Enhancing the ifcOWL ontology with an alternative representation for geometric data

Pieter Pauwels 1 Thomas Krijnen 2 Walter Terkaj 3 Jakob Beetz 2

Abstract
Over the past few years, several suggestions have been made of how to convert an EXPRESS schema into an OWL ontology. The conversion from EXPRESS to OWL is of particular use to architectural design and construction industry, because one of the key data models in this domain, namely the Industry Foundation Classes (IFC), is represented using the EXPRESS information modelling language. These conversion efforts have by now resulted in a recommended ifcOWL ontology that stays semantically close to the EXPRESS schema. Two major improvements could be made in addition to this ifcOWL basis. First, the ontology could be split into diverse modules, making it easier to use subsets of the entire ontology. Second, geometric aggregated data (e.g. lists of coordinates) could be serialised into alternative, less complex semantic structures. The purpose of both improvements is to make ifcOWL data smaller in size and complexity. In this article, we focus entirely on the second topic, namely the optimisation of geometric data in the semantic representation. We outline and discuss the diverse available options in optimising the data representations used. We quantify the impact on ifcOWL ontology and instance model size (both IFC2X3.2-TC1.owl and IFC4_ADD1owl). We conclude with an explicit recommendation and give an indication of how this recommendation might be implemented in combination with the already available ifcOWL ontology.

Keywords: BIM, IFC, OWL, linked data, geometry, data exchange

1 Introduction
1.1 Interoperable data exchange for Building Information Models

Building information modelling (BIM) is one of the most notable efforts in years regarding information management in construction industry (Eastman et al., 2008). BIM environments allow to semantically describe any kind of information about the building in a common information environment. The Industry Foundation Classes (IFC) standard (International Organization for Standardization, 2013), developed and maintained by the buildingSMART organisation (BuildingSMART International, 2015), aims at providing a central “conceptual data schema and an exchange file format for BIM data” (Liebich et al., 2013). Using the IFC data model and instance serialisation formats, BIM data can be exchanged between heterogeneous software applications covering a wide range of use cases including 4D planning, 5D cost calculation and structural analysis. In many common business scenarios, such files consist of partial domain models from different stakeholders that are exchanged frequently in iterative design and planning processes. For building projects in the later planning stages with high amounts of detail (reinforcement, ductwork etc), for large buildings or buildings with complex geometries,
instance files commonly consist of hundreds of thousands of objects and the resulting models result in large files that are resource-intensive to process.

The IFC data model is represented as a schema in the expressive EXPRESS data specification language defined in ISO 10303-11:2004 that “consists of language elements which allow an unambiguous data definition and specification of constraints on the data defined and by which aspects of product data can be specified” (International Organization for Standardization, 2004). Currently, the most commonly used IFC schemas are IFC2X3 (IFC2X3_TC1.exp) and IFC4 (IFC4_ADD1.exp). IFC2X3 is important because it has been used for more than ten years in industry. Hence, numerous sample IFC2X3 implementations in widely adopted software tools are available and in use in practice. A good public real-world data set with IFC2X3 files is available in (van Schaijk, 2016) (Dataset Schependomlaan). IFC4 is important because it is the last version of IFC and thus supersedes IFC2X3. As IFC4 is not yet widely implemented and/or certified in commercial software tools yet, no public IFC4 real-world sample files are available, besides the ones that are available as part of the IfcDoc tool (BuildingSMART International, 2016b). The IfcDoc sample files however include many of the less commonly used IFC data types, including lists of lists, binary representations, tesselated geometry, and so forth. In this article, we will use these sample IFC4 files from IfcDoc as well as the IFC2X3 data set in (van Schaijk, 2016), leading to a set of sample files that is representative for IFC2X3 as well as IFC4, and that is representative for real-world IFC files as well as semantically less common data. All test files are made available at Pauwels et al. (2016).

1.2 Background: the ifcOWL ontology

In 2013, the latest version of the IFC schema (IFC4) was published into an ISO standard (ISO 16739) not only as an EXPRESS schema, but also as an XSD schema. The main objective was to make the data model more easily available for flexible usage in XML-based environments. Due to the verbosity of the XML document format and the performance limitations of the commonly available XML processing tools and programming libraries however, this representation has been deemed inappropriate for most practical use cases and was never fully embraced by the industry. In order to benefit from the built-in capabilities to modularize and distribute models across file boundaries and network structures, to harness reasoning and standardized query capabilities and to easily integrate further vocabularies and data sets with BIM models, Semantic Web and Linked Data technologies were coming into the focus of numerous research efforts.

It is not the purpose of this article to list all the possibilities that are made available from the mere usage of linked data or semantic web technologies. Yet, Pauwels et al. (2017) (Pauwels et al., 2017) gives a broad overview of these possibilities, while referring to plenty of example implementations worldwide. This
article lists examples in three categories: (1) interoperability, (2) linking data across domains, and (3) logical inference and proofs. The usage of semantic web technologies appears to enable use cases mostly in the latter two categories. When a lot of links are made across domains (with IFC being just one of those domains), externally managed product manufacturer data is more tightly connected to building models; improved building performance analysis can be targeted; regulation compliance-checking is possible; a link with geographical and infrastructure data is possible; and so forth. These use cases make considerable use of the ease of linking data with these technologies, as well as the out-of-the-box query functionality. In the case of logical inference and proofs, regulation compliance-checking is an often mentioned use case, in which the logical model of the building (e.g. an ifcOWL representation) can naturally be combined with a rule set that represents the logical model of the building regulation. Checking the compliance with building regulations is then just a matter of starting a logic-based reasoning engine and querying for the compliance-checking result.

Using OWL as a schema modelling language, an ontology for the IFC data model was proposed by numerous authors (Schevers and Drogemuller, 2005; Agostinho et al., 2007; Beetz et al., 2009; Krima et al., 2009; Barbau et al., 2012; Pauwels and Van Deursen, 2012; Pauwels and Terkaj, 2016): ifcOWL. These different efforts are currently converging into an agreed common standard. The proposed conversion effort hereby specifically aims to keep the resulting OWL ontology as close as possible to the original EXPRESS schema of IFC. This conversion approach is documented in full detail in Pauwels and Terkaj (2016), including an extensive literature review and comparison of approaches. The proposed ifcOWL ontology is now picked up within buildingSMART International, where it might eventually become a part of the ISO 16739 standard, similar to the way in which the XSD schema became part of this standard. As a result, the IFC data model would be available in EXPRESS, XSD, and OWL (see Fig. 1), allowing the representation and usage of building data in STEP Physical File Format (SPFF), eXtensible Markup Language (XML), and the Resource Description Framework (RDF).

Figure 1: The IFC data model is available in EXPRESS, XSD, and OWL, allowing to capture and use building data using three different technologies.
The latest edition of the ifcOWL ontology can be found in the buildingSMART International web pages (BuildingSMART International, 2016a). The ifcOWL directly imports the EXPRESS ontology (Terkaj and Pauwels, 2015) and indirectly imports the LIST ontology (Pauwels and Terkaj, 2015) that was modelled after Drummond et al. (2006). The key features of the conversion pattern are summarized in Table 1, while referring to the prefixes defined for the IFC4_ADD1 ontology in Fragment 1. Specific further details, in particular regarding the conversion of aggregation data types (rows 3, 4, and 10 of Table 1), are provided in the following sections.

