Automated extraction of attributes of IFC objects based on graph theory and SPARQL query

Xiang Xinglei¹, Ma Zhiliang¹, Huang Huanmin² and Xie Peng²

¹ Department of Civil Engineering, Tsinghua University, Beijing, China
² Qianhai Digital City Technology Co., Ltd, Shenzhen, China

E-mail: xiangxl1996@163.com

Abstract. Building Information Modelling (BIM) has been widely adopted as an effective means for supporting information exchange in Architectural, Engineering and Construction (AEC) industry based on unified and platform-independent standards. Among them, the Industry Foundation Classes (IFC), an ISO standard for BIM, is the most widely used one. However, due to the complexity and flexibility of IFC, the extraction of the attributes of IFC objects is a challenging process for practitioners and software developers in AEC industry, which impedes in-depth application of BIM in many cases. To solve the problem, an approach is proposed in this paper based on graph theory and SPARQL query to automate and simplify the process. The approach consists of four operators, i.e., IFC-to-OWL Convertor, P-Path Acquirer, SPARQL Query Generator, and Ontology Reasoner. IFC-to-OWL Convertor translates IFC instance model into ifcOWL instance model, which can be queried by using SPARQL. P-Path Acquirer obtains all possible Predicate Paths (P-Paths) from an object entity to an attribute entity in the graph created based on ifcOWL schema model. Then SPARQL Query Generator generates SPARQL queries for extracting attributes based on the P-Paths. Finally, the ifcOWL instance model and the SPARQL queries are input into Ontology Reasoner to query attributes of IFC objects. The approach is validated by conducting a case study. The approach contributes to the convenient application of IFC standards in AEC industry.

1. Introduction
The rapid development of Building Information Modelling (BIM) has a huge impact on Architectural, Engineering and Construction (AEC) industry over the past decades. BIM provides an effective means for information exchange and interoperability based on unified data standards. Among them, the Industry Foundation Classes (IFC) developed by buildingSMART International are the most widely used one and has been accepted as an ISO (International Standardization Organization) standard (ISO 10303-11). The standard is defined by using EXPRESS language, while the release also provides the version described in XSD language (ISO 10303-28) alternatively.

IFC model, as a dataset storing building information, is classified into schema model and instance model. IFC schema model is a formal and overall description of the definition and structure of BIM data. It defines a generic data model for storing and exchanging building information, instead of storing information for specific buildings. According to the data model defined in IFC schema model, IFC instance model stores the BIM data for a specific building. Each IFC instance model is composed of entities defined in IFC schema model. For example, IfcWall entity defined in IFC schema model is used
to describe the concept of wall, and IfcWall_23543 can be an entity in a certain IFC instance model to represent a specific wall.

IFC schema model consists of four layers, i.e., resource layer, core layer, shared layer and domain layer [1]. The entities in a former layer can be referenced by a latter layer. Entities in the resource layer describe basic supporting concepts and data structures, for example, IfcPoint represents all points within a Cartesian coordinate. Resource layer entities cannot exist independently, but can only exist if referenced by entities in the other three layers. Except the entities in the resource layer, others all inherit from IfcRoot entity. IfcRoot has the following three sub-entities, i.e., IfcObjectDefinition, IfcPropertyDefinition and IfcRelationship. IfcObjectDefinition represents all objects or processes in AEC industry, specially including all physical product such as roofs, walls and slabs. IfcPropertyDefinition describes the characteristics, i.e., attributes that may attach to IFC entities. Further, attributes of IFC objects can also represented by IFC entities, e.g., IfcMaterial and IfcColourRgb. IfcRelationship summarizes all the relationships among entities. IFC objects in this paper are defined as the entities inherited from IfcObjectDefinition.

