/16424/ontologies/2020/10/untitled-ontology-8#>
PREFIX : <http://www.semanticweb.org/16424/ontologies/2020/10/BuildingDesignFireCodesOntology#>
SELECT DISTINCT ?class_Column_1 ?class_Column_1_id
WHERE {
            ?class_Column_1 rdf:type myclass:Column .
            ?class_Column_1 :hasGlobalId ?class_Column_1_id .
            ?class_Column_1 :hasFireResistanceLimits_hour ?dataproperty_hasFireResistanceLimits_hour_2 .
            BIND ((?dataproperty_hasFireResistanceLimits_hour_2 >= '3'^^xsd:float) AS ?Pass_hasFireResistanceLimits) .
            FILTER (?Pass_hasFireResistanceLimits = 'false'^^xsd:boolean) .
            }
```

	class_Column_1	class_Column_1_id
1	:IfcColumn321	"2TJvWVuN51mepwyN3rKILc"
2	:IfcColumn322	"2TJvWVuN51mepwyN3rKILa"
3	:IfcColumn323	"2TJvWVuN51mepwyN3rKILg"
4	:IfcColumn324	"2TJvWVuN51mepwyN3rKILe"
5	:IfcColumn325	"2TJvWVuN51mepwyN3rKILk"
6	:IfcColumn326	"2TJvWVuN51mepwyN3rKILi"
7	:IfcColumn327	"2TJvWVuN51mepwyN3rKILl"
8	:IfcColumn328	"2TJvWVuN51mepwyN3rKILG"
9	:IfcColumn330	"2TJvWVuN51mepwyN3rKILM"
10	:IfcColumn331	"2TJvWVuN51mepwyN3rKILK"
11	:IfcColumn332	"2TJvWVuN51mepwyN3rKILQ"
12	:IfcColumn333	"2TJvWVuN51mepwyN3rKILO"
13	:IfcColumn334	"2TJvWVuN51mepwyN3rKILU"

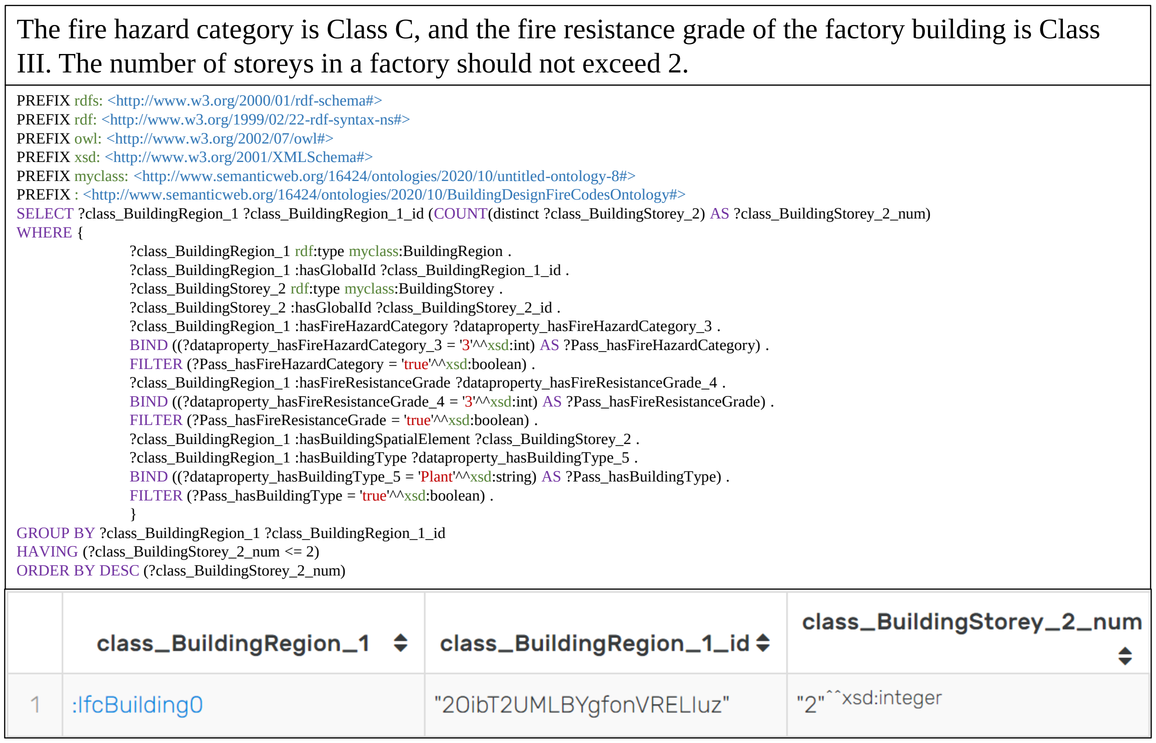
(a) rule1


The number of safety exits in the plant is not less than 2 for each fire protection zone.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX myclass: <http://www.semanticweb.org/16424/ontologies/2020/10/untitled-ontology-8#>
PREFIX : <http://www.semanticweb.org/16424/ontologies/2020/10/BuildingDesignFireCodesOntology#>
SELECT ?class_BuildingSpace_1 ?class_BuildingSpace_1_id (COUNT(distinct ?class_Doors_2) AS ?class_Doors_2_num)
WHERE {
            ?class_BuildingSpace_1 rdf:type myclass:BuildingSpace .
