Semantic Web 1 (2017) 1–17
IOS Press

BimSPARQL: A case of extending SPARQL functions for querying linked building data

Chi Zhang [a,*], Jakob Beetz [a]
[a] *Department of Built Environment, Eindhoven University of Technology, P.O. box 513, 5600 MB, Eindhoven, The Netherlands*
E-mail: c.zhang@tue.nl, j.beetz@tue.nl

Abstract. In this paper, we propose to extend SPARQL functions for querying building data. Building models represented by the Industry Foundation Classes data model are the target data sources to develop extended functions. By extending these functions, we attempt to 1) simplify writing queries according to requirement checking use cases, and 2) retrieve useful information implied in 3D geometry data with an open and extensible approach. Extended functions are modelled as RDF vocabularies and classified into groups for further extensions. We combine declarative rules used in the Semantic Web field with procedural programming to implement extended functions. Compared with query techniques developed in the conventional Building Information Modeling domain, we show the added value of such approach by providing an example of querying building and regulatory data, where spatial and logic reasoning can be applied and data from multiple sources are required. It demonstrates an approach that can be extended and applied for many other use cases. Based on the development, we discuss the applicability of proposed approach, current issues and future challenges.

Keywords: BimSPARQL, IFC, ifcOWL, SPARQL, function

1. Introduction

As integrating data in the architecture, engineering and construction (AEC) industry is becoming increasingly important, the methodology of Building Information Modeling (BIM) has been adopted by more and more industry practitioners and has led to the common data standard of Industry Foundation Classes (IFC) [11,22]. Using BIM and IFC applications to create, exchange and analyse data is the state-of-the-art of AEC daily practices. As many researchers have discussed, however, this approach has many problems that might be rooted in the closed world nature of applied conventional data modelling approaches [3]. Even using IFC-based building models, retrieval of domain specific information is currently hard apart from some proprietary solutions. Building models are used for different engineering tasks, where information needs to be derived according to use case requirements. However, BIM data models such as IFC are designed for creation and exchange of product data, but not subjected to various query and analysis tasks. Many useful relationships and properties e.g. typing, properties, spatial and topological relations etc. that are explicitly defined or implied in building models are either difficult or impossible to retrieve. Furthermore, IFC is highly limited by its schema that makes it not flexible enough to adapt to situations when data from different sources needs to be integrated and processed. Although IFC is a central data model aiming to cover the entire AEC industry, commonly used information is not specified within the scope of the IFC meta model schema, including e.g. product classifications, building requirements and regulations, and data from neighbouring industries such as urban planning and sensor networks.

Using the Resource Description Framework (RDF) and Semantic Web technologies to represent building data has been proposed time and again over the last decade [38,3,33]. Unlike conventional data mod-

*Corresponding author. E-mail: c.zhang@tue.nl.

eling approaches that are limited by the scope of their underlying schemas, these semantic technologies provide an open and common environment for sharing, integrating and linking data from different domains and databases. Semantics can be formally defined with the logic basis of these technologies and shared using web-based mechanisms such as Uniform Resource Identifiers (URIs) and the Hypertext Transfer Protocol (HTTP). The ifcOWL ontology, the Web Ontology Language (OWL) and RDF equivalent of IFC data model, has been developed and proposed in the standardization track in the research and industry community as a foundation for Semantic Web applications for the AEC domain [33]. In this context, using a standard language such as SPARQL to process building data becomes more and more practical [15]. In comparison with domain-specific query languages developed in the traditional BIM realm [29,6], this approach is especially applicable for scenarios when knowledge reasoning can be applied and federated data needs to be processed.

By just using ifcOWL and SPARQL, however, some of the aforementioned issues still remain to be addressed. As a language for querying generic RDF data, SPARQL does not contain specific vocabularies or functions to fulfill requirements common to many use cases in the AEC domain. In this paper, we use SPARQL as a base query language and propose to extend it with a set of functions specific for querying building data. This strategy has also been employed in other fields. For example, the Open Geospatial Consortium (OGC) has released GeoSPARQL as a standard set of vocabularies and functions for geospatial data [35]. We argue that the standardization and adoption of GeoSPARQL provides a reasonable indication for the feasibility of a similar approach for the AEC industry.

There are currently three components of the presented BimSPARQL framework: 1) A set of functions modelled as RDF vocabularies that can be used in SPARQL queries (see section 4); 2) A set of query transformation rules to map some of the functions to lower level constructs (see section 5); 3) A module for implementing a few geometry-related functions for deriving implicit information (see section 5). The official IFC documentation and requirement checking use cases are referenced as sources for developing functions [8,36,41,42].

The extended functions in this research are compatible with existing SPARQL environments. With SPARQL as a common interface language, extended functions can be used to query building data alone or combined with data from other sources, which in turn may have their own domain specific functions (Fig. 1). We believe that this is a generic approach that is usable in many different use cases, including e.g. multi-model collaboration, quantity take-off and cost estimation, requirement and code compliance checking etc.. As a W3C standard, SPARQL has been widely implemented by a plethora of RDF Application Programming Interfaces (APIs) and databases, hence many platforms can be used as base environments for implementing extended functions.

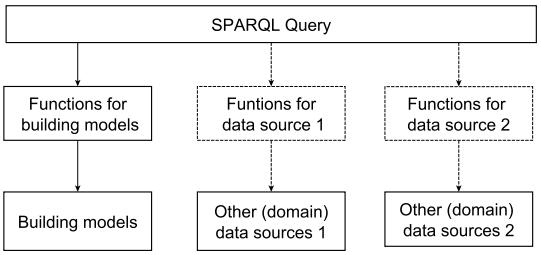

Fig. 1. SPARQL query with domain specific functional extensions

This paper is structured as follows. In section 2, the background of IFC and ifcOWL is briefly introduced and motivation of this research is elaborated. In section 3, an overview of related research is provided. The proposed functional extensions for SPARQL are introduced and classified in section 4, followed by example use cases. In section 5, implementation methods are described and a prototype is presented. In section 6, we provide a case study to evaluate the proposed approach and demonstrate the added value of this approach. A discussion about limitations and further work concludes this paper.

2. Background and motivation

In the last two decades, the IFC standard has been developed and maintained by buildingSMART as a standard data model for data exchanges among heterogeneous applications in the AEC sector [8,11]. The IFC schema is specified using the EXPRESS modeling language [20,37], while its instances are usually formatted in IFC STEP File format [21]. The comprehensiveness of the AEC domain leads IFC to one of the largest EXPRESS-based data models that provides rich constructs for modeling information. For example, the IFC4_ADD1, a recent revision of IFC standard, has

defined 768 entities and 1480 attributes on the schema level [8]. IFC has also provided a few mechanisms to extend semantics in the instance level including e.g. proxy elements, common property sets and external standard classification references. This extent has however formed a burden for the IFC implementation. On the other hand, the semantics required in the AEC industry are still way beyond all the concepts defined in IFC constructs. Therefore, a large amount of information is informally or implicitly represented and usually causes redundancies and ambiguities in IFC instances [43]. As a typical object data model, IFC structures data mainly for the purpose of data creation and exchange rather than for the understanding of the knowledge domain, and information is usually represented using relatively complex structures. All these issues have brought about difficulties regarding data query and management on IFC instance data.

