Aalto University

OPEN ACCESS

This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.

Bonduel, Mathias; Oraskari, Jyrki; Pauwels, Pieter; Vergauwen, Maarten; Klein, Ralf

The IFC to Linked Building Data Converter - Current Status

Published in:
CEUR Workshop Proceedings

Published: 01/01/2018

Document Version
Publisher's PDF, also known as Version of record

Please cite the original version:
Bonduel, M., Oraskari, J., Pauwels, P., Vergauwen, M., & Klein, R. (2018). The IFC to Linked Building Data Converter - Current Status. *CEUR Workshop Proceedings*, *2159*, 34-43. http://ceur-ws.org/Vol-2159/04paper.pdf

This material is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of the repository collections is not permitted, except that material may be duplicated by you for your research use or educational purposes in electronic or print form. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone who is not an authorised user.

The IFC to Linked Building Data Converter - Current Status*

Mathias Bonduel[1] [0000-0002-3313-924X],
Jyrki Oraskari, Pieter Pauwels[3], Maarten Vergauwen[1], and Ralf Klein[1]

[1] KU Leuven, Dept. of Civil Engineering, Technology Cluster Construction, Ghent, Belgium
mathias.bonduel@kuleuven.be
[2] Aalto University, Dept. of Computer Science, Espoo, Finland
Jyrki.Oraskari@aalto.fi
[3] Ghent University, Dept. of Architecture and Urban Planning, Ghent, Belgium

Abstract. Several methods for creating building-related Linked Data graphs exist. This paper focuses on the conversion of IFC Building Information Models (BIM) to RDF Abox graphs using the emerging W3C Linked Building Data (LBD) modular ontologies: BOT (building topology), PRODUCT (classification of building elements) and PROPS (building-related properties). The existing IFC-to-RDF converter tool, converts IFC into ifcOWL-based Abox graphs which are rather complex and difficult to implement in software applications. The IFC-to-LBD converter presented, is necessary to transform IFC building models into RDF Abox graphs structured according the new LBD ontologies. An output graph contains the relevant information of the IFC building model related to building topology, building elements classification and building-related properties. Additionally, the graph structure becomes more concise (minimum 83 % less triples) and significantly easier to query compared to the output of the ifcOWL-based IFC-to-RDF converter.

Keywords: Linked Data, IFC, converter, AECO, BIM, Linked Building Data, BOT, PROPS, PRODUCT

1 Introduction

Linked Data for the Architecture, Engineering, Construction and Operation (AECO) domain is a research topic gaining momentum during the last decade. Besides numerous scientific publications and dedicated conferences, a Linked Building Data Community Group within the W3C (W3C LBD CG) and a Linked Data Working Group (LDWG) as part of the Technical Room of buildingSMART were founded. Linked Data graphs can be created from scratch, or by converting existing structured data available in other formats than RDF. The presented research focuses on the conversion of existing Building Information Models (BIM) in IFC format to modular Linked Data Abox graphs as proposed by the W3C LBD CG [8].

* This research was funded by respectively the Research Foundation Flanders (FWO) by means of a personal Strategic Basic research grant, and the Aalto-yliopiston tekniikan tukisäätiö

1.1 BIM and IFC

Building Information Modeling or BIM is an established concept within the AECO domain, regarding the representation and exchange of structured building information. BIM is implemented in object-oriented database systems with a strong focus on 3D modeling. Different software environments apply the BIM concept and a wide variety of both open-source and commercial software applications exist such as Autodesk Revit, Graphisoft ArchiCAD, FreeCAD (with plugins) and Tekla Structures. Each of these software packages uses its own proprietary data schema to represent the structure of a building, the elements it consists of and their properties. If a BIM workflow is used to exchange building information between project stakeholders, a neutral format to exchange the models is necessary. The Industry Foundation Classes (IFC), proposed by buildingSMART International, is a neutral and open ISO standard. The IFC standard consists of an IFC schema (in both EXPRESS and XSD) and an IFC file format (in IFC-SPFF and ifcXML) for BIM data [4]. BIM software developers can implement an exporter and/or importer to convert respectively their native BIM format to the neutral IFC format, or the other way around. Many national BIM standards strongly recommend or even impose the use of IFC to improve the exchange of building information between project stakeholders.