            ?class_BuildingSpace_1 :hasGlobalId ?class_BuildingSpace_1_id .
            ?class_Doors_2 rdf:type myclass:Doors .
            ?class_Doors_2 :hasGlobalId ?class_Doors_2_id .
            ?class_BuildingSpace_1 :hasBuildingElement ?class_Doors_2 .
            ?class_BuildingSpace_1 :isFireProtectionSubdivision_Boolean ?dataproperty_isFireProtectionSubdivision_Boolean_3 .
            BIND ((?dataproperty_isFireProtectionSubdivision_Boolean_3 = 'true'^^xsd:boolean) AS ?Pass_isFireProtectionSubdivision) .
            FILTER (?Pass_isFireProtectionSubdivision = 'true'^^xsd:boolean) .
            ?class_Doors_2 :IsSecurityExits_Boolean ?dataproperty_IsSecurityExits_Boolean_4 .
            BIND ((?dataproperty_IsSecurityExits_Boolean_4 = 'true'^^xsd:boolean) AS ?Pass_IsSecurityExits) .
            FILTER (?Pass_IsSecurityExits = 'true'^^xsd:boolean) .
            }
GROUP BY ?class_BuildingSpace_1 ?class_BuildingSpace_1_id
HAVING (?class_Doors_2_num >= 2)
ORDER BY DESC (?class_Doors_2_num)
```

	class_BuildingSpace_1	class_BuildingSpace_1_id	class_Doors_2_num
1	:IfcSpace8	"1wXU_jTED61P7ZYfArwKmh"	"2"^^xsd:integer

(b) rule2


The fire hazard category is Class C, and the fire resistance grade of the factory building is Class III. The number of storeys in a factory should not exceed 2.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX myclass: <http://www.semanticweb.org/16424/ontologies/2020/10/untitled-ontology-8#>
PREFIX : <http://www.semanticweb.org/16424/ontologies/2020/10/BuildingDesignFireCodesOntology#>
SELECT ?class_BuildingRegion_1 ?class_BuildingRegion_1_id (COUNT(distinct ?class_BuildingStorey_2) AS ?class_BuildingStorey_2_num)
WHERE {
            ?class_BuildingRegion_1 rdf:type myclass:BuildingRegion .
            ?class_BuildingRegion_1 :hasGlobalId ?class_BuildingRegion_1_id .
            ?class_BuildingStorey_2 rdf:type myclass:BuildingStorey .
            ?class_BuildingStorey_2 :hasGlobalId ?class_BuildingStorey_2_id .
            ?class_BuildingRegion_1 :hasFireHazardCategory ?dataproperty_hasFireHazardCategory_3 .
            BIND ((?dataproperty_hasFireHazardCategory_3 = '3'^^xsd:int) AS ?Pass_hasFireHazardCategory) .
            FILTER (?Pass_hasFireHazardCategory = 'true'^^xsd:boolean) .
            ?class_BuildingRegion_1 :hasFireResistanceGrade ?dataproperty_hasFireResistanceGrade_4 .