<table>
<thead>
<tr>
<th>Prefix</th>
<th>URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>ifc:</td>
<td><a href="http://ifcowl.openbimstandards.org/IFC4_ADD1#">http://ifcowl.openbimstandards.org/IFC4_ADD1#</a></td>
</tr>
<tr>
<td>xsd:</td>
<td><a href="http://www.w3.org/2001/XMLSchema#">http://www.w3.org/2001/XMLSchema#</a></td>
</tr>
<tr>
<td>owl:</td>
<td><a href="http://www.w3.org/2002/07/owl#">http://www.w3.org/2002/07/owl#</a></td>
</tr>
<tr>
<td>rdfs:</td>
<td><a href="http://www.w3.org/2000/01/rdf-schema#">http://www.w3.org/2000/01/rdf-schema#</a></td>
</tr>
<tr>
<td>rdf:</td>
<td><a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#">http://www.w3.org/1999/02/22-rdf-syntax-ns#</a></td>
</tr>
<tr>
<td>list:</td>
<td><a href="https://w3id.org/list#">https://w3id.org/list#</a></td>
</tr>
<tr>
<td>expr:</td>
<td><a href="https://w3id.org/express#">https://w3id.org/express#</a></td>
</tr>
</tbody>
</table>

Fragment 1: Prefixes of the ontologies that are relevant for the reference ifcOWL ontology.

1.3 The modular structure of IFC

The current ifcOWL ontology (BuildingSMART International, 2016a) is a direct mapping of the IFC EXPRESS schema. The IFC4_ADD1 (IFC2X3_TC1) EXPRESS schema contains 768 (653) ENTITY data types, 206 (164) enumeration data types, 60 (46) select data types, 131 (117) defined data types, 46 (38) FUNCTION declarations, and 2 (2) RULE declarations. This results in an ifcOWL ontology for IFC4_ADD1 (IFC2X3_TC1) with 1313 (1093) classes, 1580 (1422) object properties, 5 (5) data properties, 13867 (11790) logical axioms, and 1158 (1018) individuals. The ontology is thus considerably big and complex to load and use. Furthermore, the ontology takes full advantage of OWL2 DL expressivity (SHIQ(D)), which can lead to a high number of assertions when handed to OWL reasoning engines. By consequence, when the ontology is referenced by an instance file in RDF, all 1313 classes, 1580 object properties, 13867 logical axioms and so forth are loaded. Moreover, it might be necessary to produce all available OWL2 DL assertions as well, leading to further overhead and delay in any software application.

Therefore, it would make sense to split the IFC ontology in separate modules, or separate smaller ontologies, so that end users and applications only need to load those ifcOWL modules that are actually going to be used. However, many of the entities and types in the IFC schema are tightly interconnected between the diverse sub-schemas. Hence, in order to make a useful modularisation, a full investigation of the schema needs to be made, and the relation between the different modules would need to be reconsidered to a significant level and detail. Such an investigation is out of scope for this research and paper. Nevertheless, we
<table>
<thead>
<tr>
<th>EXPRESS declaration</th>
<th>ifcOWL as in (BuildingSMART International, 2016a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple data type</td>
<td>owl:Class as defined in (Terkaj and Pauwels, 2015)</td>
</tr>
<tr>
<td>Defined data type</td>
<td>owl:Class</td>
</tr>
</tbody>
</table>
| Defined data type as an aggregation SET data type | - owl:Class  
- restriction on owl:ObjectProperty exp:hasSet to define the list size |
| Defined data type as an aggregation data type LIST (or ARRAY) | - OWL class  
- subclass of list:OWLList (Pauwels and Terkaj, 2015)  
- restrictions to define the list size (see Algorithms 1, 2, 3 in (Pauwels and Terkaj, 2016)) |
| Constructed SELECT data type | owl:Class and the select items are defined as its subclasses |
| Constructed ENUMERATION data type | owl:Class and the enumeration items are defined as its owl:NamedIndividual |
| Entity data type     | owl:Class                                      |
| Attribute of entity data type | - functional object property  
- explicit domains and ranges  
- owl:AllValuesFrom restriction  
- owl:qualifiedCardinality or owl:maxQualifiedCardinality restriction |
| Attribute of entity data type as a SET | - non-functional object property with specified domain and range  
- owl:AllValuesFrom restriction  
- owl:minQualifiedCardinality and/or owl:maxQualifiedCardinality restriction or owl:qualifiedCardinality restriction |
| Attribute of entity data type as a LIST (or ARRAY) | - functional object property with a subclass of list:OWLList as its range  
- owl:AllValuesFrom restriction on subclass of list:OWLList  
- restrictions to define the list size (see Algorithms 4, 5, 6 in (Pauwels and Terkaj, 2016)) |
| INVERSE attribute    | - object property  
- owl:inverseOf |
| DERIVE attribute     | N/A                                             |
| WHERE rule           | N/A                                             |
| FUNCTION             | N/A                                             |
| RULE                 | N/A                                             |

Table 1: Conversion pattern adopted to generate the reference ifcOWL ontology.

I wish to mention that the modularisation of ifcOWL is an important next step in restructuring the ontology so that it can be more efficiently used in a web context.
1.4 The representation of geometry in IFC

A large part of a typical IFC-SPF file is devoted to the representation of geometric structures, usually employing aggregation data types (e.g. ordered lists of points in Cartesian points, ordered lists of Cartesian points in polylines, and so forth). Therefore, if one aims at reducing the size and complexity of an RDF graph based on ifcOWL, two important options are available: (1) removing the geometry from the RDF representation or (2) improving the RDF representation of geometry so that it is smaller in size and complexity.

The first option (removing geometry) is a viable option, which was also proposed in Pauwels and Roxin (2016). Indeed, one can question to what extent a full semantic representation of geometry should still be included in a semantic web representation of a building model. The geometric representation typically streams directly towards an existing interpreter and corresponding 3D engine, so it makes more sense to maintain the syntax and semantics of those interpreters and 3D engines. Yet, as this ifcOWL ontology is aimed to serve as a reference standard in-sync with the EXPRESS schema of IFC, geometry needs to remain included.

This leaves the second option (improving the RDF representation of geometry), in which the geometry is at least stored as efficiently as possible, so that it can more easily flow into existing interpreters and 3D engines without needing a full semantic analysis. So, in order to obtain RDF graphs that are smaller in size and complexity, it is key to find a less complex representation for geometric information. As geometric information is commonly represented in aggregation data types (lists), it is key to find a less complex representation of lists in OWL and RDF, which was the subject of an earlier study (Pauwels et al., 2015). Therefore, it is important to investigate:

1. how aggregation data types are used for the representation of geometry in the IFC EXPRESS schema;
2. how these geometric aggregation data types can be efficiently expressed in OWL.

1.4.1 Ordered aggregations in EXPRESS

Four aggregation data types are available in EXPRESS: ARRAY, LIST, BAG, and SET. These aggregation data types each have a different semantic definition. The LIST and ARRAY data types represent ordered collections of elements, whereas the BAG and SET data types represent unordered collections (International Organization for Standardization, 2004, p.23). The BAG aggregation data type is not used in the IFC EXPRESS schema, and the SET aggregation data type can easily be captured in the RDF data model (Pauwels and Terkaj, 2016). Hence, we limit ourselves in
this article to the ordered \textbf{LIST} and \textbf{ARRAY} data types. In the case of geometry, the EXPRESS schema relies mainly on the \textbf{LIST} data type (see Sect. 2).