IFC objects link with their attribute entities in two ways, as shown in figure 1, i.e., the attributes that are directly attached to the objects (named as direct attributes), and the attributes that are connected with the objects through relationships (named as relationship attributes) [2]. Due to the complexity and flexibility of IFC standard, it is difficult to extract attributes of IFC objects. Indeed, IFC schema model defines a complex data model, consisting of 768 object entities and 1480 attributes (IFC4_ADD1). To make matters worse, for given attributes and IFC objects, the ways that the attributes attach to the objects are not unique and determined, but alternative and flexible [3]. Evidently, correctly and efficiently extracting attributes of IFC objects in IFC instance model is vital for many BIM applications, because it is often needed to obtain its attributes when an IFC object is focused. However, to extract attributes of IFC objects directly, an in-depth understanding of IFC schema model is required, which is hard for practitioners and software developers in AEC industry. To solve the problem, several approaches have been proposed.

![Figure 1. Examples of different types of attributes in IFC standard [2].](image)

An intuitive approach is to develop a specialized query language to extract attributes of IFC objects. Partial Model Query Language (PMQL) is such an early effort [4]. PMQL is an XML-based language and provides functionality select, update, and delete IFC attributes. However, writing PMQL query still requires an in-depth knowledge about IFC standard and XML language. Besides, BIMQL is an ongoing development of IFC query language [5]. Compared with PMQL, BIMQL provides a few pre-defined shortcuts functions for handling commonly used cases to simplify the query writing process, such as retrieving information from property sets that are referred by IFC objects. However, these shortcuts are very limited and it is hard to extract information in the use cases that are not pre-defined.

Furthermore, semantic web technology was introduced to extract attributes of IFC objects. It provides another ontology language, i.e., Ontology Web Language (OWL), to formally represent IFC standards instead of EXPRESS and XSD language [6]. In OWL, a set of data is organized as triples, consisting of a subject, a predicate and an object, as shown in figure 2. Subject is the domain of predicate, and object is the range of predicate. The OWL-version of IFC standard, ifcOWL, has been developed as a
counterpart of the EXPRESS version and recommended by buildingSMART [6]. In ifcOWL, IFC entities are linked with each other by using a range of predicates, named as P-Path in this paper. A structured query language, SPARQL, has been proposed to manipulate and query ifcOWL instance models, so that the process for extracting attributes is standardized and simplified greatly. Zhang et al. developed SPARQL extensions for querying ifcOWL instance model, called BimSPARQL [7], in which, a few new OWL vocabularies were pre-defined to represent the attributes of IFC objects and a set of queries were implemented to retrieve implicit attributes. Additionally, Liu et al. proposed an approach to extract construction-oriented quantity take-off (QTO) attributes based on ifcOWL [8]. However, although related researches indicate that ifcOWL and SPARQL has the great potential for extracting information, it is still challenging for the practitioners and software developers to program the SPARQL queries manually.

Another approach that is based on data format conversion is to translate IFC instance model into graph and apply graph theory for extracting attributes. Ismail et al. demonstrated the potential of using graph theory concepts to manage the huge amount of information and complex relationships of IFC instance models [2]. Tauscher et al. presents an approach towards information retrieval from IFC instance model based on graph theory and shortest path search algorithm [3]. The graph-based method for extracting IFC information does not require the users to have in-depth knowledge about IFC standard. However, current graph-based approaches have the following problems. First, the conversion process from IFC to directed graph is still not uniform. Second, although related researches have provided several successful test cases, a systematic, standardized and generic extraction method has not been established. Third, it is time-consuming to generate a graph based on a heavy IFC instance model and to apply the graph algorithm on it.

This paper proposes an approach to extract attributes of IFC objects automatically based on the integrated application of graph theory and SPARQL query. In the approach, graph theory is used to facilitate generating SPARQL queries for extracting attributes automatically based on the graph created from ifcOWL schema model. Then after converting IFC instance model into ifcOWL instance model, the SPARQL queries are executed to actually extract the attributes. The integration of graph theory and SPARQL query overcomes the shortcomings of above-mentioned approaches from the following aspects. First, the SPARQL queries are generated automatically. Second, graph algorithm is applied on the light-weight schema model instead of heavy instance model to improve extraction efficiency greatly. Third, the whole process is easy to handle and efficient, without requiring in-depth knowledge about IFC standard.