            BIND ((?dataproperty_hasFireResistanceGrade_4 = '3'^^xsd:int) AS ?Pass_hasFireResistanceGrade) .
            FILTER (?Pass_hasFireResistanceGrade = 'true'^^xsd:boolean) .
            ?class_BuildingRegion_1 :hasBuildingSpatialElement ?class_BuildingStorey_2 .
            ?class_BuildingRegion_1 :hasBuildingType ?dataproperty_hasBuildingType_5 .
            BIND ((?dataproperty_hasBuildingType_5 = 'Plant'^^xsd:string) AS ?Pass_hasBuildingType) .
            FILTER (?Pass_hasBuildingType = 'true'^^xsd:boolean) .
            }
GROUP BY ?class_BuildingRegion_1 ?class_BuildingRegion_1_id
HAVING (?class_BuildingStorey_2_num <= 2)
ORDER BY DESC (?class_BuildingStorey_2_num)
```

	class_BuildingRegion_1	class_BuildingRegion_1_id	class_BuildingStorey_2_num
1	:IfcBuilding0	"2OibT2UMLBYgfonVRELIuz"	"2"^^xsd:integer

(c) rule3

Fig. 8. Application example: use of the proposed ARC system for supporting fire protection design compliance checking

6 Discussion

This work proposes a knowledge-informed automated rule checking framework consisting of ontology-based knowledge modeling, model preparation, rule interpretation, and checking execution. The ontology-based knowledge modeling part integrates the domain-specific concepts, relations, descriptions related to regulatory documents, and domain-specific rules for model enrichment and rule interpretation. The ARC processes are enhanced using domain knowledge. A series of open-source toolkits are utilized to establish the framework. Finally, a demonstration of the rule checking application is conducted in an actual plant building as a proof of concept.

Together with the knowledge-informed automated rule checking framework, methods for semantic alignment and rule interpretation are also introduced to interpret regulation texts into a computer-processable format. Compared to the state-of-the-art methods, the scientific contributions of this study are the following:

1. This work proposes an unsupervised-learning-based semantic alignment method to automatically align the concepts and relations between regulations and BIMs. In addition, knowledge-informed conflict resolution rules are utilized to improve the accuracy of rule interpretation by detecting and resolving potential conflicts of rules. Compared with the widely used keyword mapping method or handcrafted mapping tables, the proposed method requires fewer human efforts in constructing ontologies and achieves the highest accuracy in the validation datasets, improving flexibility and generality.
2. The proposed code generation method supports a wider range of rules, including some complex rules with implicit information that needs to be reasoned. The progress is achieved for two main reasons. On the one hand, text classification considering the computability of the text is integrated into the proposed method. Different categories of rules are subsequently processed using different proper SPARQL functions, improving the accuracy and the applicability. On the other hand, the proposed code generation method utilizes SPARQL rather than widely used plain description clauses, which is suitable for a wider range of rule interpretation and rule checking.

The proposed method is applicable for creating computable rules from various textual regulatory documents. For example, in proactive design, a designer can use the proposed method to create computable rules from specific or customized textual regulatory documents to check the models. The proposed automated rule interpretation method can cover most cases. But there exist really complicated requirements in the building codes where human efforts cannot be avoided. Therefore, we provide a semiautomatic human-computer interaction function. Users can modify the structure of syntax trees or the results of the semantic alignment to ensure 100% accuracy of the interpretation result. Note that the tools used in the proposed framework are all open source and freely available. The prototype tools of the proposed framework and datasets developed in this work can also be obtained from open source, which will also facilitate the development of the ARC.

Four main limitations of this research are identified and will be studied in the future by the authors.

First, the proposed method is only tested in compulsory regulation texts from *GB 50016-2014*. The FPBO cannot be directly applied to other fields, such as earthquake resistance, energy, and sustainability. However, the rule sets developed in this study are reusable and expandable. Future research can be done to enrich the ontology and rule sets to accommodate the regulatory requirements of different fields.

Second, the performance of the end-to-end automated rule interpretation method needs to be further improved. Future researchers can develop a larger dataset and then replace the adopted methods with state-of-art deep learning models to improve the performance. For example, the adopted dictionary-based text classification method and unsupervised learning-based semantic alignment methods can be replaced by an RNN, a CNN, or an attention-based method, which can achieve a comprehensive understanding of a text, to improve the performance.