\subsection*{1.4.2 Ordered aggregations in OWL}

The representation of ordered aggregation data types is not explicitly addressed in the Web Ontology Language (OWL2 - Hitzler et al. (2012)), as they do not naturally fit into the graph structure of the RDF data model. In particular, the simplest representation based on the RDF terms \texttt{rdf:list} - \texttt{rdf:first} - \texttt{rdf:rest} (Brickley and Guha, 2014; Dean and Schreiber, 2004) often results in an ontology that is in the OWL Full profile (see Hitzler et al. (2012)), as the list can contain both classes and individuals. This implies that the resulting ontology cannot be used to exploit the inference and semantic search functionalities that are so desirable for the domain of architectural design and construction. Alternative representation options, in which a custom construct for lists is used (e.g. (Drummond et al., 2006)), typically result in complex and verbose graph structures.

As a result, the conversion of \textbf{LIST} and \textbf{ARRAY} data types from EXPRESS to OWL often results in representations of an undesirable form, making the resulting RDF graphs following an ifcOWL specification hard to use.

\subsection*{1.5 Paper aims and outline}

In this paper, we will investigate how the ifcOWL ontology can be enhanced in order to support a more efficient and scalable generation of RDF graphs representing BIM projects. In particular the attention is focused on geometric structures that are mainly represented by means of ordered aggregations in ifcOWL. Note that both unstructured faceted geometry as well as rich parametric geometry are possible in the IFC schema. Spagnuolo and Falcidieno (2009) advocate that conventional geometrical representations such as triangle meshes (unstructured faceted geometry) in the context of the semantic web do not constitute to explicit semantics. Rather, by modeling the appearance, the semantic concepts can only be understood by ‘viewing’ the element. They argue that in order to enrich such representation with meaning a viable approach is to decompose such elements into extracted features and meaningful labelled parts (parametric geometry). Although this paper looks to find a more efficient representation of geometry, it also aims at maintaining the geometric decomposition instead of reducing geometry to one simple triangulated mesh.

After analysing the prevalence of ordered aggregations in IFC (Sect. 2), we review the diverse options suggested in literature to cope with lists in OWL and also propose new options targeting the specific case of ifcOWL (Sect. 3). Each of these options is documented in detail, so that an appropriate understanding is achieved of the features, advantages and disadvantages of the diverse options.
We specifically highlight the relevance of the diverse options for ifcOWL and its usability concerns in the architectural design and construction industry. We will do this by outlining where and how ordered aggregation data types (LISTs and ARRAYs) are typically used in the EXPRESS schema of IFC. Based on these two overviews, we suggest how each ordered aggregation data type should ideally be made available in ifcOWL (Sect. 4).

2 Ordered aggregation data types in IFC

2.1 IFC schema statistics for ordered aggregation data types

There are two distinct cases of aggregation data type declarations in IFC:

1. A defined data type that is declared as an aggregation data type.
2. An aggregation data type as the type of an attribute of an entity data type.

IFC4_ADD1 contains in total four occurrences of case 1, namely IfcCompoundPlaneAngleMeasure (LIST[3:4]), IfcLineIndex (LIST[2:?]), IfcArcIndex (LIST[3:3]), and IfcComplexNumber (ARRAY[1:2]). Three of those are repeated for reference in Fragment 2. Cardinality restrictions are declared between square brackets. IFC2X3 only contains two occurrences of case 1, namely IfcCompoundPlaneAngleMeasure and IfcComplexNumber.

These defined data types can be referred to in a number of ways. In the case of IFC, they are typically referred to from the attribute of an entity data type, for example IfcSite.RefLatitude, or from a SELECT data type declaration that includes this defined data type as one of its implementing types. As a further example, the SELECT data type IfcValue can be an IfcMeasureValue, which can be an IfcComplexNumber, which is an ARRAY [1:2] OF REAL.

Fragment 2: Example of three data type declarations referring to aggregation data types (LIST, ARRAY).

```plaintext
2 WHERE
3 MinutesInRange : ABS(SELF[2]) < 60;
4 SecondsInRange : ABS(SELF[3]) < 60;
5 MicrosecondsInRange : (SIZEOF(SELF) = 3) OR (ABS(SELF[4]) < 1000000);
6 ConsistentSign :
8 OR
11 END_TYPE;
12 TYPE IfcLineIndex = LIST [2:?] OF IfcPositiveInteger;
13 END_TYPE;
14 TYPE IfcComplexNumber = ARRAY [1:2] OF REAL;
16 END_TYPE;
```

These defined data types can be referred to in a number of ways. In the case of IFC, they are typically referred to from the attribute of an entity data type, for example IfcSite.RefLatitude, or from a SELECT data type declaration that includes this defined data type as one of its implementing types. As a further example, the SELECT data type IfcValue can be an IfcMeasureValue, which can be an IfcComplexNumber, which is an ARRAY [1:2] OF REAL.
There are considerably more occurrences of ordered aggregation data types of case 2. Fragment 3 gives an example of case 2 for the attribute Coordinates of IfcCartesianPoint that consists in a LIST of IfcLengthMeasure. Of the 768 entity data types in IFC4_ADD1, 74 have attributes (excluding DERIVE attributes) that refer to LIST aggregation data types and two (IfcMaterialLayerWithOffsets and IfcMaterialProfileWithOffsets) have an attribute (OffsetValues for both of them) that refers to an ARRAY aggregation data type. In total, 98 attributes point directly to a LIST aggregation data type, and two attributes point directly to an ARRAY aggregation data type. An overview of these entities and attributes for IFC4_ADD1 is given in Table 2. Note that some of these attributes are typically applied on a subtype level, e.g. IfcTypeProduct.RepresentationMaps can be used by an instance of IfcWindowType.

1 ENTITY IfcCartesianPoint
2 SUBTYPE OF (IfcPoint);
3 Coordinates : LIST [1:3] OF IfcLengthMeasure;
4 DERIVE
5 Dim : IfcDimensionCount := HIINDEX(Coordinates);
6 WHERE
7 CP2Dor3D : HIINDEX(Coordinates) >= 2;
8 END_ENTITY;

Fragment 3: EXPRESS specification of IfcCartesianPoint

In the case of IFC2X3_TC1, 51 entities have attributes (excluding DERIVE attributes) that refer to LIST aggregation data types and zero entities have an attribute that refers to an ARRAY aggregation data type. In total, 66 attributes point directly to a LIST aggregation data type.

It is thus clear that ordered aggregation data types are predominantly LIST data types and they are mainly used via attributes of entity data types (case 2). In addition, the aggregation data types can be classified according to other dimensions. Firstly, attributes can be optional; for example, it is allowed to instantiate IfcPerson without specifying an ordered list in the attribute IfcPerson.MiddleNames. Secondly, the aggregation data types can point to one-dimensional (LIST) or two-dimensional lists (LIST OF LIST). In principle, more-dimensional aggregation data types (i.e. more than two) are possible as well, but they are not used in IFC. Finally, the aggregation data types are always associated with cardinality restrictions. These cardinality restrictions can include (1) only a lower bound (unbound in Table 3, e.g. LIST[1:?]) or (2) both a lower and an upper bound (bound in Table 3, e.g. LIST[2:3]). Table 3 illustrates the statistics of ordered aggregate data types in IFC4_ADD1 along these dimensions.