Besides the clear advantages regarding BIM software interoperability, IFC also has some limitations as indicated in [4]. First of all, the EXPRESS and XSD languages lacks methods for defining formal semantics, making it difficult to apply generic reasoning and querying methods on IFC building models. Secondly, developers can propose extensions for the IFC schema to buildingSMART, but the technology does not allow to extend the IFC schema on-the-fly in an user-friendly way. Thirdly, fine grained linking of building information stored in an IFC file to related data on the web (e.g. regulations of local authorities, geographic information, general knowledge, building product information, etc.) is not possible. Finally, the IFC schema is large - the most recent IFC schema, IFC4 Add2, contains about 1200 classes - and rather complex, making it a challenge to implement correctly in software.

1.2 The ifcOWL Ontology and IFC-to-RDF Converter

To overcome the first three limitations imposed by the usage of the IFC standard, researchers investigated the potential of Semantic Web Technologies such as web ontologies, the SPARQL query language and reasoning engines. Before developing tools to convert individual BIM models to RDF Abox graphs, a shared data structure or ontology (Tbox) had to be decided upon by the research community [1]. Over the years, several ontologies for the IFC schema were presented, resulting in a recommended ifcOWL ontology endorsed by buildingSMART [4]. The ifcOWL schema was designed to be as equivalent as possible to the original IFC EXPRESS schema, with the idea of backwards compatibility in mind. Following the development of the ifcOWL ontology, an IFC-to-RDF converter tool was developed [4]. The application converts an IFC building model to an RDF Abox graph structured according to ifcOWL, i.e. sets of RDF triples that contain assertions between individual RDF resources.

[4] https://github.com/jyrkioraskari/IFCtoRDF-Desktop

1.3 BOT-PROPS-PRODUCT Ontologies

The ifcOWL ontology and the IFC-to-RDF converter made it possible to apply Semantic Web technologies such as standardized querying (SPARQL) and reasoning methods, on IFC building models. Having IFC files available as RDF graphs also facilitates fine grained linking to other RDF datasets. The last limitation of IFC mentioned in Section 1.1 however, remains unsolved. The ifcOWL schema was developed as a Linked Data counterpart of the original IFC EXPRESS schema, resulting in one monolithic and complex ifcOWL ontology (Tbox). Converted IFC building models (Abox) are structured according to ifcOWL, so they too are large and complex. As a result, query writing is negatively influenced, making it hard to use the building data in Linked Data applications.

Alternatives, such as IfcWoD (an extension of ifcOWL)[2] and simpleBIM (a tool to simplify ifcOWL Abox graphs)[3], were proposed to overcome the above limitation of ifcOWL. Although these alternatives made query writing easier and even proved to reduce query execution times, a more generic approach using modular ontologies and RDF graphs is seen as a necessity for real industrial applications [7]. The Building Typology Ontology (BOT)[6] was developed within the W3C LBD CG as a central and modular ontology for the AEC industry, followed by the emergence of subgroups developing a whole range of other modular ontologies for building products, building-related properties, geometry, etc.

BOT is developed to allow defining a building's topology, by using dedicated BOT classes (bot:Site, bot:Building, bot:Storey, bot:Space and bot:Element) and specific BOT relations between them (e.g. bot:containsElement). The PRODUCT ontology is designed to classify individual building objects, e.g. as a p4bldg:Wall, while the PROPS ontology is used to assign properties to building-related elements. As the PROPS ontology is still in conceptual design phase, no ontology (Tbox) is available yet. The current early proposal for the PROPS ontology structure includes three levels of complexity [5], as demonstrated in Listing 1. Grouping (L2 or L3) and versioning of properties (L3) is only possible when using the more complex structures.