The remainder of the paper is organized as follows. In Section 2, the workflow of the approach is introduced. Section 3 presents the key operators of the workflow. Section 4 verifies the approach by conducting a case study. Section 5 is the conclusion.

2. Workflow of the approach

Figure 3 shows the workflow of the proposed approach. As shown in the figure, the initial input for extracting attributes of IFC objects includes the following four items, i.e., ifcOWL schema model, object entity (such as IfcWall), attribute entity (such as IfcMaterial) and IFC instance model.

The workflow is characterized by four operators, i.e., P-Path Acquirer, SPARQL Query Generator, IFC-to-OWL Convertor and Ontology Reasoner. The former two operators are executed on the ifcOWL schema model to generate the SPARQL queries for extracting attributes automatically. Namely, P-Path Acquirer is used to obtain all possible P-Paths from the attribute entity to the object entity in ifcOWL schema model, and SPARQL Query Generator is used to generate SPARQL query for extracting attributes automatically based on the P-Paths. Then IFC-to-OWL Convertor is used to translate IFC
instance model into ifcOWL instance model, which can be queried by using SPARQL. Finally, the ifcOWL instance model and the generated SPARQL queries are input into Ontology Reasoner. Consequently, the reasoning and query result in the attributes of the IFC objects.

Note that IFC-to-OWL Convertor and Ontology Reasoner can be implemented by using existing and available tools. For IFC-to-OWL Convertor, BuildingSMART provides a tool, i.e., the UGent-Aalto IFC-to-RDF converter (https://github.com/pipauwel/IFCtoRDF), which is a widely-used converter to translate IFC instance model into ifcOWL instance model. For Ontology Reasoner, the reasoning engine Jena can parse ifcOWL instance model and execute SPARQL queries efficiently (https://jena.apache.org/). Therefore, this paper focuses on how to generate SPAQRL queries for extracting attributes of IFC objects, that is, the P-Path Acquirer and SPARQL Query Generator. P-Path Acquirer is named after the meaning of acquiring all P-Paths from an object entity to an attribute entity.

2.1. Basic concept of P-Path
As mentioned in Section 1, data is organized as a series of triples in OWL, consisting of subject (domain of predicate), predicate and object (range of predicate). Also, ifcOWL schema model organizes BIM data into triples. In each triple, the predicate links its domain entity and range entity. For example, `<IfcDocumentInformation, hasDocumentReferences_IfcDocumentInformation, IfcDocumentReference> is one triple, and hasDocumentReferences_IfcDocumentInformation (predicate) links IfcDocumentInformation entity with IfcDocumentReference entity. IfcOWL4_ADD1 schema model contains 24097 such triples in total.

In ifcOWL schema model, an attribute entity often links to an object entity through a range of predicates. As an example, figure 4 shows how IfcProject links with IfcDocumentReference in ifcOWL schema model through several predicates.

Figure 3. Workflow of the approach.

Figure 4. An example of P-Path (from IfcProject to IfcDocumentReference).
In this paper, a P-Path is defined as an ordered list of predicates to link an object entity with an attribute entity in ifcOWL schema model, e.g., as the coloured predicates in figure 4, there are two P-Paths which link IfcProject with IfcDocumentReference. The first P-Path is \(<\text{relatedObjects}_\text{IfcRelAssociates}, \text{relatingDocument}_\text{IfcRelAssociatesDocument}, \text{hasDocumentReferences}_\text{IfcDocumentInformation}>\), and the second is \(<\text{relatedObjects}_\text{IfcRelAssociates}, \text{relatingDocument}_\text{IfcRelAssociatesDocument} >\).

2.2. Create predicate graph based on ifcOWL schema model

To acquire all the P-Paths from an object entity to an attribute entity by using graph theory, the concept of predicate graph (P-Graph) is introduced at first in this paper. P-Graph is defined as a directed graph \(G (V, E)\), where the vertex set \(V\) consists of all the predicates in ifcOWL schema and the edge set \(E\) stands for the reachability between predicates. It is created based on ifcOWL schema model. The reachability between predicates is defined as follows. If the range of predicate I is the super entity of or the same as the domain of predicate II, predicate I and predicate II can be the before-after adjacent vertexes in a P-Path, meaning that it is reachable from predicate I to predicate II and an edge should be added from predicate I to predicate II in the P-Graph.