Converting IFC schema and its instances to OWL and RDF was firstly proposed and implemented in [3] to facilitate use cases of data partition, data query and knowledge reasoning. It has been further developed by the buildingSMART Linked Data Working Group (LDWG) and has been in a candidate standard status since 2015 [33]. This approach profits from built-in features and methods used in the Semantic Web field. For example, a common data validation use case that requires every building element should be associated with a building storey can be implemented without hard-coding work [36,48]. The relationship between a building element and the related building storey can be defined using an instance of *IfcRelContainedInSpatialStructure*, which is an objectified relationship defined in IFC. Provided that the building model is represented in the standardized ifcOWL, the query provided in Listing 1 can check this spatial containment relationship using off-the-shelf SPARQL implementation.

As RDF and Linked Data have been growing interests in the AEC industry, it makes sense to use SPARQL as a common language to process federated data sources instead of developing many domain specific languages. Here are some possible scenarios:

- All building objects should be classified according to NL-sfb code.
- The type and thickness of walls can only be modelled according to the right combinations in table X.
- Where are the locations of companies which produce the materials used in walls placed in the hall.

All these tasks not only need to query building models formatted in e.g IFC, but also require data from other sources. We believe that they can be more easily implemented with RDF and SPARQL technologies without relying on proprietary systems.

```
SELECT ?e
WHERE{
  ?e a ifc:IfcBuildingElement .
  FILTER NOT EXISTS{
    ?r ifc:relatedElements ?e .
    ?r a ifc:IfcRelContainedInSpatialStructure
       .
    ?r ifc:relatingStructure ?storey .
    ?storey a ifc:IfcBuildingStorey .
  }
}
```

Listing 1: Query to check the spatial containment relationship for all building elements. [2]

The conversion program between IFC and ifcOWL is almost a one-to-one process, hence the data structures in IFC instances are reflected in the output RDF data. Since standard SPARQL queries are only processed by matching the data graph pattern in RDF, the resulting queries are usually more complex than the logic structures in use cases. For example, in the query case of Listing 1, it is better to have a shortcut relationship between a building element and a storey rather than an objectified one. There are many commonly used structures that can be simplified all over IFC data to simplify query and make properties and relationships closer to the understanding of knowledge domains.

Another problem is that SPARQL can hardly retrieve useful information in the scenarios of geometry and spatial reasoning tasks. Geometry data is the usually the majority in a building model (see Table 7) and can provide large amount of information for domain end users. Although IFC has provided many ways to explicitly model geometry-related properties and topological relationships e.g. the *IfcRelContainedInSpatialStructure* relationship used in Listing 1, they are usually not obligated and not always reliable due to lack of rigidness in IFC data model and AEC domain. Furthermore, there is still much geometry related information impractical or impossible to be explicitly

[2] In this paper, all the used properties defined in ifcOWL use their original IFC names to compact format in query listings e.g. ifc:relatedElements is used to represent the standardized ifc:relatedElements_IfcRelContainedInSpatialStructure.

specified in IFC data. For example, there are some specific topological relationships such as the "touching" relationship between the upper surface of a wall to the bottom surface of the floor slab above (see Listing 7) [41], or some dynamic properties such as distances between elements and escaping routes between spaces [42].

There are many commonly used concepts across use cases. Using the query in Listing 1 as an example, the spatial containment relationship is required in many data validation use cases, and are also important in many cases including e.g. regulatory compliance checking and cost estimation. By wrapping them as functions used in a standard language, we are able to reuse them in many different cases.

3. Related research

3.1. BIM query techniques

There have been many technologies attempting to query and analyse IFC data. Some commercial platforms such as Solibri Model Checker provide functions for querying IFC data [42]. However, the semantics of query functions in these proprietary systems are not transparent and the usage of them is limited by provided interfaces for users.

There are some standard query languages have been used for querying IFC data. As IFC data model is based on EXPRESS, the standard languages of EXPRESS and EXPRESS-X can be used [20]. Some commercial platforms such as Jotne EDModelChecker have taken this approach [23]. However, the EXPRESS language family has not gained popularity outside the STEP initiative in either engineers' or software developers' communities, and there is a very limited set of tools to support them. Some other attempts have used Structured Query Language (SQL) to query IFC data [24,25]. These attempts either have severe performance issues, or are not intuitive enough for end users.

BimQL is the first implemented and open source domain specific query language for querying IFC data [29]. It is implemented in the open source bimserver.org platform [3]. It provides full create, retrieve, update and delete (CRUD) functionalities to manipulate IFC data. Besides using concepts in the IFC schema, BimQL also provides a few shortcut functions for handling common use cases such as deriving information from common modeling constructs in the IFC model referred to as property sets and quantity sets. However, these functions are very limited and BimQL has not been further developed.

Geometry and spatial information in building models is specially focused by a spatial query language introduced in [6]. This approach is further developed as a query language named QL4BIM for querying IFC data [9]. It has provided a few topological and spatial operators and use R-Tree spatial indexes to optimize query performance [13].

There are also query languages tailored for specific use cases such as building code compliance checking. The Building Environment Rule and Analysis (BERA) Language is a domain-specific language dedicated to evaluate building circulation and spatial programs [28]. For this purpose, it has defined a simple internal data model containing a small subset of IFC with related concepts such as floor, space and door etc. Related path-finding algorithms are developed to generate circulation routes between spaces. As a language, however, BERA has limited expressive power and only supports some specific cases on building circulation rules. BIM Rule Language (BimRL) is a more recent research project [10]. It is technically a domain specific query language designed to facilitate accessing information for use cases of regulatory compliance checking. BimRL has provided a suite of components including a simplified data schema and a light weight geometry engine. IFC building models are loaded through an Extract-Transform-Load (ETL) process into data warehouse. The language has an SQL-like syntax to check building models in terms of the defined data schema and implemented functions. It is currently implemented based on a relational database.

The above technologies have provided inspiring domain specific algorithms for querying building data. Currently, however, no query language has been standardized or widely adopted by the AEC industry. We argue that this might be because these technologies are limited by the closed conventional data modeling approaches that are not sustainable in the AEC domain, which continuously needs changes, extensions and customizations according to different contexts and use cases. All these domain specific BIM query languages are designed based on fixed internal data models (usually an IFC equivalent or a simplified subset of it) and additional functions are hard-wired on top of them. Although some of them have provided programming interfaces for further extensions, the development work needs significant efforts and are usually limited by the data captured in its internal data model.

3.2. Applying Semantic Web technologies for querying BIM models

In recent years, Semantic Web and Linked Data technologies have received more and more attention as a knowledge modeling approach in the AEC industry and a number of research prototypes have been developed. A recent and comprehensive overview of them is provided in [34]. Here, we only briefly describe cases related to data query and knowledge reasoning tasks.

Regarding data query for use cases in the AEC domain, one of the early examples is described in [46]. Conformance constraints are interpreted and formalized as SPARQL queries in this paper. A similar method is developed in [7], which has introduced a semi-automatic process to transform regulatory texts to SPARQL queries. A limitation of both efforts is that they mainly focus on formalizing building regulations into a query language without specifying on how to map the used terminologies to building data models.

A number of researchers have applied Semantic Web technologies in different sub-domains in the context of the AEC industry to facilitate knowledge modeling and rule checking. In [30], a remarkable approach for facilitating regulatory compliance checking has been introduced based on N3Logic and EYE reasoning engine [4], and a test case of an acoustic performance checking is presented. In [28], an OWL ontology has been used for reasoning tasks in cost estimation cases. There are also cases regarding energy management and simulation, construction management, and job hazard analysis etc. [1,47,49]. All these examples have proved that different knowledge reasoning tasks in the AEC industry can be facilitated by properly using Semantic Web technologies.