Listing 1: Currently proposed levels of complexity in the PROPS ontology, demonstrated here for the case when the actual property value is a literal

```
#PROPS level of complexity 1 (L1)
inst:slab_A props:phaseCreated_simple "New Construction" .

#PROPS level of complexity 2 (L2)
inst:slab_B props:phaseCreated inst:property_B .
inst:property_B schema:value "New Construction" .

#PROPS level of complexity 3 (L3)
inst:slab_C        props:phaseCreated inst:property_C .
inst:property_C seas:evaluation inst:state_C .
inst:state_C a opm:CurrentState ;
        schema:value "New Construction" ;
        prov:generatedAtTime "2018-01-03T13:35:23Z" .
```

[5] https://github.com/w3c-lbd-cg/lbd/blob/gh-pages/presentations/props/presentation_LBDcall_20180312_final.pdf

1.4 IFCtoLBD Converter

Ontologies (Tbox) in the Semantic Web domain are mainly used to allow instantiating them into RDF data instances (Abox). As stated before, such RDF graphs can be made from scratch or by converting other data, such as conventional BIMs, into RDF. It is for example possible to develop a dedicated plugin for existing BIM software environments to convert proprietary BIM data in RDF graphs, as demonstrated in [5] for the BOT ontology and Revit BIM authoring tool. On the other hand, correctly exported IFC files from different proprietary BIM systems can also be converted to RDF graphs. The IFC-to-RDF converter translates IFC to RDF Abox graphs according to the ifcOWL ontology. In this paper, a similar approach is used to convert IFC to modular Linked Building Data (LBD) graphs, aiming initially at the BOT, PROPS and PRODUCT ontologies. Relevant information from IFC building models is extracted and transformed into effective and simple Abox RDF graphs suited for usage in Linked Data applications.
All URI prefixes mentioned in this paper are assembled in Listing 2.

Listing 2: Used URI prefixes in this paper

```
@prefix inst: <https://example.com/> . # demo namespace for node instances
@prefix bot: <https://w3id.org/bot#> .
@prefix props: <https://w3id.org/props#> .
@prefix p4bldg: <https://w3id.org/product/BuildingElements#> .
@prefix schema: <http://schema.org/> .
@prefix seas: <https://w3id.org/seas/#> .
@prefix opm: <https://w3id.org/opm#> .
@prefix prov: <https://www.w3.org/TR/prov-o/#> .
@prefix express: <https://w3id.org/express#> .
# similar for other ifcOWL versions:
@prefix ifcowl: <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>.
```

2 The Current State of the IFCtoLBD converter

2.1 Implementation Details

The IFCtoLBD converter is an open-source project written in Java using the well known Jena framework, and is available on Github [6]. The proposed converter can handle IFC files from all actively used IFC schemas, from IFC2x3 TC1 till IFC4 Add2. An IFC building model can be converted to one RDF file combining all selected LBD Abox modules (BOT, PROPS and/or PRODUCT) or to multiple separate RDF files per selected LBD module (Abox). Using such Abox modules, which can be used independently from each other, it becomes easier to understand the structure of the combined graph.
If the user selects the PROPS module, he or she can choose one of the three PROPS levels of complexity (see Listing 1). If PROPS L2 or L3 is selected, the user can choose to define the in between property instance nodes (L2 and L3) and/or the state instance nodes (L3) as blank nodes or instances with a stable URI. If the above nodes do not have a stable URI, they are not unique outside the database or file they reside in, making it impossible to connect them to other resources outside their original environment. On

[6] https://github.com/jyrkioraskari/IFCtoLBD

the other hand, using blank nodes can decrease the file size of the exported RDF and increase the human readability of the Turtle file. As the PROPS ontology is still in a conceptual phase, specific datatype and object properties are not defined there. For the moment, the converter creates these properties on-the-fly based on the IFC property name. If the property value is a literal and the user selects PROPS L1, the suffix `_simple` is added at the end of the URI of the generated predicate. In this way, there is a distinction between e.g. props:phaseCreated in PROPS L2 and L3 (an owl:ObjectProperty) and props:phaseCreated_simple in PROPS L1 (an owl:DatatypeProperty).

2.2 The Conversion Process

The IFCtoLBD converter first uses the existing IFC-to-RDF converter internally to transform the original IFC file to a temporary ifcOWL Abox graph (Figure 1). That graph is converted incrementally to one or multiple LBD Abox graphs, according to the settings of the user. By first converting the IFC file to an ifcOWL-based Abox graph, the overall conversion process becomes slightly slower and there is an extra dependency on the ifcOWL ontology. However, by implementing this intermediate step, using the well-tested IFC-to-RDF converter and the reliable ifcOWL ontology, it becomes possible for the converter to query RDF patterns directly. When converting very large IFC building models, a connection with a RDF database system is necessary, but implementing this requires only minor changes to the current application.