Figure 5 shows two example cases of reachable predicates in the P-Graph. In Case 1, predicate I is hasAssignments_IfcObjectDefinition, with the range of IfcRelAssigns. Predicate II is relatingGroup_IfcRelAssignsToGroup, with the domain of IfcRelAssignsToGroup. Because IfcRelAssigns is the super-entity of IfcRelAssignsToGroup, hasAssignments_IfcObjectDefinition and relatingGroup_IfcRelAssignsToGroup can be the before-after adjacent vertexes in a P-Path. Then an edge between them will be added into the P-Graph. Similarly, Case 2 shows an example where the range of predicate I (hasProperties_IfcPropertySet) is the same as the domain of predicate II (name_IfcProperty).

Figure 5. Example cases of reachable predicates in the P-Graph.

After adding the edges into edge set according to the above rules, in most cases, the result is complete. But there are some special situations that need to be handled by using the following two operations to modify the edge set of P-Graph, as shown in figure 6.

Operation 1: Add edges between the predicates that have same domain of IfcRelationship and its sub-entities. The rationale for doing this can be explained by using figure 6. The figure shows an example of relatedObjects_IfcRelDefines and relatingPropertyDefinition_IfcRelAssociates, which shares the same domain IfcRelDefinesByProperties. In ifcOWL schema model, IfcPropertySetDefinitionSelect can be linked to IfcObjects through the two predicates, i.e., relatedObjects_IfcRelDefines and relatingPropertyDefinition_IfcRelAssociates, to represent the properties of IFC objects. It means that the two predicates can be the before-after adjacent vertexes in a P-Path and should be connected with an edge in the P-Graph. Therefore, an edge between such predicates needs to be added to modify the P-Graph.
Operation 2: Delete edges between inverse predicates. Inverse predicates in ifcOWL schema model are a pair of predicates with opposite meanings. The inverse predicates are used to describe the same semantics between entities from opposite direction, only one of the both is used to represent the meaning of the specific entities in ifcOWL instance model. It means that inverse predicates cannot be the before-after adjacent vertexes in a P-Path. However, they are considered incorrectly in the above-mentioned two cases and many redundant cycles are introduced into the P-Graph. Therefore, edges for inverse predicates need to be deleted to modify the P-Graph. As an example, in the figure 6, engagedIn_IfcPerson and thePerson_IfcPersonAndOrganization is a couple of inverse predicates. Such inverse predicates are declared explicitly in ifcOWL schema model and ifcOWL4_ADD1 declares 94 pairs of inverse predicates.

| Operation 1 | Add edge between predicates with the same domain of IfcRelationship and its sub-entities |
|-------------|----------------------------------------------------------------------------------------|
| I IfcRelDefinesByProperties relatedObjects_IfcRelDefinesByProperties IfcObject | Edges in P-Graph relatedObjects_IfcRelDefinesByProperties |
| II IfcRelDefinesByProperties relatingPropertyDefinition_IfcRelDefinesByProperties IfcPropertySet DefinitionSelect | relatingPropertyDefinition_IfcRelDefinesByProperties |

| Operation 2 | Delete edge between inverse predicates |
|-------------|----------------------------------------------------------------------------------------|
| I IfcPerson engagedIn_IfcPerson IfcPersonAndOrganization | Edges in P-Graph engagedIn_IfcPerson |
| II IfcPersonAndOrganization thePerson_IfcPersonAndOrganization IfcPerson | thePerson_IfcPersonAndOrganization |

Figure 6. Examples of special operations to modify the edge set of P-Graph.

Finally, P-Graph based on ifcOWL4_ADD1 has been generated in this research, with 1556 vertex and 8794 edges. Figure 7 shows the partial P-Graph with the vertex relatedObjects_IfcRelAssociates.