Currently however, a systematic way to query data from building models using Semantic Web technologies is still missing. One of the possible reasons is that a authorative and stable standard ifcOWL ontology has only been established very recently and its adoption in suitable use cases will likely take a few more years. The most similar work that has overlaps with this research are the IfcWoD and SimpleBIM [12,32] ontologies. They both attempt to transform ifcOWL data to a more compact graph to ease query and improve runtime performance. The difference is that they mainly focus on developing a standard ontology as an alternative of ifcOWL to simplify the data graph, while this research is a framework that mainly considers the query functions with respect to semantics in common use cases and further extensions of them. A main outcome of this difference, functions related to geometry data are considered. To our knowledge, it is the first time to combine analyzing IFC geometry data with rule-based reasoning technologies.

3.3. Functional extensions of SPARQL in other industries

Extending SPARQL with additional functions has been proposed and implemented in other fields. The most inspiring ones are geospatial and geographical domains as they have many things in common with AEC industry. The stSPARQL in Strabon and the GeoSPARQL standard from Open Geospatial Consortium (OGC) have specified many topological and geospatial functions for 2D geometry data [26,35] . They have been implemented by spatial database systems including Strabon, Parliament and uSeekM [14]. Some other RDF APIs and triple stores like the Jena framework, Allegrograph and Virtuoso have also implemented geospatial functions. To our knowledge, these vocabularies and functions developed in the Semantic Web world have mainly considered 2D geometry and cannot be directly reused for building models.

The AEC industry also has significant differences from e.g. geospatial field. There are many disciplines and use cases in different contexts, in which the amounts of required properties and relationships are almost unlimited. There are much more sophisticated reasoning tasks related to 3D geometry. Therefore, the system we need is beyond a fixed set of vocabularies but rather a framework that can relatively easy to reuse and extend functions to adapt with different situations.

4. Vocabularies

Building data captured by the IFC data model is the target source for developing extended functions. The IFC documentation and requirement checking use cases from the Dutch Rgd BIM Norm, the Norwegian Statsbygg BIM Manual and in some checks that have been implmented in the Solibri Model Checker and International Building Code are reviewed to determine the structure of vocabularies [8,36,41,42,19]. Most of the referenced cases are data quality validation requirements, which are associated with the IFC data model and are the most fundamental and commonly-used requirement checking cases. From reviewing the above sources, we have extracted many properties and relationships that are repeatedly required in use cases (see

section 4.1, 4.2, 4.3 and 4.4). The implemented functions are wrappers of modular low-level code to derive such information and use them in different scenarios. Due to the complexity of the AEC industry, however, it is not possible for a single organization to list all required functions for all common task scenarios. Instead, they are classified based on required data inputs from IFC building models since they are very much related to further implementations and extensions (see section 5).

Information in IFC-based building models can be roughly grouped into a) domain semantics that usually explicitly represented by e.g. object types, relationships, and properties, and b) geometric data, which is a low-level technical description captured by geometry objects associated with *IfcProduct* instances. Due to the lack of support for parametric geometry description on the levels of the meta model and the implementation, these two kinds of information are almost independent from each other. In fact, building models in real practices often contain information that is inconsistent between these two subsets [39]. We thus argue that query functions should be categorized to identify which subsets of the model are most frequently used to derive data from. As shown in Figure 2 and listed in Table 1, the proposed domain vocabularies are classified into four groups to derive data from these two subsets of either geometric or non-geometric information in IFC models. Sections 4.1 and 4.2 describe functions used to extract information only from domain semantic subset of models, while sections 4.3 and 4.4 describe functions to mainly analyse geometric aspects. Besides these four vocabularies that are defined for building objects, we also propose a vocabulary in section 4.5 to materialize and process geometry data. It is considered as a lower level layer independent with domain information and can provide additional functions for some use cases e.g. example in Listing 7. For each category and subcategory, some function examples are provided to show how to apply them on an ifcOWL instance data set and query examples are provided to demonstrate a use case.

There are generally two ways to extend SPARQL with domain specific functionality. The first method is to add operators in expressions (e.g. *FILTER* expression). The second one is to define a function as an RDF property, which is referred to as a *computed property* or *property function* to generate output based on its bound subject or object [44]. The difference is that a property function is also an RDF property that can have domain(s) and range(s). In the research presented in this paper, most of the extended functions are defined as property functions as we argue that they are more intuitive and can be materialized into RDF graphs for specific applications in order to improve runtime performance. This topic is discussed further in section 7.3. Functions are modelled as RDF vocabularies with their respective URIs. Due to the flexibility and openness of the RDF technology, additional vocabularies can always be added.

4.1. Functions for schema level semantics

Functions in this group are defined to wrap commonly used structures defined on the IFC schema level. We model these functions mainly from the *fundamental concepts and assumptions* specified in the official IFC documentation [8]. These fundamental concepts describe recommended and commonly used structures in IFC instances. Each of the fundamental concepts defines how a domain concept or relationship should be represented in IFC. Many of them have complex structures to represent semantics. For example, if we need to assert the relationship from the Statsbygg BIM norm that "a window is placed in a wall" [41], this is realized in an IFC instance model with two objectified relationships and an opening element as illustrated in Figure 3. Another requirement for wrapping functions is that semantics in IFC data need to be more specified or generalized. For example, the aggregation relationship between spatial objects (e.g. site,building,space)is semantically different from that between building elements (e.g. wall,slab,stair), but in IFC instances they are all represented using the same structure (*IfcRelAggregates*). On the contrary for example, there are several means to assign a material to a building product, while in some use cases they need to be more generalized to simplify queries.

```
SELECT ?window ?wall
WHERE{
  ?window a ifc:IfcWindow .
  ?window schm:isPlacedIn ?wall .
  ?wall a ifc:IfcWall .
  FILTER NOT EXISTS {
    ?wall schm:isContainedIn ?storey .
    ?window schm:isContainedIn ?storey .
    ?storey a ifc:IfcBuildingStorey .
  }
}
```

Listing 2: Query to check the consistency that every window and related walls are contained in the same storey

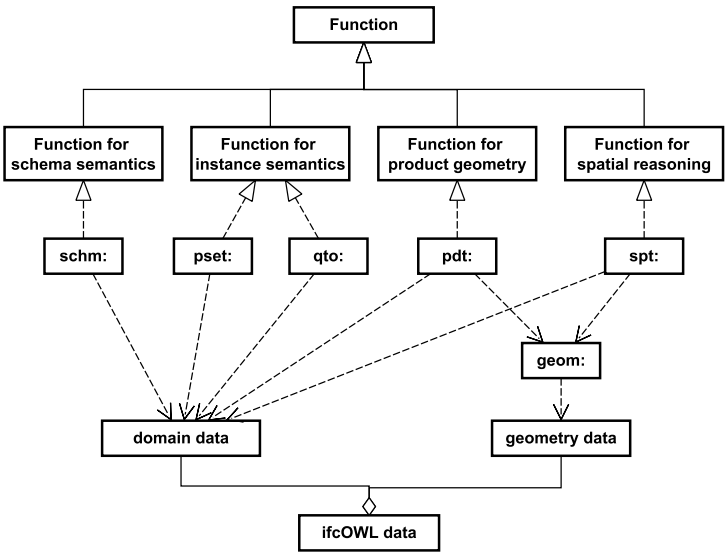

Fig. 2. Conceptual relationships between vocabularies and IFC data

Table 1
Vocabulary prefixes used in this paper and descriptions

Prefix	Description
`schm:`	Shortcut properties and relationships for IFC schema level semantics (see section 4.1)
`pset:`	Short cut properties for instance level property sets (see section 4.2)
`qto:`	Shortcut properties for instance level quantity sets (see section 4.2)
`pdt:`	Properties for single product based on geometry data (see section 4.3))
`spt:`	Properties and relationships based on geometry data of multiple products (see section 4.4)
`geom:`	Lower level geometry library for materializing geometry data and computations on geometry objects (see section 4.5)