When the intermediate ifcOWL Abox graph is available, the converter iteratively creates new LBD nodes starting from the relevant ifcOWL nodes, followed by corresponding LBD edges between these new nodes (Figure 1).The converter follows the controlled graph traversal using path templates: it starts from the IfcSite instance node and traverses the ifcOWL-based graph towards its IfcBuilding instance nodes, while simultaneously looking for the IFC properties of the IfcSite instance. The search is not started from the IfcProject instance, since there is no corresponding class in the BOT ontology. In a successive phase, it queries all instances of IfcBuildingStorey of the previously found IfcBuilding instances, together with the properties of these same IfcBuilding instances. This approach is executed until all IFC building elements and their properties are found.

Graph nodes By using the corresponding classes between the ifcOWL and the LBD modular ontologies (BOT, PRODUCT, PROPS) displayed in Table 1, the newly created instance resources get the right LBD OWL classes assigned. If PROPS L2 or L3 is selected, new instance nodes for properties are created. Unique property set (pset) instances do exist in the ifcOWL-based Abox graphs and as such, they can be converted directly into new instances of props:PsetGroup. In the case of PROPS L3, also new instance nodes for property states are made for each property. The URIs of the new building element instance nodes are constructed as `<base URL><human readable label>_<GUID>`. The human readable label is derived from the PRODUCT class or - if it is not a bot:Element - from the corresponding BOT class. The GUID is always the IFC GUID of the corresponding IFC object in hexadecimal notation.

Instantiated property nodes (PROPS L2 and L3) have URIs in the form of `<base URL><propertyName>_<GUID>`, where the GUID is identical to the one of the connected BOT resource. In the case of property set instance nodes (PROPS L2 and L3), the URIs are constructed as `<base URL>psetGroup_<psetName>`. Similarly, the