As can be concluded from Table 3, LIST data types are predominantly one-dimensional and unbound. Of these unbound LIST data types, 72 out of 89 (80,8%) have cardinality restrictions set to [1:?] for IFC4_ADD1. So, of the 100 ordered aggregation data types, 72 are of the form LIST[1:?] OF [type range]. In IFC4_ADD1, 48 distinct type ranges are used. The most prevalent type ranges
Table 2: Overview of the entity data types in IFC4_ADD1 and their attributes that point to LIST and ARRAY aggregation data types

<table>
<thead>
<tr>
<th>IFC2X3</th>
<th>IFC4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>case 1</td>
</tr>
<tr>
<td>LIST</td>
<td>1</td>
</tr>
<tr>
<td>ARRAY</td>
<td>1</td>
</tr>
<tr>
<td>optional</td>
<td>n/a</td>
</tr>
<tr>
<td>required</td>
<td>n/a</td>
</tr>
<tr>
<td>1D</td>
<td>2</td>
</tr>
<tr>
<td>2D</td>
<td>0</td>
</tr>
<tr>
<td>bound</td>
<td>2</td>
</tr>
<tr>
<td>unbound</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3: Ordered aggregate data types can be categorised along a number of dimensions: LIST vs. ARRAY, optional vs. required, one-dimensional vs. two-dimensional, bound vs. unbound.

and their number of occurrences are reported in Table 4. Note that seven of these data type ranges are very closely related to the simple data types that are set by the IFC schema. For example, IfcLengthMeasure is a defined data type that points directly to the simple data type REAL. Seven other prevalent data type ranges are entity data types, whereas two data type ranges are SELECT data types.
Table 4: The number of occurrences of the most important (occurrence higher than 1) type ranges associated to a LIST or ARRAY data type.

<table>
<thead>
<tr>
<th>Type</th>
<th>Occurrences</th>
<th>Distance from simple data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>IfcLengthMeasure</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>IfcValue</td>
<td>8</td>
<td>n/a (SELECT)</td>
</tr>
<tr>
<td>IfcLabel</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>IfcInteger</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>IfcParameterValue</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>IfcCartesianPoint</td>
<td>4</td>
<td>n/a (ENTITY)</td>
</tr>
<tr>
<td>IfcGridAxis</td>
<td>4</td>
<td>n/a (ENTITY)</td>
</tr>
<tr>
<td>IfcPositiveInteger</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>IfcReal</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>IfcActorRole</td>
<td>3</td>
<td>n/a (ENTITY)</td>
</tr>
<tr>
<td>IfcAppliedValue</td>
<td>3</td>
<td>n/a (ENTITY)</td>
</tr>
<tr>
<td>IfcAddress</td>
<td>2</td>
<td>n/a (ENTITY)</td>
</tr>
<tr>
<td>IfcBendingParameterSelect</td>
<td>2</td>
<td>n/a (SELECT)</td>
</tr>
<tr>
<td>IfcIdentifier</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>IfcOrientedEdge</td>
<td>2</td>
<td>n/a (ENTITY)</td>
</tr>
<tr>
<td>IfcSurfaceTexture</td>
<td>2</td>
<td>n/a (ENTITY)</td>
</tr>
</tbody>
</table>

2.2 IFC-SPF instance file statistics for ordered aggregation data types

Not all the LIST and ARRAY data types defined in the IFC schema are commonly used in practice. In order to obtain an estimate of the impact on actual IFC-SPF files, we have investigated how many times these data types occur in a test set of IFC files. The test set consists of the IFC2X3 file that is part of the public Schependonlaan dataset (van Schaijk, 2016) and the IFC4 files that are available as part of the IfcDoc tool (BuildingSMART International, 2016b). Not many IFC4 files are currently available, as commercial implementation of the IFC4 standard is still underway. It is expected, nonetheless, that similar results will be obtained for IFC4 files as soon as they become more common in construction industry practice. For reference, we have therefore also analysed the statistics for the IFC4_ADD1 examples provided as part of the IfcDoc tool. The results of this assessment are displayed in Table 5 for IFC2X3_TC1 (top) and IFC4_ADD1 (bottom).

This table shows that only a small number of the LIST and ARRAY data types is ever instantiated in IFC-SPF files. In the IFC4_ADD1 reference test set, only 12 distinct LIST and ARRAY attributes are used of the 66 possible attributes pointing towards LIST and ARRAY data types (18.2%). This is even lower for the IFC2X3_TC1 reference test set (10/66 = 15.2%). Of these cases, only two to three represent a significant share (i.e. higher than 3%) in the total test set, namely IfcCartesianPoint.Coordinates (16.4% and 14.3%), IfcDirection.DirectionRatios (2.2% and 8.7%) and IfcPolyLoop.Polygon (7.5% and 3.4%), all of them representing geometric concepts. As all IfcPolyLoop.Polygon attributes point to an IfcCartesianPoint instance, these lines in Table 5 are clearly correlated. This also means that a relevant part of an IFC-SPF file is typically made of IfcPolyLoop and IfcCartesianPoint instances, which are in the end simply IfcLengthMeasure instances, or REAL values.

Table 5 furthermore shows that a number of these LIST and ARRAY data ty-
TABLE 5: The share (%) of LIST and ARRAY data type instances in the IFC2X3 Dataset Schependomlaan (top) and the IFC4_ADD1 IfcDoc example files (bottom).

3 Alternative representations for ordered aggregation data types

As was already mentioned, ordered sequences cannot be easily represented in an RDF graph, because RDF relies on a triple structure that inherently allows to link only two concepts, not collections of multiple concepts. Therefore, complex expressions are typically needed to represent ordered aggregation data types in an RDF graph, as already discussed in several other studies (Pauwels and Terkaj, 2016; Hoang and Törmä, 2014; Hoang, 2015; Barbau et al., 2012; Krima et al., 2009). Considering that a relevant part of an IFC-SPF file typically consists of LIST data type instantiations (see Sect. 2), the conversion of these data types to OWL classes plays an important role in terms of usability.

We consider two extreme dimensions in terms of usability, namely semantic
precision versus computational performance. In many cases, improved semantic precision goes at the expense of computational performance and vice versa. In other words, lengthy representations of LIST types result in precise and correct definitions, but they require notably more space to be stored and more time to be loaded in-memory and used. If performance matters significantly, then it is a better option to use semantically less precise representations of the same information. Therefore, we investigate a number of conversion procedures, while looking specifically at semantic precision and computational performance:

1. Regular rdf:list concepts natively available in RDF
2. General purpose list concepts
3. Customised typed concepts that do not reflect a list structure
4. Customised untyped concepts referring to portions of Well-Known Text (WKT)

For each of these four procedures, we will show how a reference instance of IfcCartesianPoint can be converted (see the regular IFC-SPF expression in Fragment 4). Indeed, this is one of the most often used ordered aggregation data types (see Table 5), but it is definitely not the only one (see Table 2 to 4). Hence, the examples in this section only serve to illustrate the conversion procedure itself. The Turtle syntax (Beckett and Berners-Lee, 2011) will be used to present the OWL and RDF fragments, while referring to the namespaces defined in Fragment 5. Note that the five namespaces in lines 5 to 10 in Fragment 5 are dummy namespaces that are used to distinguish between the ontologies and approaches used in the following sections of this paper. Only one of them should eventually be used and it should reside under namespace http://ifcowl.openbimstandards.org/ifcOWL# or similar. Also the namespace at line 15 is a dummy namespace that represents the instance graph of a BIM model in IFC. All test files used for this section are made available at (Pauwels et al., 2016).