Figure 7. Partial P-Graph with the vertex relatedObjects_IfcRelAssociates.

2.3. Get all P-Paths on the P-Graph based on depth first search algorithm
Based on the P-Graph, all possible P-Paths from an object entity to an attribute entity can be obtained by using Depth-First Search (DFS) algorithm. DFS is an algorithm for traversing or searching graph data structures, which provides a systematic and effective way to find all the paths between reachable vertexes.
Given the P-Graph, all the P-Paths from an object entity to an attribute entity are obtained through the following two steps.

1. Get Start Predicates set (SP-set) and End Predicates set (EP-set). SP-set is a set of predicates whose domain is the object entity. EP-set is a set of predicates whose range is the attribute entity. SP-set and EP-set are obtained from ifcOWL schema model directly.

2. For each predicate-pair \([sp, ep]\) (\(sp \in SP\)-set, \(ep \in EP\)-set), DFS algorithm is employed to find all the P-Paths from \(sp\) to \(ep\) in the P-Graph. The process is as follows.

Algorithm depth-first search

Input: start predicate (SP), end predicate (EP)

Output: All P-Paths

1. function DFS (SP, EP, Current_Path)
2. if SP==EP:
3.     All_P_Paths ← All_P_Paths + Current_Path
4.     return
5. end if
6. SP. STATE ← ONGOING
7. for \(P \in SP\). Neighbours do
8.     if \(P\). STATE == ONGOING then
9.         Current_Path ← Current_Path + P
10.        DFS (P, EP, Current_Path)
11.        Current_Path ← Current_Path - P
12.     end if
13. end for
14. P. STATE ← DONE

3. SPARQL Query Generator: Generate SPARQL queries based on P-Paths

Given the following three user input items, i.e., an object entity, an attribute entity, all P-Paths from the object entity to the attribute entity. SPARQL Query Generator is used to generate SPARQL queries for extracting attributes of IFC objects automatically. SPARQL Query Generator is implemented through two steps, i.e., to design SPARQL template for extracting attributes, and to make a program which generate queries based on the template and user input items.

3.1. Design SPARQL template for extracting attributes

Because SPARQL is a highly standardized and structured query language, as shown in Listing 1, SPARQL queries template for extracting attributes needs to be pre-designed. In this paper, two templates are designed to extract attribute for type and individual object. In Listing 1, the bold key words need to be filled out by using above mentioned input items. Specially, PREDICATE_x means a vertex within a P-Path from an object entity to an attribute entity and \(m\) is the length of the P-Path.

In Listing 1, Template A is used to extract all objects of specific type and their attribute values from ifcOWL instance model. For example, all walls and their material can be extracted by using Template A. Template B is used to extract the attributes of an individual object. In Template A, object_entity is the type entity of an object, such as IfcWall, while object_entity is the individual name of an object in Template B, such as IfcWall_54627.

3.2. Make a program which generate queries based on the template and user input items

Given SPARQL query template, the above-mentioned three user input items are used to fill out the template to generate the SPARQL queries for extracting attributes, i.e., an object entity, an attribute
entity and the P-Paths from the object entity to the attribute entity. Listing 2 gives an example to show the SPARQL queries generation process by filling out the template using user input items. In Listing 2, the object entity is IfcWall, the attribute entity is IfcMaterial and the P-Path is <relatedObjects_IfcRelAssociates, relatingMaterial_IfcRelAssociatesMaterial, materials_IfcMaterialList, hasContents>. After filling out the Template A by using these input items, the generated SPARQL query is used to extract material for all walls in an ifcOWL instance model.