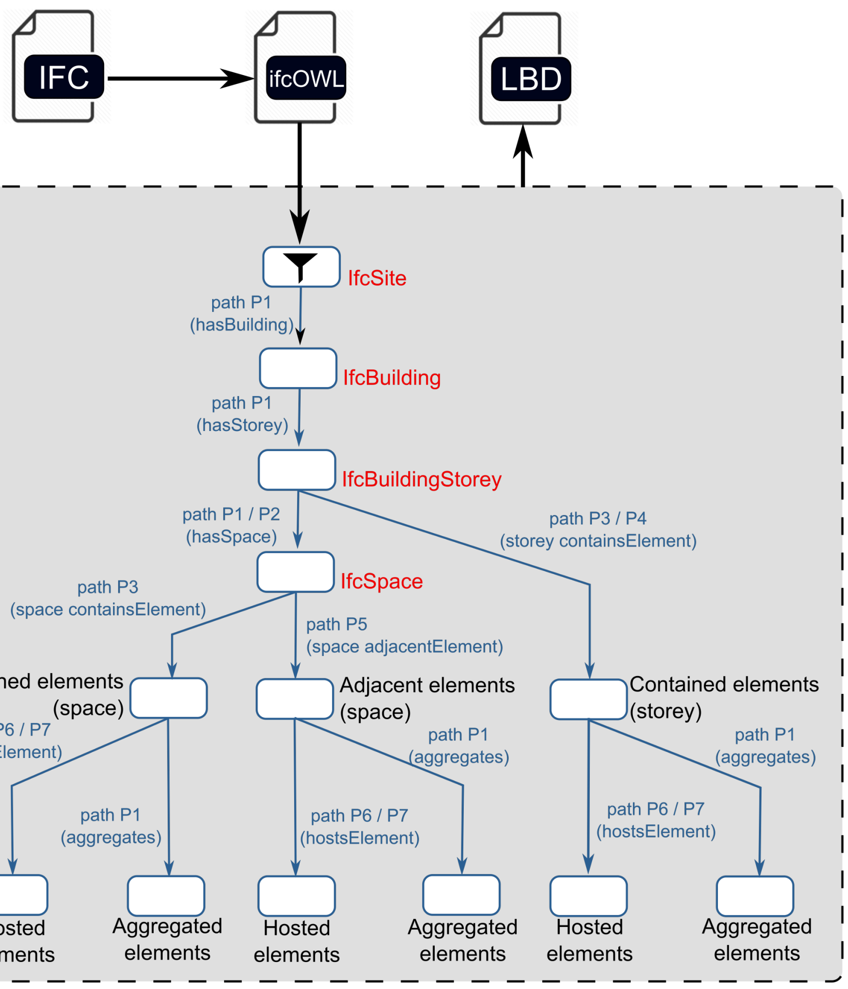

Fig. 1: Implemented IFCtoLBD conversion process regarding BOT

URIs of attribute group instances follow the `<base URL>attributeGroup` structure. Finally, the state instances (PROPS L3) have URIs in the form of `<base URL>state_<propertyName>_<GUID>_<timestamp>`. An extra filter was implemented in the code to exclude redundant properties, i.e. properties that contain no value, an empty string value or a string value that is identical to the property name.

Graph edges At the moment of writing, the converter simplifies seven different ifcOWL Abox triple patterns into seven corresponding direct BOT relation (see Table 2), depending on the type of start node. The bot:adjacentZone relations are not yet implemented as no clear property path between such IFC elements was found. The pattern P1 for finding the IfcBuilding from the respective IfcSite is given in Listing 3 using the SPARQL language.

Table 1: Corresponding ifcOWL and LBD classes

ifcOWL classes	**BOT classes**
IfcSite	Site
IfcBuilding	Building
IfcBuildingStorey	Storey
IfcSpace	Space
IfcElement / IfcFurnitureType	Element
ifcOWL classes categorized as bot:Element	**PRODUCT classes**
IfcWall / IfcWallStandardCase	Wall
IfcDoor	Door
...	...
ifcOWL classes	**PROPS classes (L2 and L3)**
IfcPropertySet	PsetGroup

Table 2: ifcOWL property paths and corresponding LBD relation or property path

ifcOWL property paths	**ifcOWL class of start node**	**Corresponding LBD property or property path**
P1	IfcSite	bot:hasBuilding
P1	IfcBuilding	bot:hasStorey
P1 / P2	IfcBuildingStorey	bot:hasSpace
P3 / P4	IfcBuildingStorey	bot:containsElement
P3	IfcSpace	bot:containsElement
P5	IfcSpace	bot:adjacentElement
P6 / P7	IfcWall / IfcRoof / ...	bot:hostsElement
P1	IfcStair / ...	bot:aggregates
P8a, P8b and P8c (properties part of psets)	(any building element found above)	PROPS L1 / L2 / L3 (see Listing 1)
P9 + specific property paths (IFC attribute properties)	(any building element found above)	PROPS L1 / L2 / L3 (see Listing 1)

Listing 3: Property path P1 query to find the IfcBuilding instances of a known IfcSite

```
SELECT ?building WHERE {
  inst:IfcSite_38274 ^ifcowl:relatingObject_IfcRelDecomposes ?relJoin .
  ?relJoin ifcowl:relatedObjects_IfcRelDecomposes ?building . }
```

Regarding the building-related properties, a distinction between IFC properties stored in property sets and IFC attribute properties is made. Attribute properties do not reside in property sets but are directly attached to its building element in IFC. In the first case, three triple patterns from the ifcOWL Abox graph are needed (Listing 4): one for finding the properties in property sets of a building element (P8a), one for the property name (P8b) and one for the property value of such a property (P8c). In the second case, the general rule (path P9) is to list all attributes that have a String, Integer, Double, or Boolean data type. This approach only retrieves tag_IfcElement (the BIM Autoring Tool

ID or batid), globalId_IfcRoot (the IFC GUID), name_IfcRoot, objectType_IfcObject, refElevation_Ifcsite and longName_IfcSpatialStructureElement, while for example IfcMaterial, IfcPresentationLayer, country_IfcPostalAddress, etc. are not covered by this path. Specific property paths will be necessary to convert these other attribute properties.