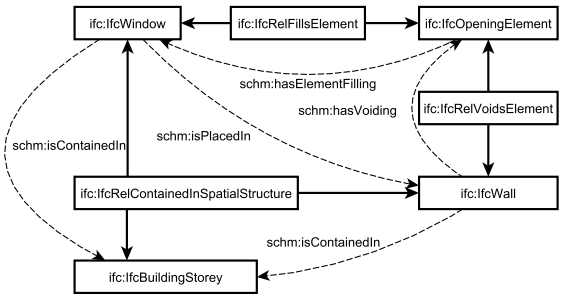

Fig. 3. Example of shortcut functions for schema level semantics

Functions in this category are defined to address these issues. For example, Figure 3 illustrates that direct relationships (dashed line) are defined between building objects. Listing 2 shows an example query to check whether a window and its related wall are contained in the same building storey [41]. This query uses the shortcut functions *schm:isPlacedIn* and *schm:isContainedIn*. Following the same approach, over 40 relationships are wrapped as functions (see Appendix A). Some frequently used examples are listed in Table 2.

Table 2
Example functions for schema level semantics

Prefix	Description
schm:hasProperty	retrieves property instances for an object
schm:hasMatrial	retrieves material instances associated with an object
schm:hasSpaceBoundary	retrieves the boundary elements (e.g. wall, door or virtual boundary) for a space
schm:isDecomposedByElement	retrieves child elements for an element

4.2. Functions for instance level semantics

Functions in this group are provided to represent IFC instance level semantics. As mentioned in section 2, IFC instances can be semantically extended by property sets and quantity sets. Semantics of these extended properties are identified by their names defined in external documentations.

The property sets and quantity sets offically defined by buildingSMART are considered. In total, there are more than 2000 properties and more than 200 quantities grouped within 410 property sets and 92 quantity sets in the official IFC 4 documentation [8]. In this vocabulary, properties and quantities are defined as functions modeled as RDF properties using the prefixes `pset:` and `qto:`. These functions provide shortcuts to directly connect building objects and property values instead of using complex structures in IFC instances.

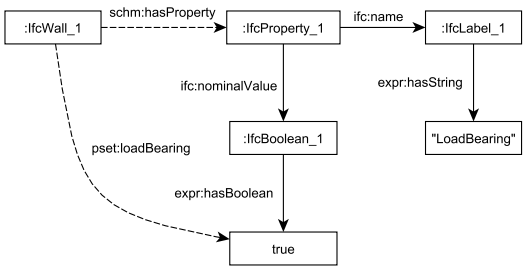

Fig. 4. Example of short cut functions for property sets

```
SELECT ?storey (COUNT(?wall) AS ?q)
WHERE{
  ?wall a ifc:IfcWallStandardCase .
  ?wall pset:loadBearing true .
  ?wall qrw:isContainedIn ?storey .
  ?storey a ifc:IfcBuildingStorey .
} GROUP BY ?storey
```

Listing 3: Query to count load bearing walls for each storey

A typical example is illustrated in Figure 4, a wall that has "LoadBearing" property is represented as an *IfcWall* associated with an *IfcProperty* instance. A shortcut property *pset:loadBearing* is defined to associate the subject with the wall and value of the property instance. Listing 3 shows a quantity take off example to use this function, which is to count the load bearing walls on each building storey. All the properties of simple data types (instances of *IfcPropertySingleValue* and *IfcPhysicalSimpleQuantity*) can use this mechanism and have been defined in our vocabulary. They are the majority in property sets and quantity sets, and they are also most frequently required in use cases.

4.3. Functions for product geometry

Functions in this category are introduced to derive properties based on the geometric representations of a single building product. The vocabulary is identified by the prefix `pdt:`. In IFC model instances, geometry data is represented by geometry objects associated with related building products. The IFC meta model offers a number of means to represent geometry for building products. The most common way is the *Body* representation, which defines 3D volumetric shape of products. However, many geometry types to describe a *Body* geometry in IFC including e.g. Boundary Representation (Brep), Constructive Solid Geometry (CSG) or Non Uniform Rational B-Splines (NURBS). In this research, they are unified as triangulated boundary representation to ease developing analysis algorithms. The 3D geometry representation of a product is either represented by a single triangulated surface or represented by multiple surfaces associated with its child element products (Figure 5).

Based on the triangulated representation, many general geometry properties are derived, including axis-aligned bounding box, minimum volume bounding box, basic dimensions (e.g. height, volume) and partial geometry (e.g. upper surface). Combined with product types and some common assumptions (e.g. a wall length is longer than its thickness), many more specific product properties can be retrieved. These common properties include some defined examples in Table 3.

Although many of these properties can be represented by property sets and quantity sets (see section 4.2), they are not mandatory and are not always reliable in building models [39] . In fact, a typical example of requirement checking is to check the consistency between property sets (or quantity sets) and properties derived from geometry representations [41,42]. Directly deriving information from geometric data provides another dimension to enrich data and ensure consistency. Listing 4 shows an example to compare height of a wall derived from its geometric representation with its height quantity with some tolerance value [42].

Table 3
Example functions for product geometry

Prefix	Description
pdt:hasOverallHeight	returns the height of axis aligned bounding box of a product
pdt:hasUpperSurface	returns the upper surface of a product, which is defined as surfaces that have the highest elevation and have normals of nearly (0,0,1), represented as a well known text string value (see section 4.5)
pdt:hasSpaceArea	returns the bottom surface area of a space
pdt:hasWindowArea	returns the area of the projected profile on the largest surface of the minimum bounding box of a window
pdt:hasGrossWallArea	returns the area of the projected profile on the largest surface of the minimum bounding box of a wall

```
SELECT ?w
WHERE{
  ?w a ifc:IfcWall .
  ?w pdt:hasOverallHeight ?heightg .
  ?w qto:height ?height .
  FILTER (?heightg<?height-0.01||?heightg>?
    height+0.01)
}
```

Listing 4: Query to check the consistency between the height represented as a quantity for a wall and height derived from the geometry representation of a wall

4.4. Functions for spatial reasoning

Functions in this group are to derive information related to spatial reasoning, which needs geometric and location data of multiple building products. This vocabulary is identified by the prefix of `spt:`. They are additionally classified and described in following sections.

4.4.1. Relationships between products

Functions in this category are used to derive relationships between two products. We have defined some general topological relationships that belong to this group, which are applicable for all building products. They are related to many use cases including e.g. clash detection and quantity take-off. Function examples are listed in Table 4.

Table 4
Example functions for relationships between products

Prefix	Description
spt:touches	retrieve touched products or verify whether two products are touched
spt:disjoints	retrieve disjoined products or verify whether two products are disjoined
spt:intersects	retrieve intersected products or verify whether two products are disjoined
spt:contains	retrieve contained products or verify whether a product is contained in another

Listing 5 shows an example to query all walls which touch doors.

```
SELECT ?wall
WHERE{
  ?wall a ifc:IfcWall .
  ?door a ifc:IfcDoor .
  ?wall spt:touches ?door .
}
```

Listing 5: Query to find all walls that touch doors.