Fragment 4: Reference example of IfcCartesianPoint instance described according to the IFC EXPRESS schema.

```
#37=IFCCARTESIANPOINT((0.,0.,-350.));
```

1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
2 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
3 @prefix owl: <http://www.w3.org/2002/07/owl#> .
4 @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
5 @prefix list: <https://w3id.org/list#> .
6 @prefix expr: <https://w3id.org/express#> .
7 @prefix olo: <http://purl.org/ontology/olo/core#> .
8 @prefix ifcowl: <http://ifcowl.openbimstandards.org/ifcOWL#> .
9 @prefix ifcowl2a: <http://ifcowl.openbimstandards.org/ifcOWL2a#> .
10 @prefix ifcowl2b: <http://ifcowl.openbimstandards.org/ifcOWL2b#> .
11 @prefix ifcowl3a: <http://ifcowl.openbimstandards.org/ifcOWL3a#> .
12 @prefix ifcowl3b: <http://ifcowl.openbimstandards.org/ifcOWL3b#> .
3.1 Procedure 1: standard RDF lists

Lists can be represented in RDF by linking each item in the sequence to the next item by using `<rdf:List>`, `<rdf:first>` and `<rdf:rest>` declarations (Brickley and Guha, 2014; Dean and Schreiber, 2004). It is however commonly used in the serialisation of OWL. If it is then also used in RDF graphs, then it would mean that the same concept (`<rdf:List>`) is available both in the ontology (TBox) and in the instances (ABox), which is not allowed in OWL(2) DL. Hence, using this construct would result in an ontology that is beyond OWL DL and is thus undecidable. In other words, this choice impedes the use of reasoning engines that rely on Description Logics (DL (Baader and Nutt, 2003)). In addition, this conversion option results in verbose representations for the instantiation of the LIST concepts in the ontology. The reference instance of `IfcCartesianPoint` (see Fragment 4) results in the expression given in Fragment 6. A visual diagram of this structure is displayed in Fig. 2.
It can be clearly seen that the OWL expression in Fragment 6 is far more verbose and complex (21 triples) compared to the SPF representation in Fragment 4. Considering the high number of lists that are defined in the IFC EXPRESS schema and instantiated in IFC files, the impact on complexity and data volume is high.

### 3.2 Procedure 2: general purpose list concepts

Another option consists in defining a general purpose List concept for OWL. By doing so, one is not interfering with the rdf:List concept that is commonly used for RDF serialisations and thus belongs to the ABox. As such, it is possible to keep the ontology in OWL DL expressiveness and decidability, which allows to use reasoning or inference engines. This is a major improvement over the above Procedure 1, because logical inference is one of the most important additional features provided by OWL that is not available in other data modelling approaches.

As a result, this is the currently recommended procedure in the ifcOWL ontology, as this is the only procedure that allows to remain in OWL DL and that maintains both the aggregation type information and order that is available in the original
EXPRESS data type declarations.

We distinguish two main proposals in defining such list concepts, Procedure 2a and 2b, and we document them in greater detail in this section. For both approaches, we indicate what they (can) result in when using them for ordered aggregations in ifcOWL, as proposed by Krima et al. (2009); Barbau et al. (2012); Pauwels and Terkaj (2016); Hoang and Törmä (2014); Hoang (2015).

3.2.1 Procedure 2a: using the linked list proposal by Drummond et al.

An early proposal for handling ordered aggregations in OWL without the use of \texttt{rdf:List} was made by Drummond et al. (2006). They found that OWL DL did not provide in-built support for ordered lists and proposed an OWLList ontology. This ontology is available at Information Management Group, University of Manchester (2015). In their proposal, they follow a structure that is similar to the original \texttt{rdf:List} structure (see Fig. 2 and Fragment 6). A schema of their proposal is shown in Fig. 3, reproduced from Drummond et al. (2006).

![Diagram of the List structure proposed by Drummond et al. (2006) for representing ordered aggregations.](image)

In their proposal, Drummond and colleagues propose to follow the standard structure of linked lists, namely, “each item is held in a ‘cell’ (\texttt{OWLList}); each cell has contents (‘head’) and a pointer to the next cell (‘tail’); and the end of the list is indicated by a terminator (\texttt{EmptyList}) which also serves to represent the empty list.” (Drummond et al., 2006). A similar linked list proposal is also recommended in the Semantic Web Best Practices document regarding n-ary relations on the semantic web (Noy and Rector, 2006, Pattern 2).

The linked list approach is followed by Krima et al. (2009); Barbau et al. (2012), Pauwels and Terkaj (2016) and the ifcOWL currently hosted under buildingSMART (BuildingSMART International, 2016a). As an example, Fragment 7 shows how the \texttt{IfcCartesianPoint} given in Fragment 4 can be converted using the approach proposed by (BuildingSMART International, 2016a) and consistent with Pauwels and Terkaj (2016). This example is graphically represented in Fig. 4.

1 \texttt{ifcinst:IfcCartesianPoint_37}
The proposal by Pauwels and Terkaj (2016) relies on the property definitions as defined by Drummond et al. (2006). These properties can be exploited to generate customized OWL declarations in the ifcOWL ontology to capture the original specification in the IFC EXPRESS schema, including the cardinality constraints on the list size (see Fragment 8).

Fragment 7: Representation of the example IfcCartesianPoint instance in Fragment 4, following the conversion procedure as proposed by (BuildingSMART International, 2016a) and documented in Pauwels and Terkaj (2016).

Fragment 8: OWL declarations for the IfcCartesianPoint class.
Figure 4: Diagram of the representation of an `ifc:CartesianPoint` using the linked list approach proposed by Drummond et al. (2006), and as implemented by Pauwels and Terkaj (2016) for ifcOWL.

```owl
owl:allValuesFrom
[
  rdf:type owl:Restriction ;
  owl:onProperty list:hasNext ;
  owl:someValuesFrom ifcowl2a:IfcLengthMeasure_List
]
.

[ rdf:type owl:Restriction ;
  owl:onProperty ifcowl2a:coordinates_IffCartesianPoint ;
  owl:allValuesFrom
  [ rdf:type owl:Restriction ;
    owl:onProperty list:hasNext ;
    owl:allValuesFrom
      [ rdf:type owl:Restriction ;
        owl:onProperty list:hasNext ;
        owl:allValuesFrom
          [ rdf:type owl:Restriction ;
            owl:onProperty list:hasNext ;
            owl:allValuesFrom
              [ rdf:type owl:Restriction ;
                owl:onProperty list:hasNext ;
                owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger
              ]
            ]
          ]
        ]
      ]
    ]
  ]
].
```
The classes and properties defined in the ifcowl2a namespace by Pauwels and Terkaj (2016) that allow to represent the relations (edges) in the RDF graph following ifcOWL (see example in Fragment 7).

The proposal by Barbau et al. (2012) is slightly different, as they propose to use more specific property definitions that are subproperties of those introduced by Drummond et al. (2006). For the example of LIST OF IfcLengthMeasure, which is used in the definition of an IfcCartesianPoint, this results in the sub-properties that are displayed in Fragment 9. Such a structure is more restricting than the proposal made by Pauwels and Terkaj (2016) and displayed in Fragment 8, but it also generates considerably more complexity and verbosity in the ontology, because similar properties as shown in Fragment 9 must be generated for all 48 type ranges displayed in Table 2, thus resulting in 192 additional property declarations. Moreover, this conversion pattern lacks the representation of cardinality constraints.

Fragment 8: The classes and properties defined in the ifcowl2a namespace by Pauwels and Terkaj (2016) that allow to represent the relations (edges) in the RDF graph following ifcOWL (see example in Fragment 7).

The proposal by Barbau et al. (2012) is slightly different, as they propose to use more specific property definitions that are subproperties of those introduced by Drummond et al. (2006). For the example of LIST OF IfcLengthMeasure, which is used in the definition of an IfcCartesianPoint, this results in the sub-properties that are displayed in Fragment 9. Such a structure is more restricting than the proposal made by Pauwels and Terkaj (2016) and displayed in Fragment 8, but it also generates considerably more complexity and verbosity in the ontology, because similar properties as shown in Fragment 9 must be generated for all 48 type ranges displayed in Table 2, thus resulting in 192 additional property declarations. Moreover, this conversion pattern lacks the representation of cardinality constraints.