Listing 4: Property paths P8a, P8b and P8c for finding property sets of a certain building element and the properties (values and property names) they contain

```
SELECT ?psets ?propNames ?propValues WHERE {
  # path P8a: properties of property sets of a know element
  inst:IfcSlab_37864 ^ifcowl:relatedObjects_IfcRelDefines ?relJoin .
  ?relJoin ifcowl:relatingPropertyDefinition_IfcRelDefinesByProperties
    ?psets .
  ?psets ifcowl:hasProperties_IfcPropertySet ?propSingleVal .

  # path P8b: property names
  ?propSingleVal ifcowl:name_IfcProperty ?propNameJoin .
  ?propNameJoin express:hasString ?propNames .

  # path P8c: property values
  ?propSingleVal ifcowl:nominalValue_IfcPropertySingleValue ?propValJoin .
  ?propValJoin express:hasString | express:hasBoolean | express:hasDouble
        | express:hasInteger | express:hasLogical ?propValues . }
```

2.3 Duplex House IFC case study

The current IFCtoLBD converter was tested on the IFC file of the Duplex apartment benchmark model (`Duplex_A_20110907.ifc`), published by NIBS [7]. The initial IFC building model in SPF format (IFC2x3 TC1) of 2.3 MB is converted into an RDF graph using the existing IFC-to-RDF converter (ifcOWL) and compared with the results of the new IFCtoLBD converter (BOT-PROPS-PRODUCT). The conversion with the IFCtoLBD converter was executed three times, with different settings for the PROPS level of complexity. In the case of PROPS L2 and L3, all property and state instance nodes are converted into a stable URI instead of using blank nodes.

Reduced graph size The results of the conversion to LBD Abox graphs are presented in Table 3. The ifcOWL Abox Turtle file counts 227,143 triples (17428.0 KB file size). First of all, there is a clear decrease in number of triples, because all geometry-related triples from the ifcOWL-RDF are excluded: [3] noted a reduction in triples of about 38 % only by removing the IFC geometry-related triples. The number of property steps between building-related elements (e.g. spaces and storeys) decreased from two steps to one by applying the BOT ontology properties. Building-related properties are handled now as is typically done in a Semantic Web environment, leaving out redundant intermediate nodes. In this way, the number of steps/relations between a building element and a property value is reduced from five to one, two or three steps depending on the selected PROPS level of complexity. The element's property name is now modeled as a datatype/object property (L1) or object property (L2 and L3), instead of a string literal.

[7] https://www.nibs.org/?page=bsa_commonbimfiles

In total, a decrease of 83% in the overall number of triples (80% in file size) is demonstrated when all three modules are exported to one file with PROPS L3 selected, compared to the ifcOWL Abox graph of the same IFC building model. When other PROPS levels are selected, or other Abox modules are left out, the file size and number of triples can even be further reduced. The use of Abox modules make it easier to share and study specific parts of the converted LBD graph. Regarding the PROPS module, users can select the most appropriate level of complexity they need for their project.

Table 3: IFCtoLBD converter output results per module

BOT	PRODUCT	PROPS		TOTAL
45.5 KB (729 triples)	18.8 KB (263 triples)	PROPS L1	397.7 KB (6099 triples)	441.2 KB (7091 triples)
		PROPS L2	1665.3 KB (19831 triples)	1708.4 KB (20823 triples)
		PROPS L3	3399.0 KB (38128 triples)	3442.4 KB (39120 triples)

Simplified queries As the original ifcOWL Abox graph reduced significantly in number of triples by applying the LBD ontologies, the resulting Abox graph structure becomes simpler while the information in the graph is equivalent to the one in the ifcOWL-based Abox graph. As a logical consequence, query writing also becomes more clear and shorter, as shown in Listing 5 for BOT (find the bot:Building instances of a specific bot:Site) and Listing 6 for PROPS (find the loadBearing property value of a specific element). These queries can be compared with the earlier presented queries for the ifcOWL Abox graph, respectively in Listing 3 and Listing 4. In the first case, the number of intermediate steps between both objects drops from two to one. In the second case, the amount of intermediate steps between a property value and the object is reduced from five to one (PROPS L1).