4.4.2. Property for groups of products

Functions in this group are used to derive properties for groups of products. Distance between products is a typical example. Many building regulations, building codes and BIM requirement manuals constrain the distance between building components, such as interference between building elements, clearance before openings, heights of floors etc. The exact semantics of the notion "distance" can vary between contexts. We currently offer the concepts in Table 5.

Table 5
Example functions as properties for groups of products

Prefix	Description
spt:distance	return the shortest distance between two products in 3D space
spt:distanceZ	return the vertical distance between bounding boxes of two products
spt:distanceXY	return the shortest distance between the projections of two products on horizontal plane

An example query is provided in Listing 6 to check whether there are columns too close to a window by

selecting pairs of column and window which are on the same storey and have in-between distance shorter than 0.3 meter.

```
SELECT ?column ?window ?d
WHERE {
  ?column a ifc:IfcColumn .
  ?column schm:isContainedIn ?storey .
  ?window schm:isContainedIn ?storey .
  ?window a ifc:IfcWindow .
  (?column ?window) spt:distanceXY ?d .
  FILTER (?d<0.3)
}
```

Listing 6: Query to check the distance between each pair of a window and a column on the same storey

4.4.3. Property and relationships based on spatial relationships

There are also use cases not only require geometry data of referenced products, but also require to process geometry data of other specific types of related building products. For examples, spatially identifying whether a wall is external requires geometry data of all the main building objects e.g. wall, colume, window, and retrieving a walking path between two spaces require geometry data of all the related spaces, obstructs and openings. The semantics of these properties sometimes require knowledge from AEC sub-domains to specify. We currently only provide two example functions listed in Table 6 for this group.

Table 6
Example function as property based on spatial relationships

Prefix	Description
spt:isOutside	spatially identifying whether a building element is outside based geometry and location data of all the main building objects.
spt:hasUpperFloor	returns the above building storey for a building storey and require to process all the floor slabs contained in all building storeys

An example query is shown in Listing 7. It is to check whether every wall touches the bottom surface of a floor slab on the above storey.

4.5. Geometry library

This vocabulary includes geometry related concepts that are materialized in RDF graphs. They are considered as general geometry concepts that provide additional layer independent with domain information. Similar with GeoSPARQL, we define the *geom:Geometry* as the class for geometry objects. As mentioned in section 4.3, triangulated representation is used to represent *Body* geometry data (Fig 5). As geometry data for a product is usually processed as a whole, Extended Well Known Text (EWKT) string literals that have been used in PostGIS are adopted to keep materialized triples in significantly smaller size. Table 7 lists a comparison between triple count of building models in ifcOWL, geometry subsets of them and the triple count of geometry data represented in EWKT format. Besides the triangulated representations that are by default always materialized, the axis aligned bounding boxes and minimum volume bounding boxes for products are also provided in this vocabulary as they are required by many use cases. In future research, other types of geometry representations can also be extended if they are required.

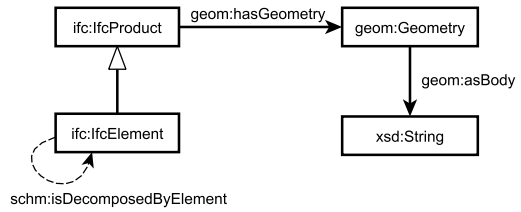

Fig. 5. SPARQL query with domain specific functional extensions

```
SELECT ?wall
WHERE{
  ?wall a ifc:IfcWall .
  ?wall schm:isContainedIn ?storey .
  ?storey spt:hasUpperFloor ?storey2 .
  FILTER NOT EXISTS{
    ?slab schm:isContainedIn ?storey2 .
    ?slab a ifc:IfcSlab .
    ?wall pdt:hasUpperSurface ?ws .
    ?slab pdt:hasBottomSurface ?ss .
    FILTER (geom:touches3D(?ws,?ss))
  }
}
```

Listing 7: Query to select all walls which do not touch any floor slab on the above floor

Another requirement that can possibly be addressed by this vocabulary is to process temporarily generated geometry data in query runtime. The query example in List 7 shows a use case of them. In this example, the upper surface and bottom surface are derived in

Table 7
Triple count in ifcOWL instances, the geometry subsets of them, and geometry triples represented using `geom:` and EWKT

Id	Model	Triples	Geometry Triples	EWKT Geometry Triples
M1	Duplex_A_20110505.ttl	298,085	222,212	819
M2	091210Med_Dent_Clinic_Arch.ttl	1,763,837	1,088,878	10,029
M3	161210Med_Dent_Clinic_Combined.ttl	14,487,715	12,580,688	22,578

query runtime as partial geometries of a wall and a slab, and they are additionally checked by the function *geom:touches3D* to identify their topological relationships.

5. Implementation

In our implementation prototype of the proposed functions, we attempt to minimize hard coding to make defined functions more portable, more transparent for public reviews and easier to adapt and extend by the research and development communities. Declarative rules are used to specify the semantics of many functions and map them to ifcOWL data. Functions defined in section 4.1 and 4.2 can all be implemented by rule languages e.g. SWRL, N3Logic or by SPARQL itself using frameworks like SPARQL Inferencing Notation (SPIN) [18,4,40]. We choose SPIN for the implementation, as it uses SPARQL and already has a few open source implementations which enhance future compatibility. SPIN provides a meta modeling vocabulary to wrap SPARQL queries as functions and allows their cascading use. For example, the function of *schm:isContainedIn* in Listing 2 is implemented by associating it with the query in Listing 8. It will trigger this query as a subquery when it is called. For the flexible reuse, customization and extendability of such domain-specific functionalities, such modular abstraction is benificial.

When dealing with geometry related reasoning tasks, rule languages are usually not sufficiently expressive to implement sophisticated and computational intensive algorithms. Geometry data in IFC or ifcOWL is preprocessed and transformed to RDF data represented by the vocabulary described in section 4.5. Functions described in section 4.3, 4.4 and 4.5 are implemented using low-level procedural programming using e.g. geometry kernels. Many existing general geometry and domain specific algorithms that can be reused and a range of implementations is available.For example, functions in section 4.4.1 are implemented by computing on triangles of both products to determine their relations, similar with algorithms described in [9].

```
SELECT ?a2
WHERE {
  ?a1 ifc:relatedElements ?arg1 .
  ?a1 ifc:relatingStructure ?a2 .
  ?a1 a ifc:IfcRelContainedInSpatialStructure
  ?arg1 a ifc:IfcElement .
  ?a2 a ifc:IfcSpatialStructureElement .
}
```

Listing 8: Query to implement the function *schm:isContainedIn*

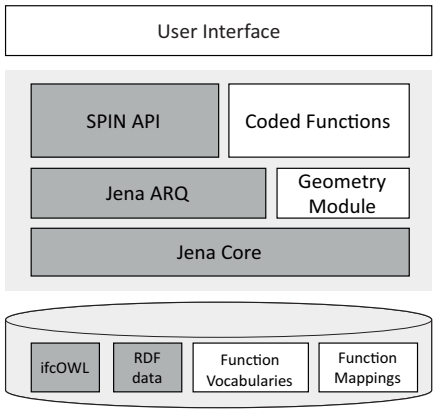

Fig. 6. Implementation architecture (blank blocks are added modules)

The functional extensions introduced here are implemented based on the Open Source Apache Jena framework and SPIN API (see Fig. 6). In this implementation, all the extended functions are processed at query runtime in a backward chaining order. The data flow is illustrated in Figure 7. The ifcOWL instances and SPARQL queries are the input of the system. Depending on the size of ifcOWL files, we can choose to load them into memory or materialize them into a graph persisted into a Jena TDB triple store. When a property function is referred to during a query execution, a SPIN rule as a subquery or a snippet of pro-

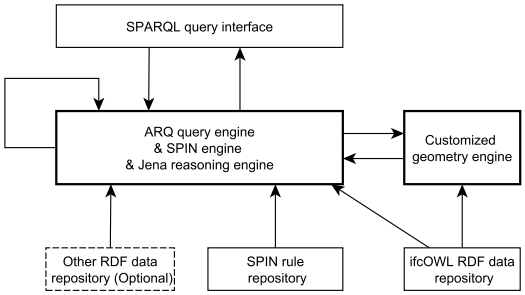

Fig. 7. Data flow of querying and reasoning process

gramming code to retrieve related values is triggered. Since a SPIN rule is also a SPARQL query that can call extended functions, this process recursively continues until no functions are left to be called. This process is compatible with other reasoning technologies.