Third, the scope of rule categories covered by automated rule interpretation needs to be further expanded. Although the direct attribute-related requirements and part of the indirect attribute-related requirements can be automatically interpreted by the proposed end-to-end pipeline, the existential requirements and spatial geometry-related requirements are not yet supported. Besides, some complex rules that require additional information such as reference to rules from other sections, figures, and tables are not yet considered. In the future, the following extensions can be pursued to further improve this research. SPARQL functions for querying spatial geometry-related building data in RDF format should be extended. Approaches for transforming spatial geometry-related building data into RDF format should be further researched. The automated rule interpreting method that considers rules from other sections, figures, and tables, should be further researched.

Fourth, in this work, for research purposes, the attributes of the objects are all stored in the proper place rather than in the "name" or the "description" of the objects. While, in the real world, some of the information that needs to be queried may be stored in other places which may increase the difficulty of parsing files. Future works should pay more attention to accommodating more situations in practice.

7 Conclusion

In this research, we propose a knowledge-informed ARC framework consisting of four parts: ontology-based knowledge modeling, model preparation, rule interpretation, and checking execution. Based on the framework, an ontology is first established to represent domain knowledge, including concepts, synonyms, relationships, constraints, etc. To address the hard-coded manner and improve the generalization of concept alignment, an unsupervised learning-based semantic alignment method and knowledge-informed conflict resolution are proposed and adopted. To identify the proper SPARQL functions for complex rules, a domain-specific text classification method, which can identify the proper code generation method for texts, is proposed and used. Then, the regulatory texts are automatically interpreted into SPARQL, which is suitable for a wider application range for checking rules covering implicit information that need to be reasoned. The proposed algorithms are implemented in a system. To demonstrate the system, a plant building is automatically checked using some rules selected from *GB 50016-2014*. The experimental results indicate a number of contributions. First, the proposed semantic alignment method and conflict resolution method achieve an accuracy of 90.1% and exceed the commonly used dictionary-based matching method by 34.3% accuracy. This means that this method can not only save manual effort in mapping tables or ontology construction but can also improve the accuracy. Second, the proposed rule interpretation method can support a wider range of rules. Third, the efficiency of the proposed rule interpretation method is improved by more than 5 times compared with manual interpretation by experts.

In this study, the tools utilized are all available in open source and free to access, and the prototype of the automated rule interpretation and developed datasets are all available in open source. These contributions can considerably promote the research and application of ARC.

Acknowledgment

The authors are grateful for the financial support received from the National Natural Science Foundation of China (No. 51908323, No. 72091512), the National Key R&D Program (No. 2019YFE0112800), and the Tencent Foundation through the XPLORER PRIZE.

References：

[1] Nawari, N. O. (2018). Building information modeling: automated code checking and compliance processes. CRC Press. https://doi.org/10.1201/9781351200998

[2] Lee, Y. C., Eastman, C. M., & Lee, J. K. (2015). Automated rule-based checking for the validation of accessibility and visibility of a building information model. 2015 International Workshop on Computing in Civil Engineering, pp. 572-579. https://doi.org/10.1061/9780784479247.071

[3] Lee, Y. C., Ghannad, P., Dimyadi, J., Lee, J. K., Solihin, W., & Zhang, J. (2020). A comparative analysis of five rule-based model checking platforms. In Construction Research Congress 2020: Computer Applications, pp. 1127-1136. https://doi.org/10.1061/9780784482865.119