Listing 5: SPARQL query on the output of the IFCtoLBD converter (all instances of bot:Building of a known site)

```
SELECT ?site ?buildings WHERE {
  BIND(inst:site_7b7032cc-b822-417b-9aea-642906a29bd7 AS ?site)
  ?site bot:hasBuilding ?buildings . }
```

Listing 6: SPARQL query on the output of the IFCtoLBD converter (the loadBearing property value of a certain slab; the property is of PROPS L1)

```
SELECT ?propValue WHERE {
  BIND(inst:slab_982f59b0-f2e1-485f-8ce1-c9f6117b70a9 AS ?slab)
  ?slab props:SlabCommon_loadBearing_simple ?propValue . }
```

3 Conclusion

The IFCtoLBD converter facilitates the conversion of IFC building models to modular LBD Abox graphs structured according the BOT, PROPS and PRODUCT ontologies designed within the W3C LBD CG. The resulting Abox graphs are more user-friendly than previous ifcOWL Abox graphs, because the graph structure is simplified and closer to what would be expected in a Semantic Web environment. As a consequence, the output graphs are smaller in size and SPARQL queries become significantly shorter and easier to understand, as demonstrated with the Duplex house use case. Users can test the three different PROPS levels as currently proposed in the W3C LBD CG, and provide crucial feedback to the PROPS ontology subgroup. By using PROPS L3, it becomes possible to do - with little extra implementation effort - versioning of properties in IFC building models.

Future research includes more elaborate use cases for the converter and a detailed analysis of the improved query execution times compared to ifcOWL Abox graphs. The conversion of IFC unit information using the Custom Datatype properties (CDT) ontology and a feasibility study regarding ifcOWL property paths corresponding to bot:adjacentZone relations and bot:adjacentElement relations between bot:Storey and bot:Element instances will be addressed. The conversion of all relevant IFC attribute properties will also be documented and implemented in future versions of the converter.

References

1. Beetz, J., Van Leeuwen, J., De Vries, B.: IfcOWL: A case of transforming EXPRESS schemas into ontologies. Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM **23**(1), 89–101 (2009). https://doi.org/10.1017/S0890060409000122
2. Mendes de Farias, T., Roxin, A.M., Nicolle, C.: IfcWoD, Semantically Adapting IFC Model Relations into OWL Properties. In: Proc. of the 32nd CIB W78 Conference 2015, 27th-29th October 2015, Eindhoven, The Netherlands. pp. 175–185 (2015)
3. Pauwels, P., Roxin, A.: SimpleBIM : From full ifcOWL graphs to simplified building graphs. In: Christodoulou, S., Scherer, R. (eds.) eWork and eBusiness in Architecture, Engineering and Construction (ECPPM). pp. 11–18. CRC Press, Limassol, Cyprus (2016)
4. Pauwels, P., Terkaj, W.: EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. Automation in Construction **63**, 100–133 (2016). https://doi.org/10.1016/j.autcon.2015.12.003
5. Rasmussen, M.H., Hviid, C.A., Karlshøj, J.: Web-based topology queries on a BIM model (2017), presented at 5th LDAC workshop. Dijon, France
6. Rasmussen, M.H., Pauwels, P., Hvidd, C.A., Karlshøj, J.: Proposing a Central AEC Ontology That Allows for Domain Specific Extensions. In: LC3 2017: Proceedings of the Joint Conference on Computing in Construction. pp. 237–244. Heraklion, Greece (2017)
7. Terkaj, W., Pauwels, P.: A Method to generate a Modular ifcOWL Ontology. In: Proceedings of the 8th Workshop Formal Ontologies Meet Industry, Joint Ontology Workshops 2017, CEUR Workshop Proceedings. vol. 2050. Bolzano, Italy (2017)
8. Terkaj, W., Schneider, G.F., Pauwels, P.: Reusing Domain Ontologies in Linked Building Data : the Case of Building Automation and Control. In: Proceedings of the 8th Workshop Formal Ontologies Meet Industry, Joint Ontology Workshops 2017, CEUR Workshop Proceedings. vol. 2050. Bolzano, Italy (2017)