Fragment 9: The classes and properties defined in the ifcowl2a namespace by Pauwels and Terkaj (2016) that allow to represent the relations (edges) in the RDF graph following ifcOWL (see example in Fragment 7).
Fragment 9: The classes and properties defined in the ifcOntoStep namespace by Krima et al. (2009); Barbau et al. (2012) that allow to represent ordered lists.

3.2.2 Procedure 2b: using the Ordered List Ontology (OLO)

The Ordered List Ontology (OLO) is a proposal by Abdallah and Ferris (2010) that does not rely on the linked list concept, but rather maintains order in a list using olo:Slot class instances that have a specific olo:index data type properties and olo:item object properties. A schematic overview of this structure is provided in Fig. 5.

In this proposal, the distinct items in the ordered list are not directly linked to each other. Accessing the distinct members of the ordered lists should thus not happen by iterating through all elements of the ordered list, but rather by selecting the item that one needs, using the index that is provided with that item. However, it must be noted that the representation of the item index as a literal is semantically weak and error prone, since general purpose reasoners could fail to detect possible mistakes in the indexes. Optionally, one could include an additional property olo:next Abdallah and Ferris (2010) to explicitly represent the direct
Figure 5: Diagram of the List structure proposed by Abdallah and Ferris (2010) for representing ordered aggregations.

links between list items, but in this case the solution becomes very similar to the proposal by Drummond et al. (2006).

Similar to Fragment 8 and 9, several additional concepts and properties must be declared in the ifcOWL ontology (see Fragment 10), even if the ontology relies directly on the generic properties `expr:slot`, `expr:item`, `expr:length` and `expr:index`, without requiring more specific and more restricted subproperties.

```plaintext
1  ifcowl2b:IfcCartesianPoint rdfs:type owl:Class ;
2   rdf:subClassOf
3      [ rdf:type owl:Restriction ;
4        owl:onProperty ifcowl2b:coordinates_IfcCartesianPoint ;
5        owl:onClass ifcowl2b:IfcLengthMeasure_List ;
6        owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger
7 ] ,
8      [ rdf:type owl:Restriction ;
9        owl:onProperty ifcowl2b:coordinates_IfcCartesianPoint ;
10       owl:allValuesFrom
11      [ rdf:type owl:Restriction ;
12        owl:onProperty olo:slot ;
13        owl:onClass ifcowl2b:IfcLengthMeasure_Slot ;
14        owl:maxQualifiedCardinality "3"^^xsd:nonNegativeInteger
15     ] ,
16      [ rdf:type owl:Restriction ;
17        owl:onProperty ifcowl2b:coordinates_IfcCartesianPoint ;
18        owl:allValuesFrom ifcowl2b:IfcLengthMeasure_List
19     ] .
20
21  ifcowl2b:IfcLengthMeasure_List rdf:type owl:Class ;
22    rdfs:subClassOf olo:OrderedList ,
23    [ rdf:type owl:Restriction ;
24     ] .
25
26  ifcowl2b:IfcLengthMeasure_Slot rdf:type owl:Class ;
27    rdfs:subClassOf olo:OrderedList ,
28    [ rdf:type owl:Restriction ;

The properties defined in the ifcowl2b namespace that allow to represent ordered list structures relying on the OLO ontology.

The adoption of a solution based on OLO ontology results in an RDF graph for an IfcCartesianPoint as displayed in Fig. 6. As a further example the proposal for an ifcOWL ontology made by Hoang and Törmä (2014); Hoang (2015) relies on the OLO ontology.

Although the mentioned Procedures 2a and 2b are viable alternatives that allow to build an ifcOWL ontology in OWL DL, their results do not improve Procedure 1 in terms of compactness and computational efficiency. Indeed, the representation in Procedure 2a (see Fragment 7) is about as long as the representation in Fragment 6. The key differences can be found in lines 7, 8, 16, 17, 25 and 26, where the properties ifcowl2a:hasContents and ifcowl2a:hasNext replace the standard RDF terms rdf:first and rdf:rest.

3.3 Procedure 3: customized list concepts

A third option that could be considered is to add entirely new concepts that are customized for the specific list, as was proposed earlier in citetegice2015. For example, in the case of the IfcCartesianPoint (Fragment 4), it is possible to create new properties to directly point to each item in a list of two or three coordinates (IfcLengthMeasure), thereby implicitly allowing to distinguish between 2D and 3D IfcCartesianPoint concepts.

The results of the application of Procedure 3a are reported in Fragment 11 and shown in Fig. 7, where the object property ifcowl2:coordinates_IfcCartesianPoint (e.g. line 3 in Fragment 7) is replaced by three separate object properties that point directly to the appropriate IfcLengthMeasure instances (line 3-5 in Fragment 11).

<table>
<thead>
<tr>
<th>ifcinst:IfcCartesianPoint_37</th>
<th>rdf:type ifcowl3a:IfcCartesianPoint , owl:NamedIndividual;</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
Figure 6: Diagram of the representation of an `IfcCartesianPoint` using the OLO approach proposed by Abdallah and Ferris (2010). Note that we added the `expr:hasDouble` property that is required to link a class instance with its actual data value.

```
3 ifcowl3a:xCoordinate_of_IfcCartesianPoint ifcinst:IfcLengthMeasure_371inst ;
4 ifcowl3a:yCoordinate_of_IfcCartesianPoint ifcinst:IfcLengthMeasure_372inst ;
5 ifcowl3a:zCoordinate_of_IfcCartesianPoint ifcinst:IfcLengthMeasure_373inst .
6
7 ifcinst:IfcLengthMeasure_371inst
8   rdf:type ifcowl3a:IfcLengthMeasure , owl:NamedIndividual ;
9   expr:hasDouble "0.0"^^xsd:double .
10
11 ifcinst:IfcLengthMeasure_372inst
12   rdf:type ifcowl3a:IfcLengthMeasure , owl:NamedIndividual ;
13   expr:hasDouble "0.0"^^xsd:double .
14
15 ifcinst:IfcLengthMeasure_373inst
16   rdf:type ifcowl3a:IfcLengthMeasure , owl:NamedIndividual ;
17   expr:hasDouble "-350.0"^^xsd:double .
```

Fragment 11: Representation of the example `IfcCartesianPoint` instance of Fragment 4 following the conversion `Procedure 3a` that explicitly adds new semantic concepts.

Furthermore, the proposed `Procedure 3b` replaces the object property `ifcowl2:coordinates_-IfcCartesianPoint` (line 3 in Fragment 7) with three separate data type properties, as shown in Fragment 12 and Fig. 8. The data type properties point directly to the appropriate literal values of type `ifcowl4:IfcLengthMeasure` (line 3-5 in Fragment 12), which is then defined as an equivalent class (`owl:equivalentClass`)
of \texttt{xsd:double}. Such a formal representation is a major improvement over the representations presented in Fragment 6 and 7 (and 11) in terms of length, complexity and readability. The resulting RDF graphs following this conversion procedure can thus be loaded and used far more efficiently. In addition, the resulting RDF graph is also more meaningful, as the relations between the \texttt{IfcCartesianPoint} concept and its coordinates are made explicit.

Figure 7: Diagram of the representation of an \texttt{ifc:CartesianPoint} using Procedure 3a.

Figure 8: Diagram of the representation of an \texttt{ifc:CartesianPoint} using Procedure 3b.

However, it must be noted that both \texttt{Procedure 3a} and \texttt{3b} require the generation of concepts that are beyond the original EXPRESS schema. Indeed, the
new properties cannot be mapped to definitions in the EXPRESS schema where there is only ‘a list with ordered coordinates’, thus jeopardizing the use of general purpose SPF to RDF (and vice versa) converters. Moreover, Procedure 3b makes direct use of data type properties, thus discarding the ‘objectification’ pattern that was proposed in Schevers and Drogemuller (2005) and used in (Beetz et al., 2009; Barbau et al., 2012; Pauwels and Terkaj, 2016). This objectification pattern is required to correctly convert SELECT data types in EXPRESS to equivalent expressions in OWL. Hence, there is a risk that the proposal in Fragment 12, if generically applied to all ordered lists encountered in an EXPRESS schema, leads to infeasible representations of SELECT data types in OWL. Finally, this approach works fairly well with a bounded ordered list like IfcCartesianPoint, as we can explicitly name each of the two or three datatype properties. Although IfcCartesianPoint happens to be one of the most elaborately used aggregation data types, 90 percent of the aggregation data types is unbounded (see Table 3), for which this procedure is not feasible.

3.4 Procedure 4: well-known text (WKT)

3.4.1 Well-Known Text for IFC

The suitability and usefulness of describing extensive collections of, for example, Cartesian points using notations compatible with DL mechanisms can be considered limited in many use cases. Namely, using general purpose inference mechanisms to address complex computational geometry problems will limit the performance or will be impossible without dedicated (procedural, non-DL) functions as provided by geometry kernel libraries. When interacting with such structures, most applications will map and transform them into more efficient, intermediary structures. Even if dedicated support for ordered collections in reasoning and query engines might be expected in the future, the verbosity of standard approaches discussed in Sect. 3.1 and 3.2 will most certainly hamper their efficient uptake and use at larger scales on data volume to be processed alone.

Therefore, a fourth procedure considered here relies on the usage of well-known text (WKT - (International Organization for Standardization, 2011)). WKT has been proposed in the geospatial domain, where large amounts of geometrical information must be represented both efficiently (short representation) and in a semantically meaningful manner. In this proposal, numeric values that are part of a particular concept or entity (e.g. a Cartesian point) are represented using (1) a keyword in upper case followed by the parameters of the object in parentheses. This procedure applied to the case of the IfcCartesianPoint results in the formal representation illustrated in Fragment 13 and Fig. 9.