6. Application example

The advantage of using SPARQL and Semantic Web technologies as the base query environment is that they can be leveraged to facilitate knowledge reasoning and data integration tasks. With these capabilities, defined functions are more easily be reused and extended for specific applications. An application example is presented in this section in a regulatory compliance checking scenario that requires to extend case specific functions to query both building models and regulatory data.

The example listed as follows is taken from the International Building Code (IBC), which is developed by the International Code Council (ICC) and used as a base code standard in United States [19]. This rule example is from the *Chapter 7 Fire and Smoke Protection Features*, and is used to check opening area on external walls to evaluate their fire performance. This example requires to process domain semantic data and geometry data in building models and external tabulated data defined in the IBC document.

– *705.8.4 Where both unprotected and protected openings are located in the exterior wall in any story of a building, the total area of openings shall be determined in accordance with the following:*

$$(Ap/ap) + (Au/au) \leq 1 \quad (1)$$

where:
Ap = Actual area of protected openings.
ap = Allowable area of protected openings.
Au = Actual area of unprotected openings.
au = Allowable area of unprotected openings.

Additionally, the allowable opening areas for protected and unprotected openings (*ap* and *au*) are determined by the Table 705-8 in IBC that describes their relations with fire separation distance. Table 6 shows one row of it, which defines that when the fire separation distance is between 15 to 20 feet and the opening is unprotected and the space is non-sprinklered, the allowed proportion (*au* in the equation) between opening area and external wall area is up to 25 percent.

Table 8
One row of Table 705-8 in International Building Code [19]

Fire separation distance	Degree of opening protection	Allowable area
15 to less than 20	Unprotected, Non-sprinklered	25%

In this example, external wall instances in a dataset have to be checked and analysed to derive related properties and relationships. In addition to the data captured in the IFC building data sets, the referenced table in this example can be considered as a small dataset that needs to be processed to derive allowable protected openings and unprotected openings for each wall. It is transformed to RDF format with the methodology in [45] and processed along with the building model. A general algorithm in a procedural pseudocode notation is specified in Algorithm 1 to check building models and find out external walls which violate this requirement.

Case specific functions are extended for deriving some of these properties based on functions provided in BimSPARQL. For example, the value ***Ap*** used in Algorithm 1 is specified as the proportion between all the protected windows in the wall and the gross area of the wall. Based on predefined functions in BimSPARQL and SPIN rules, this function can be extended with the query in Listing 10 (see Appendix B). Using the same method, functions are extended for this case and listed in Table 9. SPIN rules for defining these case specific functions based on ifcOWL and BimSPARQL vocabularies are listed in Appendix B. As the Table 705-8 in IBC is also processed, a function ibc:allowableArea_T705-8 is also defined to process this external dataset.With all these functions extended for this case, the query in Listing 10 is used to check the opening area of all external walls.

Algorithm 1. procedure for checking rule 705.8.4

1: **for** each external wall ***w*** **do**
2: compute the area percentage of protected openings ***Ap***;
3: compute the area percentage of unprotected openings ***Au***;
4: compute the fire separation distance ***fsd***;
5: find the sprinkler status of related space ***sp***;
6: find the allowable area of protected and unprotected openings ***ap*** and ***au*** from Table 705.8 with ***fsd*** and ***sp***;
7: check the equation (1) with ***Ap***, ***Au***, ***ap***, ***au*** and return result for ***w***;
8: **end for**

```
SELECT ?wall
WHERE{
  # find external walls
  ?wall a ifc:IfcWall .
  ?wall pset:isExternal true .
  # compute Ap and Au values
  ?wall ibc:hasAp ?Ap .
  ?wall ibc:hasAu ?Au .
  # compute fire separation distance
  ?wall ibc:hasFireSeparationDistance ?d .
  # find sprinkler status of related space
  ?wall schm:isContainedIn ?storey .
  ?storey pset:sprinklerProtection ?bool .
  # find ap and au values from the table
  BIND (ibc:allowableArea_T705-8(?d,true,?
    bool) AS ?ap) .
  BIND (ibc:allowableArea_T705-8(?d,false,?
    bool) AS ?au) .
  # filter out walls that have issues
  FILTER ((?Ap/?ap+?Au/?au)>1) .
}
```

Listing 9: Query to check the spatial containment structure for all building elements not related to a building storey.

As proof of concept, a building model that has 189,778 triples is checked using the query in Listing 9. This prototype implementation generates a visualization of the result that is provided in Figure 8.

7. Discussion

The aim of this research is to provide an extensible framework to simplify query and retrieve useful information from IFC based building models according to use case requirements. Added value and current limitations are discussed here.

Table 9
Extended functions for the rule case 705.8.4 in IBC

Prefix	Description
ibc:hasAp	retrieve the proportion between all the protected windows in the wall and the gross area of the wall
ibc:hasAu	retrieve the proportion between all the unprotected windows in the wall and the gross area of the wall
ibc:hasFireSeparationDistance	retrieve the shortest horizontal distance between a wall and lot lines
ibc:allowableArea_T705-8	retrieve allowable area (au or ap) from Table705-8 based on fire separation distance and sprinkler protection status

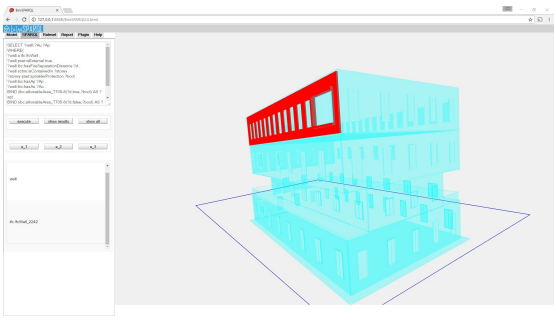

Fig. 8. Snapshot of the query result of the GUI

7.1. Flexibility

In comparison with domain specific query languages that are developed from scratch, the approach introduced in this paper leverages Semantic Web technologies to provide a more interoperable, modular and flexible mechanism to extend functions in order to address the many use cases for information extraction and validation of the AEC industry. As shown in section 6, query functions for specific use cases can be easily extended by adding additional declarative rules based on procedural functionality. They are modular and flexible to adapt to the various forms of presenting facts in IFC data sets. For example, a protected opening in another building case, created by another author using a different BIM authoring tool might be different from how it is defined in Listing 14. It is easier to change or replace this rule without affecting other rules. External, linked datasets can be addressed using the same technology as long as they are captured as RDF or provide SPARQL endpoint services [5,16].

From the use case perspective, a current limitation of this approach is related to query functions that require to instantiate classes. For example, identifying the shortest path between two rooms usually needs to instantiate a path object. Even with procedural coding, SPARQL functions are not suitable to create new resources as they have side effect for the original RDF graphs. More investigations are required to properly adapt this type of query functions in an RDF and Semantic Web environment.