Listing 1. SPARQL query template for extracting attributes.

| Template A for Type | Template B for Individual |
|---------------------|---------------------------|
| SELECT ?variable_1, ?variable_m+3 WHERE { ?variable_1 rdf:type object_entity ?variable_1 PREDICATE_1 ?variable_2 . ?variable_2 PREDICATE_2 ?variable_3 . ...... ?variable_m PREDICATE_m ?variable_m+1 . ?variable_m+1 rdf:type attribute_entity . ?variable_m+1 ifc:name_attribute_entity ?variable_m+2 . ?variable_m+2 express:hasString ?variable_m+3 } | SELECT ?variable_1, ?variable_m+2 WHERE {? object_entity PREDICATE_1 ?variable_1. ?variable_1 PREDICATE_2 ?variable_2 . ...... ?variable_m-1 PREDICATE_m ?variable_m . ?variable_m rdf:type attribute_entity . ?variable_m ifc:name_attribute_entity ?variable_m+1. ?variable_m+1 express:hasString ?variable_m+2 } |

Listing 2. An example of SPARQL query generation process by filling out the template (SPARQL Query to extract material attribute for walls).

Input items:
(1) object entity: IfcWall (2) attribute entity: IfcMaterial
(3) P-Path: <relatedObjects_IfcRelAssociates, relatingMaterial_IfcRelAssociatesMaterial, materials_IfcMaterialList, hasContents>

SPARQL Query Template A for type
SELECT ?variable_1, ?variable_m+3 WHERE { ?variable_1 rdf:type object_entity ?variable_1 PREDICATE_1 ?variable_2 . ?variable_2 PREDICATE_2 ?variable_3 . ...... ?variable_m PREDICATE_m ?variable_m+1 . ?variable_m+1 rdf:type attribute_entity . ?variable_m+1 ifc:name_attribute_entity ?variable_m+2 . ?variable_m+2 express:hasString ?variable_m+3 . }

Generated SPARQL Query
SELECT ?variable_1, ?variable_7 WHERE { ?variable_1 rdf:type ifc:IfcWall ?variable_2 ifc:relatedObjects_IfcRelAssociates ?variable_1. ?variable_2 ifc:relatingMaterial_IfcRelAssociatesMaterial ?variable_3. ?variable_3 ifc:materials_IfcMaterialList ?variable_4 . ?variable_4 list:hasContents ?variable_5 . ?variable_5 rdf:type ifc:IfcMaterial . ?variable_5 ifc:name_IfcMaterial ?variable_6 . ?variable_6 express:hasString ?variable_7 . }
4. Validation and Case Study
To validate the feasibility and effectiveness of the proposed approach, a prototype composed of P-Path Acquirer and SPARQL Generator is developed by using python, which was used to generate SPARQL queries for extracting attribute based the following input items automatically, i.e., ifcOWL schema model, an object entity and an attribute entity. Then the SPARQL queries and the ifcOWL instance model converted from IFC instance model by using IFC-to-OWL Convertor are input into Ontology Reasoner to extract attribute for IFC objects.

As a case study, the approach is tested by using a public IFC 2x3 instance model available online, which is a duplex apartment BIM model [9], as shown in figure 8. In the IFC instance model, there are 48 IfcWallStandardCase entities to represent walls. The proposed approach is used to extract material attribute for the 48 walls as validation. For this purpose, the object entity is set as IfcWallStandardCase and the attribute entity is set as IfcMaterial.

Consequently, 17 SPARQL queries are generated to query material attribute for walls. Listing 3 shows one of the generated SPARQL queries as an example. The query result indicates the approach extracts attribute for IFC objects completely and correctly, as figure 9 shows. The summary of query results is given in Table. 1.

![IFC instance model of a duplex apartment BIM model.](image1)

Figure 8.

![Jena query result interface for material attribute of all IfcWallStandardCase entities.](image2)

Figure 9.
Listing 3. One P-Path from IfcWallStandardCase to IfcMaterial and corresponding SPARQL Query.