[4] Xu, X., & Cai, H. (2021). Ontology and rule-based natural language processing approach for interpreting textual regulations on underground utility infrastructure. Advanced Engineering Informatics, 48, 101288. https://doi.org/10.1016/j.aei.2021.101288

[5] Eastman, C., Lee, J. M., Jeong, Y. S., & Lee, J. K. (2009). Automatic rule-based checking of building designs. Automation in Construction, 18(8), pp. 1011-1033. https://doi.org/10.1016/j.autcon.2009.07.002

[6] Beach, T. H., Hippolyte, J. L., & Rezgui, Y. (2020). Towards the adoption of automated regulatory compliance checking in the built environment. Automation in Construction, 118, 103285. https://doi.org/10.1016/j.autcon.2020.103285

[7] Ismail, A. S., Ali, K. N., & Iahad, N. A. (2017). A review on BIM-based automated code compliance checking system. In 2017 International Conference on Research and Innovation in Information Systems (ICRIIS), 1-6. https://doi.org/10.1109/ICRIIS.2017.8002486

[8] Fuchs, S. (2021). Natural language processing for building code interpretation: systematic literature review report. https://www.researchgate.net/profile/Stefan-Fuchs-8/publication/351354243_Natural_Language_Processing_for_Building_Code_Interpretation_Systematic_Literature_Review_Report/links/6093496e299bf1ad8d7d8a0f/Natural-Language-Processing-for-Building-Code-Interpretation-Systematic-Literature-Review-Report.pdf

[9] Dimyadi, J., Pauwels, P., & Amor, R. (2016). Modelling and accessing regulatory knowledge for computer-assisted compliance audit. Journal of Information Technology in Construction, 21, pp. 317-336. https://biblio.ugent.be/publication/8041842

[10] Nawari, N. O. (2019). A generalized adaptive framework (GAF) for automating code compliance checking. Buildings, 9(4), 86. https://doi.org/10.3390/buildings9040086

[11] Hjelseth, E., & Nisbet, N. (2011). Capturing normative constraints by use of the semantic mark-up RASE methodology. In Proceedings of CIB W78-W102 Conference, 1-10. https://d1wqtxts1xzle7.cloudfront.net/54368036/Capturing_normative_constraints_by_use_of_the_semantic-with-cover-page-v2.pdf?Expires=1658212065&Signature=UQF8UdUxqr15pebOiMUArEL0qIvaQCv21cKKDzvEQkE6sPaInnr1uV67I0TwNq4uWISWVDi3mrndKOYWapxGYK054qHLwrRqEX7so0Hph2obse67u0Qy146xo9Y7H-bmi66T~rIQ4SRP0-LCa5RgPtx4Yi5uYlIL-WSR1Hh8vLfC-

gdOccfSuZ8SCD0k8EjDyhjhax8rbOX2bX5iXF6FQUWcezGA-CuEY7GvucVhVsTsh60P2Tl2JI96I5U7ALOejh6OwQg9UzCw02DgHKNvQCMMt~1n4II7Dij3q~1quSQCbyx6cNXk3ju7KZBifWWeoDnOqM4Or5LIXdedsGMboA__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

[12] Zhang, J., & El-Gohary, N. M. (2015). Automated information transformation for automated regulatory compliance checking in construction. Journal of Computing in Civil Engineering, 29(4), B4015001. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000427

[13] Zhou, P., & El-Gohary, N. (2017). Ontology-based automated information extraction from building energy conservation codes. Automation in Construction, 74, pp. 103-117. https://doi.org/10.1016/j.autcon.2016.09.004

[14] Zhang, J., & El-Gohary, N. M. (2016). Semantic NLP-based information extraction from construction regulatory documents for automated compliance checking. Journal of Computing in Civil Engineering, 30(2), 04015014. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000346

[15] Zhou, P., & El-Gohary, N. (2021). Semantic information alignment of BIMs to computer-interpretable regulations using ontologies and deep learning. Advanced Engineering Informatics, 48, 101239. https://doi.org/10.1016/j.aei.2020.101239

[16] Zhang, C., Beetz, J., & de Vries, B. (2018). BimSPARQL: Domain-specific functional SPARQL extensions for querying RDF building data. Semantic Web, 9(6), 829-855. https://doi.org/10.3233/SW-180297

[17] Beach, T. H., Rezgui, Y., Li, H., & Kasim, T. (2015). A rule-based semantic approach for automated regulatory compliance in the construction sector. Expert Systems with Applications, 42(12), 5219-5231. https://doi.org/10.1016/j.eswa.2015.02.029

[18] Fenves, S. J. (1966). Tabular decision logic for structural design. Journal of the Structural Division, 92(6), 473-490. https://doi.org/10.1061/JSDEAG.0001567