```
1 ifcinst:IfcCartesianPoint_37
2 rdf:type ifcowl4:IfcCartesianPoint ;
3 ifcowl4:coordinates_ifcCartesianPoint "POINT Z(0.0 0.0 -350.0)"^^ifcowl4:wktPoint
```
Fragment 13: Representation of the example IfcCartesianPoint instance in Fragment 4, following the conversion Procedure 4 that relies on the use of WKT.

Figure 9: Diagram of the representation of an ifc:CartesianPoint using the WKT approach (Procedure 4).

This novel procedure based on WKT relies on the OWL definitions shown in Fragment 14 and provides a formal representation that is clearly a relevant improvement over the representations presented in Fragments 6, 7, 11, and 12 in terms of length, complexity and readability.

Fragment 14: The classes and properties defined in the ifcowl4 namespace that allow to represent ordered lists.

In the case of Fig. 9, we propose to re-use the WKT agreements made in the geospatial domain as a basis. Such re-use of existing vocabularies is also in line with W3C’s Linked Data Best Practices (Hyland et al., 2014). As an example, it was agreed in the geospatial domain to represent a Cartesian point using the POINT WKT string. Additional elements that are currently not available in the geospatial WKT proposal would have to be added though, either in the original geospatial WKT list, or in a construction industry-specific list that imports the original geospatial WKT.
The advantage with respect to Procedure 3 is that this approach can likely remain closer to the original EXPRESS schema: there are no additional concepts and the instance representation in Fragment 4 is highly similar to the instance representation in Fragment 13. A possible loss in terms of semantic precision can be limited by formally defining a customised data type that is pointed to by `ifcowl4:coordinates`, instead of purely relying on informal agreements between the end users on how the WKT string must be composed. This customized data type can be restricted by exploiting facets such as `xsd:pattern`, as shown in lines 17–22 of Fragment 14 for `ifcowl4:wktPoint`. However, it remains to be seen if the correctness of these data type values can actually be checked by semantic inference engines. Yet, such a trade-off between semantic precision and computational efficiency needs to be considered for domains with lots of numeric value aggregations, such as the geospatial domain and the geometric parts of the construction industry.

The mapping to WKT offers a plausible alternative to separate instantiations for coordinate components, because the equivalents for these concepts exist in the WKT specification. Higher order geometrical representations also have WKT equivalents. For example, the `IfcPolyline` entity, a representation item consisting of a single attribute typed as a LIST of `IfcCartesianPoint`, can accurately be expressed as a LINESTRING WKT (see Fragment 15). Such an approach would reduce the number of nodes in the RDF graph even more, as multiple `IfcCartesianPoint` instances would be folded into a single WKT string object.