7.2. Portability

Declarative rules can enable more portable implementations for functions. As many functions are defined using SPIN rules, they can be reused by query environments which have implemented SPIN (e.g. Topbraid SPIN API or Eclipse RDF4J) and potentially be reused by those which have implemented SPARQL. Hardwired functions that are wrapped by SPARQL can be implemented on a variety of e.g. geometry engines.

7.3. Query performance

At present query performance is not in the main focus of the research presented here. It depends on the implementation of the rule languages and geometry analysis algorithms that are used. The prototype implementation has used the SPIN framework, which is based on the Jena ARQ query engine and Jena TDB triple store. Therefore, query performance in this aspect mainly depends on the optimization mechanism in Jena to preferentially execute the most selective triple patterns in SPARQL queries. Extensive performance tests using SPIN and the Jena TDB triple store along with comparisons to other rules languages and implementation have been published in [31]. A limitation of the current implementation is mainly related to geometry-related functions. By using a plain RDF triple store such as the Jena TDB in a normal laptop environment, the query in Listing 5 runs around 28 seconds to retrieve 149 results on the M3 building model in Table 7 (with 14,487,715 plus 22,578 triples). The function *spt:touches* computes 1430*247 times on 1430 walls and 247 doors. In future developments, this process can be optimized by e.g. integrating spatial indices and caching mechanisms.

As RDF graphs are flexible, another direction for optimizing performance might be to materialize required triples into RDF graphs for specific applications. As described in section 4.5, besides ifcOWL data only the triangulated boundary representation of products are currently materialized and all the functions are processed at runtime. Some of the geometry-related algorithms are computational intensive (e.g.compute the minimum volume bounding box of a product). We recommend to pre-process such data sets and to materialize their results into the RDF datasets. For specific applications, functions can also be treated as normal RDF properties. For example, if an application frequently requires upper boundary surfaces of products, we recommend to treat the property of *pdt:hasUpperSurface* as a normal RDF property to materialize triples. A dynamic approach to automatically materialize triples according to use cases needs additional investigation and future research.

8. Conclusion

This research provides a general framework to define and extend functions for querying IFC-based building data. A set of functions are classified and introduced and two different approaches are used to implement them. The work presented here should be regarded as a general framework and proof of concept for a modular, scaleable approach to address the large amounts of domain specific query requirements in the AEC domain. As more and more data is represented by RDF and Linked Data technologies, this approach has considerable advantages over the current practices to process building related data in proprietary information silos using one-of-a-kind island solutions.

The link of the vocabularies, transformation rules and source code repository of the prototypical reference implementation is provided in Appendix A. As discussed in section 7, the current performance shortcomings of the solutions introduced here are mainly related to the implementation issues. More efficient algorithms e.g. to pre-process datasets according to spatial indices need to be developed. In future research, more use cases should be investigated and implemented to gradually extend the functionality for specific subdomains in AEC industry and to combine data from different sources. From a technical perspective, optimization for query performance is necessary as it is important for the scalability of this approach.

References

[1] K. Baumgartel, M. Kadolsky and R.J. Scherer, (2014). An ontology framework for improving building energy performance by

utilizing energy saving regulations. In Proceedings of European Conference on Product and Process Modelling (pp. 519-526).

[2] J. Beetz, L. van Berlo, R. de Laat and P. van den Helm, (2010). BIMserver.org-An open source IFC model server. In Proceedings of the CIB-W78 conference.

[3] J. Beetz, J.P. van Leeuwen and B. de Vries, (2009). "IfcOWL: A case of transforming EXPRESS schemas into ontolo-gies." Artificial Intelligence for Engineering Design, Analysis and Manufacturing. vol. 23. no. Special Issue 01. 89-101.

[4] T. Berners-Lee, D. Connolly, L. Kagal, Y. Scharf, and J. Hendler, (2008). N3logic: A logical framework for the world wide web. Theory and Practice of Logic Programming, 8(03), pp.249-269.

[5] C. Bizer and R. Cyganiak, (2006). D2r server-publishing relational databases on the semantic web. In Poster at the 5th International Semantic Web Conference (pp. 294-309).

[6] A. Borrmann and E. Rank, (2009). Topological analysis of 3D building models using a spatial query language. Advanced Engineering Informatics, 23(4), 370-385.

[7] K.R. Bouzidi, B. Fies, C. Faron-Zucker, A. Zarli and N.L. Thanh, (2012). Semantic web approach to ease regulation compliance checking in construction industry. future internet, 4(3), pp.830-851.

[8] buildingSMART International, (2013). Industry Foundation Classes Release 4 (IFC4). Available on: http://www.buildingsmart-tech.org/ifc/IFC4/final/html/.

[9] S. Daum and A. Borrmann, (2014). Processing of topological BIM queries using boundary representation based methods. Advanced Engineering Informatics, 28(4), 272-286.

[10] J. Dimyadi, W. Solihin, C. Eastman, R. Amor, (2016). Integrating the BIM Rule Language into Compliant Design Audit Processes. Proceedings of 34th CIB W78 conference, October 31st to November 2nd, Brisbane, Australia.

[11] C. M. Eastman, P. Teicholz, R. Sacks and K. Liston, (2011). BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors. John Wiley & Sons.

[12] T. M. Farias de, A. Roxin and C. Nicolle, (2015). IfcWoD, semantically adapting IFC model relations into OWL proper-ties. arXiv preprint arXiv:1511.03897.

[13] A. Guttman, (1984). R-trees: a dynamic index structure for spatial searching (Vol. 14, No. 2, pp. 47-57). ACM.

[14] G. Garbis, K. Kyzirakos and M. Koubarakis, (2013). Geographica: A benchmark for geospatial rdf stores (long ver-sion). In The Semantic Web-ISWC 2013 (pp. 343-359). Springer Berlin Heidelberg.

[15] S. Harris, A. Seaborne and E. Prudhommeaux, (2013). SPARQL 1.1 query language. W3C Recommendation, 21.

[16] T. Heath and C. Bizer, (2011). Linked data: Evolving the web into a global data space. Synthesis lectures on the semantic web: theory and technology, 1(1), 1-136.

[17] J. Herring, (2011). OpenGIS Implementation Standard for Geographic information-Simple feature access-Part 1: Common architecture. OGC Document, 4(21), pp.122-127.

[18] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, (2004). SWRL: A semantic web rule language combining OWL and RuleML. W3C Member submission, 21, p.79.

[19] IBC. (2006). International building code. International Code Council, Inc.(formerly BOCA, ICBO and SBCCI), 4051, pp.60478-5795.

[20] ISO, (1994). ISO 10303-11: Industrial automation systems and integration-Product data representation and exchange-Part 11: Description methods: The EXPRESS language reference manual. International Organization for Standardization.

[21] ISO, (2002). ISO 10303-21: Industrial automation systems and integration - Product data representation and exchange - Part 21: Implementation methods: Clear text encoding of the exchange structure. International Organization for Standardization.

[22] ISO, (2013). ISO 16739:2013 Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries. International Organization for Standardization.

[23] Jotne, (2005) . Jotne EDMmodelServer. Available on: http://www.jotneit.no/edmmodelserver-ifc.

[24] H.S. Kang, and G. Lee, (2009). Development of an object-relational IFC server. Proceedings of ICCEM/ICCPM, 2009.

[25] A. Kiviniemi, M. Fischer and V. Bazjanac, (2005). Integration of multiple product models: Ifc model servers as a potential solution. In Proc. of the 22nd CIB-W78 Conference on Information Technology in Construction.