[19] Nadkarni, P. M., Ohno-Machado, L., & Chapman, W. W. (2011). Natural language processing: an introduction. Journal of the American Medical Informatics Association, 18(5), 544-551. https://doi.org/10.1136/amiajnl-2011-000464

[20] Xu, X., & Cai, H. (2020). Semantic approach to compliance checking of underground utilities. Automation in Construction, 109, 103006. https://doi.org/10.1016/j.autcon.2019.103006

[21] Kuske, D., & Schweikardt, N. (2017, June). First-order logic with counting. In 2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) (pp. 1-12). IEEE. https://doi.org/10.1109/LICS.2017.8005133

[22] Zhou, Y. C., Zheng, Z., Lin J.R., Lu X.Z. (2022). Integrating NLP and Context-Free Grammar for complex rule interpretation towards automated compliance checking. Computers in Industry, 142, 103746. https://doi.org/10.1016/j.compind.2022.103746

[23] Shen, W., Wang, J., & Han, J. (2014). Entity linking with a knowledge base: issues, techniques, and solutions. IEEE Transactions on Knowledge and Data Engineering, 27(2), 443-460. https://doi.org/10.1109/TKDE.2014.2327028

[24] Karadeniz, I., & Özgür, A. (2019). Linking entities through an ontology using word embeddings and syntactic re-ranking. BMC bioinformatics, 20(1), 1-12. https://doi.org/10.1186/s12859-019-2678-8

[25] Morgan, A. A., Lu, Z., Wang, X., Cohen, A. M., Fluck, J., Ruch, P., Divoli, A., Fundel, K., Leaman, P., Hakenberg,

J., Sun, C., Liu, H., Torres, R., Krauthammer, M., Lau, W., Liu, H., Hsu, C., Schuemie, M., Cohen K.& Hirschman, L. (2008). Overview of BioCreative II gene normalization. Genome Biology, 9(2), 1-19. https://doi.org/10.1186/gb-2008-9-s2-s3

[26] Ghiasvand, O., & Kate, R. J. (2014). UWM: Disorder mention extraction from clinical text using CRFs and normalization using learned edit distance patterns. Proceedings of the 8th International Workshop on Semantic Evaluation, 828–832. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.722.4597&rep=rep1&type=pdf

[27] Mohan, S., Angell, R., Monath, N., & McCallum, A. (2021, August). Low resource recognition and linking of biomedical concepts from a large ontology. In Proceedings of the 12th ACM conference on bioinformatics, computational biology, and health informatics (pp. 1-10). https://doi.org/10.1145/3459930.3469524

[28] IFCwiki, Open source projects supporting IFC, http://www.ifcwiki.org/index.php/Open_Source (accessed: June 16, 2021).

[29] Ontotext GraphDB. (2021). GraphDB. https://www.ontotext.com/products/graphdb/ (accessed: June 24, 2021).

[30] O'Sullivan, D. T. J., Keane, M. M., Kelliher, D., & Hitchcock, R. J. (2004). Improving building operation by tracking performance metrics throughout the building lifecycle (BLC). Energy and Buildings, 36(11), 1075-1090. https://doi.org/10.1016/j.enbuild.2004.03.003

[31] Rasmussen, M., Pauwels, P., Lefrançois, M., Schneider, G. F., Hviid, C., & Karlshøj, J. (2017). Recent changes in the building topology ontology. LDAC2017 - 5th Linked Data in Architecture and Construction Workshop. https://hal-emse.ccsd.cnrs.fr/emse-01638305

[32] The Ministry of Public Security of the People's Republic of China (2014). Code for fire protection design of buildings (GB 50016-2014). (in Chinese) https://www.soujianzhu.cn/NormAndRules/NormContent.aspx?id=323 (accessed: July 19, 2022)

[33] Qiu, J., Qi, L., Wang, J., & Zhang, G. (2018). A hybrid-based method for Chinese domain lightweight ontology construction. International Journal of Machine Learning and Cybernetics, 9(9), 1519-1531. https://doi.org/10.1007/s13042-017-0661-0

[34] Musen, M. A. (2015). The protégé project: a look back and a look forward. AI matters, 1(4), 4-12. https://doi.org/10.1145/2757001.2757003

[35] Bloch, T., & Sacks, R. (2020). Clustering information types for semantic enrichment of building information models to support automated code compliance checking. Journal of Computing in Civil Engineering, 34(6), 04020040. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000922

[36] Jurasky, D., & Martin, J. H. (2020). Speech and language processing: an introduction to natural language processing. Computational Linguistics and Speech Recognition, 2020. https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf

[37] Zhang, R., & El-Gohary, N. (2019). A machine learning approach for compliance checking-specific semantic role labeling of building code sentences. In Advances in Informatics and Computing In Civil and Construction Engineering, 561-568. Springer, Cham. https://doi.org/10.1007/978-3-030-00220-6_67

[38] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805. https://doi.org/10.48550/arXiv.1810.04805

[39] Chomsky, N. (1956). Three models for the description of language. IRE Transactions on Information Theory, 2(3), https://doi.org/10.1109/TIT.1956.1056813

[40] Parr, T. (2013). The definitive ANTLR 4 reference. Pragmatic Bookshelf. ISBN: 9781680505016

[41] Lilleberg, J., Zhu, Y., & Zhang, Y. (2015). Support vector machines and word2vec for text classification with semantic features. In 2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing, 136-140. https://doi.org/10.1109/ICCI-CC.2015.7259377

[42] Mikolov, T., Yih, W. T., & Zweig, G. (2013). Linguistic regularities in continuous space word representations. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human language technologies, 746-751. https://aclanthology.org/N13-1090.pdf

[43] Rehurek, R., & Sojka, P. (2010). Software framework for topic modelling with large corpora. In Proceedings of the LREC 2010 workshop on new challenges for NLP frameworks. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.695.4595

[44] Godin, F., Vandersmissen, B., De Neve, W., & Van de Walle, R. (2015, July). Multimedia lab@ acl wnut ner shared task: Named entity recognition for twitter microposts using distributed word representations. In Proceedings of the workshop on noisy user-generated text (pp. 146-153). https://aclanthology.org/W15-4322.pdf

[45] Savigny, J., & Purwarianti, A. (2017, August). Emotion classification on youtube comments using word embedding. In 2017 international conference on advanced informatics, concepts, theory, and applications (ICAICTA) (pp. 1-5). IEEE. DOI: 10.1109/ICAICTA.2017.8090986

[46] Wikimedia. Chinese corpus. https://dumps.wikimedia.org/zhwiki/latest/zhwiki-latest-pages-articles.xml.bz2 (accessed: June 22, 2021).

[47] Soujianzhu. Chinese Rules. https://www.soujianzhu.cn/default.aspx (accessed: June 22, 2021). (in Chinese)

[48] Zheng, Z., Lu, X. Z., Chen, K. Y., Zhou, Y. C., & Lin, J. R. (2022). Pretrained domain-specific language model for natural language processing tasks in the AEC domain. Computers in Industry, 142, 103733. https://doi.org/10.1016/j.compind.2022.103733

[49] Prud'hommeaux, E., Seaborne, A. (2008). SPARQL Query Language for RDF. W3C recommendation. http://www.w3.org/TR/rdf-sparql-query/ . (accessed: June 24, 2021).

[50] Harris, S., Seaborne, A. (2013). SPARQL 1.1 Query Language. W3C Recommendation. https://www.w3.org/TR/sparql11-query/ (accessed: June 24, 2021).

[51] Solihin, W., & Eastman, C. (2015). Classification of rules for automated BIM rule checking development. Automation in Construction, 53, 69-82. https://doi.org/10.1016/j.autcon.2015.03.003

[52] Salama, D. M., & El-Gohary, N. M. (2016). Semantic text classification for supporting automated compliance checking in construction. Journal of Computing in Civil Engineering, 30(1), 04014106. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000301

[53] Caldas, C. H., Soibelman, L., & Han, J. (2002). Automated classification of construction project documents. Journal of Computing in Civil Engineering, 16(4), 234-243. https://doi.org/10.1061/(ASCE)0887-3801(2002)16:4(234)

APPENDIX

Appendix A. Model preparation

IFC is a commonly used collaboration format in BIM-based projects, and many BIM software programs support IFC format. Thus, using IFC files for data exchange can improve the scalability of ARC systems. However, the EXPRESS Schema is adopted in IFC, which cannot be directly parsed by the reasoning engine. To date, several works have been proposed to convert IFC into IFCOWL, but the above methods convert all instances and attributes in IFC to IFCOWL, resulting in information redundancy. In addition, the concepts and relations in the FPBO are different from those in the IFCOWL, so the existing converters cannot be used directly. Therefore, a new RDF converter is customized and developed in this study to convert IFC data into the RDF format. Fig. 1 illustrates the workflow of the model preparation method consisting of two main steps: model conversion and model enrichment.