| A P-Path from IfcWallStandardCase to IfcMaterial | A SPARQL query to extract material attribute for walls |
|--------------------------------------------------|-----------------------------------------------------|
| ifc:relatedObjects_IfcRelAssociates \( \downarrow \) ifc:relatingMaterial_IfcRelAssociatesMaterial \( \downarrow \) ifc:forLayerSet_IfcMaterialLayerSetUsage \( \downarrow \) ifc:materialLayers_IfcMaterialLayerSet \( \downarrow \) list:hasContents \( \downarrow \) ifc:material_IfcMaterialLayer | SELECT ?variable_1 ?variable_9
WHERE {
?variable_1 rdf:type ifc:IfcWallStandardCase.
?variable_2 ifc:relatedObjects_IfcRelAssociates ?variable_1.
?variable_2 ifc:relatingMaterial_IfcRelAssociatesMaterial ?variable_3.
?variable_3 ifc:forLayerSet_IfcMaterialLayerSetUsage ?variable_4.
?variable_4 ifc:materialLayers_IfcMaterialLayerSet ?variable_5.
?variable_5 list:hasContents ?variable_6.
?variable_6 ifc:material_IfcMaterialLayer ?variable_7.
?variable_7 rdf:type ifc:IfcMaterial.
?variable_7 ifc:name_IfcMaterial ?variable_8.
?variable_8 express:hasString ?variable_9.
} |

Table 1. Summary of the query result.

| Material                  | Masonry - Brick | Plasterboard | Metal - Stud Layer | Concrete - Cast in Situ |
|---------------------------|-----------------|--------------|--------------------|-------------------------|
| IfcWallStandardCase       | 12              | 23           | 6                  | 7                       |

5. Conclusion
This paper proposed an automatic approach for extracting attributes of IFC objects, based on graph theory and SPARQL queries. In this approach, DFS algorithm are employed to find all the P-Paths from an object entity to an attribute entity in ifcOWL schema model. Then SPARQL queries for extracting attributes are generated by filling out pre-defined templates using the P-Path vertexes. Finally, after converting IFC instance model into ifcOWL format, Jena reasoner is used to parse the instance model and execute SPARQL queries to extract attribute for IFC objects. The correctness and effectiveness are validated by using a test case.

The proposed approach has the following advantages compared with existing researches.

(1) Automation. SPARQL generation process is automatic based on graph algorithm, instead of manual compilation.

(2) User-friendliness. The approach does not require users to have an in-depth knowledge about the underlying structure of IFC standard.

(3) High efficiency. Compared with the existing approaches to translate large-scale instance model into graph, the approach generates P-Path based on schema model, which reduces the computation and complexity.

The approach contributes to the convenient application of IFC standards in AEC industry.
Acknowledgements
The study has been supported by Shenzhen Qianhai Digital City Technology Co. Ltd.

References
[1] BuildingSMART International 2013 IFC4 Documentation Available online: https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/ (accessed on 24 May 2021)
[2] Ismail A and Nahar A 2017 Application of graph databases and graph theory concepts for advanced analysing of BIM models based on IFC standard 24th Proc. Int. Workshop on Intelligent Computing in Engineering (Nottingham) pp 146–157
[3] Tauscher E, Bargstädt H J and Smarsly K 2016 Generic BIM queries based on the IFC object model using graph theory, Int. Conf. 16th on Computing in Civil and Building Engineering (Japan: Osaka)
[4] Adachi Y 2003 Overview of partial model query language Proc. of the 10th ISPE Inte. Conf. Concurrent Engineering (Portugal: Madeira)
[5] Mazairac W and Beetz J 2013 BIMQL - An open query language for building information models Adv. Eng. Inform. 27 444–456
[6] Pauwels P and Terkaj W 2016 EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology Autom. Constr. 63 100-133
[7] Zhang C, Beetz J and Vries B D 2018 BimSPARQL: Domain-specific functional SPARQL extensions for querying RDF building data SWJ. 9 829–855
[8] Liu H, Lu M and Al-Hussein M 2016 Ontology-based semantic approach for construction-oriented quantity take-off from BIM models in the light-frame building industry Adv. Eng. Inform. 30 190–207
[9] DuplexModel-IFC-2011-05-05 2011 Available online:http://smartlab1.elis.ugent.be:8889/IFC-repo/http.openifcmodel.cs.auckland.ac.nz/030811DuplexModel-IFC-2011-05-05/ (accessed on 24 May 2021)