[26] K. Kyzirakos, M. Karpathiotakis and M. Koubarakis, (2012). Strabon: a semantic geospatial DBMS. In International Semantic Web Conference (pp. 295-311). Springer Berlin Heidelberg.

[27] J.K. Lee, C.M. Eastman and Y.C. Lee, (2015). Implementation of a BIM domain-specific language for the building environment rule and analysis. Journal of Intelligent & Robotic Systems, 79(3-4), p.507.

[28] S.K. Lee, K.R. Kim and J.H. Yu, (2014). BIM and ontology-based approach for building cost estimation. Automation in construction, 41, pp.96-105.

[29] W. Mazairac and J. Beetz, (2013). BIMQL-An open query language for building information models. Advanced Engi-neering Informatics, 27(4), 444-456.

[30] P. Pauwels, D. Van Deursen, R. Verstraeten, J. De Roo, R. De Meyer, R. Van de Walle and J. Van Campenhout, 2011. A semantic rule checking environment for building performance checking. Automation in Construction, 20(5), pp.506-518.

[31] P. Pauwels, T. M. de Farias, C. Zhang, A. Roxin, J. Beetz, J. De Roo and C. Nicolle, (2016). Querying and reasoning over large scale building data sets: an outline of a performance benchmark. In Proceedings of the International Workshop on Semantic Big Data (SBD '16). ACM, New York, NY, USA, , Article 11 , 6 pages.

[32] P. Pauwels and A. Roxin, (2016). SimpleBIM: From full ifcOWL graphs to simplified building graphs. eWork and eBusiness in Architecture, Engineering and Construction: ECPPM 2016: Proceedings of the 11th European Conference on Product and Process Modelling (ECPPM 2016), Limassol, Cyprus, 7-9 September 2016.

[33] P. Pauwels and W. Terkaj, (2016). EXPRESS to OWL for construction industry: towards a recommendable and usable ifcOWL ontology. Automation in Construction, 63, 100-133.

[34] P. Pauwels, S. Zhang, Y.C. Lee, (2017) .Semantic web technologies in AEC industry: A literature overview, Automation in Construction, Volume 73, January 2017, Pages 145-165.

[35] M. Perry and J. Herring, (2012). OGC GeoSPARQL-A geographic query language for RDF data. OGC Implementa-tion Standard. Sept.

[36] D. Rillaer van, J. Burger, R. Ploegmakers and V. Mitossi, (2012). Rgd BIM Standard, version 1.0.1. 1-29.

[37] D.A. Schenck and P.R. Wilson, (1994). Information modeling the EXPRESS way. Oxford University Press.

[38] H. Schevers and R. Drogemuller, (2005). Converting the Industry Foundation Classes to the Web Ontology Language, in: Semantics, Knowledge and Grid, 2005. SKG'05. First International Conference on. Beijing, China Nov. 27, 2005 to Nov. 29, 2005
[39] W. Solihin, C. Eastman, Y.C. Lee, (2015). Toward robust and quantifiable automated IFC quality validation. Advanced Engineering Informatics, Volume 29, Issue 3, pp. 739-756.
[40] SPIN. (2011). SPARQL Inferencing Notation. http://spinrdf.org/.
[41] Statsbygg, (2011). Statsbygg Building Information Modelling Manual Version1.2. Available at: http://www.statsbygg.no/bim, accessed January 2014.
[42] Solibri, (2000) . Solibri Model Checker . Available at: https://www.solibri.com/products/solibri-model-checker/, accessed January 2016.
[43] M. Venugopal, C.M. Eastman, R. Sacks, J. Teizer, (2012). Semantics of model views for information exchanges using the industry foundation class schema, Advanced Engineering Informatics, Volume 26, Issue 2, April 2012, Pages 411-428,
[44] W3C (2014). SPARQLExtensionsComputed Properties. https://www.w3.org/wiki/SPARQL/Extensions/Computed_Properties.
[45] W3C (2015). Generating RDF from Tabular Data on the Web. https://www.w3.org/TR/2015/WD-csv2rdf-20150416.
[46] A. Yurchyshyna and A. Zarli, (2009). An ontology-based approach for formalisation and semantic organisation of conformance requirements in construction. Automation in Construction, 18(8), pp.1084-1098.
[47] B.T. Zhong, L.Y. Ding, H.B. Luo, Y. Zhou, Y.Z. Hu, H.M. Hu, (2012). Ontology-based semantic modeling of regulation constraint for automated construction quality compliance checking, Automation in Construction, Volume 28, Pages 58-70.
[48] C. Zhang, J. Beetz, M. Weise (2015). Interoperable validation for IFC building models using open standards, ITcon Vol. 20, Special issue ECPPM 2014 - 10th European Conference on Product and Process Modelling, pg. 24-39.
[49] S. Zhang, F. Boukamp and J. Teizer, (2015). Ontology-based semantic modeling of construction safety knowledge: Towards automated safety planning for job hazard analysis (JHA). Automation in Construction, 52, pp.29-41.

Appendix

A. Resources

The defined vocabularies along with transformation rules and related source code are published on https://github.com/BenzclyZhang/BimSPARQL.

B. SPIN rules for implementing functions for case in section 6

```
SELECT (?a/?wallArea AS ?Ap)
WHERE{
?arg1 pdt:hasGrossWallArea ?wallArea .
?arg1 ibc:hasProtectedOpeningArea ?a
}
```

Listing 10: Query to implement the function *ibc:hasAp* referenced in Listing 9

```
SELECT (?a/?wallArea AS ?Au)
WHERE{
?arg1 pdt:hasGrossWallArea ?wallArea .
?arg1 ibc:hasUnProtectedOpeningArea ?a
}
```

Listing 11: Query to implement the function *ibc:hasAu* referenced in Listing 9

```
SELECT (MIN(?d) AS ?distance)
WHERE{
?line a ifc:IfcAnnotation .
?line ifc:name ?name .
?name expr:hasString "Lot Line" .
(?arg1 ?line) spt:distanceXY ?d .
}GROUP By ?arg1
```

Listing 12: Query to implement the function *ibc:hasFireSeparationDistance* referenced in Listing 9

```
SELECT ?ap
WHERE {
?b1 ibc:minFSDistance ?min .
?b1 ibc:maxFSDistance ?max .
?b1 ibc:openingProtection ?arg2 .
?b1 ibc:sprinklerProtection ?arg3 .
?b1 ibc:allowableArea ?ap .
FILTER ((qudtspin:convert(?arg1, unit:Meter,
  unit:Foot) >= xsd:double(?min)) && (
  qudtspin:convert(?arg1, unit:Meter, unit:
  Foot) < xsd:double(?max))) .
}
```

Listing 13: Query to implement the function *ibc:allowableArea_T705-8* referenced in Listing 9

```
SELECT (SUM(?windowArea) AS ?area)
WHERE{
?window schm:isPlacedIn ?arg1 .
?window pset:fireRating "OH-45" .
?window pdt:hasWindowArea ?windowArea .
} GROUP BY ?arg1
```

Listing 14: Query to implement the function *ibc:hasProtectedOpeningArea* referenced in Listing 10

```
SELECT (SUM(?windowArea) AS ?area)
WHERE{
?window schm:isPlacedIn ?arg1 .
FILTER NOT EXISTS{
?window pset:fireRating "OH-45" .
}
?window pdt:hasWindowArea ?windowArea .
} GROUP BY ?arg1
```

Listing 15: Query to implement the function *ibc:hasProtectedOpeningArea* referenced in Listing 11