The model conversion aims to convert the IFC data into RDF format and consists of three sub steps: concept mapping, IFC parsing, and turtle file generation.

1. Concept mapping. The concepts of IFC are aligned with those of FPBO via the mapping table established as shown in Table A.1. The concepts of IFC objects are mapped to the classes in the FPBO; for example, '*IfcBuildingStory*' corresponds to '*BuildingStory*'. The concepts of IFC attributes are mapped to the data properties in the FPBO; for example, '*fire_resistance_rating*' corresponds to '*hasFireResistanceLimits_hour*'. The object properties among instances in FPBO are determined by three relationships in the IFC file, including decomposition, contains, and BoundedBy. First, the domain objects and range objects of the above three relationships can be obtained by parsing IFC, and then the FPBO classes of the objects can be determined. The object property between the two objects can be uniquely determined by a HashMap with two keys, where one key is the class of the domain object and the other key is that of the range object.

2. IFC parsing. Different tools are available for developing in IFC format. The IfcOpenShell (Python), IfcPlusPlus (C++), and xBIM toolkits (.NET) are the most famous. IfcOpenShell is based on OpenCascade technology and features other tools, such as blender importers. This paper utilizes IFCOpenShell to parse IFC files and store the necessary objects and attributes in memory. The in-memory objects and attributes are then mapped to the FPBO concepts according to the aforementioned ontology mapping result.

3. Turtle file generation. Terse RDF Triple Language is a common format for ontology data exchange. Then, the mapped data stored in memory are outputted according to the Turtle syntax into a Turtle file.

In this work, the size of the converted IFCOWL file is nearly 200 times of the .ttl file converted using FPBO and the corresponding converter. The IFCOWL file is too large so the opening and operations (e.g., ontology alignment and model enrichment) via protégé is too slow, which is not suitable for the research purpose.

Model enrichment aims to infer implicit information and supply missing information in the Turtle model. Model enrichment was performed based on the Pellet inference machine and the SWRL rules, which are defined as follows:

Category 1: Transitivity of the contain relationship among spatial elements. For example, space A contains space B, and space B contains space C. Therefore, space A contains space C.

SWRL1: `BuildingRegion(?x) ^ BuildingStorey(?y) ^ BuildingSpace(?z) ^ hasBuildingSpatialElement(?x,?y) ^ hasBuildingSpatialElement(?y,?z) -> hasBuildingSpatialElement(?x,?z)`.

Category 2: Transitivity of the contain relationship between spatial elements and component elements. For example, space A contains space B, and space B contains element C. Therefore, space A contains element C.

SWRL2: `BuildingRegion(?x) ^ BuildingSpace(?y) ^ hasBuildingSpatialElement(?x,?y) ^ BuildingComponentElement(?z) ^ hasBuildingElement(?y,?z) -> hasBuildingElement(?x ,?z)`.

The above SWRL can support the derivation of implicit information by a reasoning engine. SWRL1 can derive the implicit information on which building spaces are contained in a building. SWRL2 can derive the implicit information on which building components are contained in a building.

Table A.1. Part of the ontology mapping dictionary

IFC Schema	FPBO concepts
IfcBuildingStorey	BuildingStorey
IfcBuilding	BuildingRegion
IfcSpace	BuildingSpace
IfcColumn	Column
IfcBeam	Beam
IfcSlab	Floor
IfcWallStandardCase	Wall
IfcWall	Wall
IfcWindow	Window
IfcDoor	Doors
IfcStair	Stairs
GlobalId	hasGlobalId
fire_resistance_rating (user defined attribute)	hasFireResistanceLimits_hour