```
#54=IFCPOLYLINE((#44,#45,#46,#47,#48,#44));
#44=IFCCARTESIANPOINT((-5000.,-180.));
#45=IFCCARTESIANPOINT((5000.,-180.));
#46=IFCCARTESIANPOINT((5000.,180.));
#47=IFCCARTESIANPOINT((-5000.,180.));
#48=IFCCARTESIANPOINT((-5000.,-180.));
8 LINestring(-5000.0 -180.0, 5000.0 -180.0, 5000.0 180.0, -5000.0 180.0, -5000.0 -180.0, -5000.0 -180.0)
```

Fragment 15: An IfcPolyline in IFC-SPF (lines 1-6) and resulting WKT LINESTRING (line 8)

The outlined WKT representation method can be used for higher order faceted boundary representations as well. As an example, the `IfcTriangulatedFaceSet`, which is newly introduced in IFC4 and contains two attributes pointing to two-dimensional lists (2-dimensional matrix of `INTEGER` values), can be converted to a POLYHEDRALSURFACE Z.

3.4.2 Representing ordered lists with WKT strings

With the existing set of WKT definitions, the representations that can be expressed are limited to polyhedral surfaces and line segments. As a reference, Table 6 shows the attributes listed earlier in Table 5 and their equivalents in WKT. The last column indicates what IFC EXPRESS modules the concepts are part of,
thereby giving an indication to what extent the concepts represent geometry or not. Note that this entity listing is to be interpreted hierarchically: instances of IfcCartesianPoint that are solely referenced as part of an IfcPolyline will not be instantiated in the resulting graph when representing them using WKT expressions. Instead, they will be directly included in the IfcPolyline representation.

<table>
<thead>
<tr>
<th>IFC Attribute</th>
<th>WKT Literal</th>
<th>IFC module</th>
</tr>
</thead>
<tbody>
<tr>
<td>IfcPolyline.Polygon</td>
<td>LINES (Z)</td>
<td>Topology Resource</td>
</tr>
<tr>
<td>IfcPolyline.Points</td>
<td>LINES (Z)</td>
<td>Geometry Resource</td>
</tr>
<tr>
<td>IfcCartesianPointList3D.CoordList</td>
<td>MULTIPOINT Z</td>
<td>Geometric Model Resource</td>
</tr>
<tr>
<td>IfcEdgeLoop.EdgeList</td>
<td>not in WKT</td>
<td>Topology Resource</td>
</tr>
<tr>
<td>IfcVirtualGridIntersection.IntersectingAxes</td>
<td>not in WKT</td>
<td>Geometric Constraint Resource</td>
</tr>
<tr>
<td>IfcVirtualGridIntersection.OffsetDistances</td>
<td>not in WKT</td>
<td>Geometric Constraint Resource</td>
</tr>
<tr>
<td>IfcCompositeCurve.Segments</td>
<td>not in WKT</td>
<td>Geometry Resource</td>
</tr>
<tr>
<td>IfcProductRepresentation.Representations</td>
<td>non-geometric</td>
<td>Representation Resource</td>
</tr>
<tr>
<td>IfcTypeProduct.RepresentationMaps</td>
<td>non-geometric</td>
<td>Kernel</td>
</tr>
<tr>
<td>IfcRelConnectsPathElements.RelatedPriorities</td>
<td>non-geometric</td>
<td>Shared Bldg Elements</td>
</tr>
<tr>
<td>IfcRelConnectsPathElements.RelatingPriorities</td>
<td>non-geometric</td>
<td>Shared Bldg Elements</td>
</tr>
<tr>
<td>IfcMaterialLayerSet.MaterialLayers</td>
<td>non-geometric</td>
<td>Material Resource</td>
</tr>
<tr>
<td>IfcMaterialList.Materials</td>
<td>non-geometric</td>
<td>Material Resource</td>
</tr>
<tr>
<td>IfcMaterialProfileSet.MaterialProfiles</td>
<td>non-geometric</td>
<td>Material Resource</td>
</tr>
<tr>
<td>IfcPropertyEnumeration EnumerationValues</td>
<td>non-geometric</td>
<td>Property Resource</td>
</tr>
</tbody>
</table>

Table 6: Representation of the IFC attributes listed in Table 5 to WKT literals.

Compared to the body of representation items that are available in the IFC schema, the number of directly matching WKT representations is limited. For example, non-geometric concepts are not included, nor are particular kinds of geometry that are not used in the geospatial domain (e.g. directions or vectors). This is to be expected as IFC is conceived to communicate parametric geometry with design intent. Examples of this include the Segments attribute of IfcCompositeCurve and the EdgeList attribute of IfcEdgeLoop. Such structures are suitable for precise and curved solid boundary representation models, beyond what is typically required in the geospatial domain. The IntersectingAxes and OffsetDistances attributes of IfcVirtualGridIntersection point to a LIST[2:2] of IfcGridAxis and LIST[2:3] of IfcLengthMeasure respectively. The latter is identical to the Coordinates attribute of IfcCartesianPoint. Although we could use the WKT expression POINT (Z) here, the semantics of IfcVirtualGridIntersection.OffsetDistances are different. Hence, a distinct WKT expression is recommended instead (which does not exist currently).

If we keep our focus on the purely geometric items in Table 6, only 9 concepts remain, of which 4 have a direct WKT equivalent. If we also aim to serialise the semantically rich and curved-surface geometries in IFC as WKT, two options are available:

1. tessellate semantically rich and curved-surface geometries into faceted polyhedral surfaces and line segments, making the geometries less accurate

2. extend the WKT vocabulary to incorporate the superset of geometrical definitions offered by IFC
The first option results in a loss of information, as design intent cannot be included in the explicit tessellated geometric representations. To make it somewhat more specific, using this option would imply that, for example, an IfcCompositeCurve with circular segments would be approximated as a sequence of linear segments, for which the WKT equivalent LINESTRING exists. The second option seems unfeasible due to the necessary implementation effort in linked data reasoners and query engines to make them include and understand the strings that are specific to IFC or STEP (ISO 10303). If we could deploy this strategy, however, it would result in newly introduced WKT strings, such as DIRECTION Z(1.0 0.0 0.0) and OFFSETDISTANCE Z(20.0 5.0 15.5) for the easier examples in Table 6.

3.4.3 Representing unordered geometric data with WKT strings

Some geometry is unordered in nature. This geometry is typically represented in EXPRESS using SETs instead of LISTs. As we focused almost entirely on ordered aggregations in this paper (LIST), this geometry is not included in the above study. Two easy examples in this regard are IfcFace.Bounds and IfcConnectedFaceSet.CfsFaces. These attributes point toward unbounded sets of IfcFaceBounds and IfcFaces respectively. Although these concepts can be represented in a semantically rather straightforward fashion, this representation still generates a lot of data. Namely, a IfcConnectedFaceSet can point to a very large number of IfcFace instances (SET), each of which point to a number of IfcFaceBound instances (SET), which in turn typically point to an IfcPolyLoop instance, which points to a number of IfcCartesianPoint instances (LIST) (see Fragment 16). It is the level of indirection here, the number of intermediate instances, that generates a highly excessive amount of triples. This, however, is not specific to ifcOWL.

Nevertheless, instead of only representing the IfcPolyLoop instances as WKT strings, one could just as well turn the entire IfcConnectedFaceSet instance into one complete WKT literal. Examples of prevalent unordered geometry aggregates include IfcFace.Bounds and IfcConnectedFaceSet.CfsFaces, which can be represented using the POLYGON Z and POLYHEDRALSURFACE Z WKT string, respectively.

```
1 inst:IfcConnectedFaceSet_358 a ifcowl:IfcConnectedFaceSet ;
2 ifcowl:cfsFaces_IfcConnectedFaceSet
3   inst:IfcFace_367 ,
4   inst:IfcFace_370 .
5
6 inst:IfcFace_367 a ifcowl:IfcFace ;
7 ifcowl:bounds_IfcFace
8   inst:IfcFaceBound_368 ,
9   inst:IfcFaceBound_369 .
10
11 inst:IfcFaceBound_368 a ifcowl:IfcFaceBound ;
12 ifcowl:bound_IfcFaceBound inst:IfcPolyLoop_362 ;
13 ifcowl:orientation_IfcFaceBound inst:IfcBoolean_1571 .
```
3.4.4 To tessellate or not to tessellate?

From the above discussion, we can easily see two to three important decisions that need to be made for this WKT method to generate a representation that is usable throughout the industry:

1. Should we limit to ordered aggregations in the geometry (LISTs) or should we include WKT serialisation for unordered aggregations as well (SETs), e.g IfcConnectedFaceSet?

2. Should we limit WKT serialisation to pre-existing valid WKT expressions and remain faithful to their exact semantics; or should we introduce novel WKT expressions, such as an DIRECTION Z() for IfcDirection?

3. Should we always leave the instance files intact, or do we allow to simplify and tessellate instances in instance files to coerce them into pre-existing WKT expressions? E.g approximate curved surfaces by planar polygonal patches as a POLYHEDRALSURFACE Z.

These decisions have a considerable impact on the end result. Considering our review in the above section, we recommend at the moment to (1) not include unordered aggregations; (2) to limit to pre-existing valid WKT expressions; and (3) to keep the original instance representation in the original IFC-SPF files. This is a conservative and prudent recommendation, as it would mean that those instances that do not have a direct WKT equivalent (e.g. the middle 5 in Table 6) will not obtain any alternative WKT representation next to their existing ifcOWL representation.

As an indication, we briefly outline what a more progressive decision would result in, namely, when (1) including ordered aggregations, (2) limiting to existing valid WKT expressions, and (3) simplifying geometric instances into pre-existing WKT expressions. In this case, a lot of geometric data needs to be tessellated into a static geometric representation with a semantically poor and thus unchangeable structure. This tessellated geometric representation includes only the exact absolute coordinates of the particular element. As this is done also for unordered aggregations (Sect. 3.4.3), also the data that was shown earlier in Fragment 16 would be tessellated entirely into one POLYHEDRALSURFACE Z, losing all the geometric semantics defined in Fragment 16.
However, the resulting global coordinates of a building element in IFC are not solely dependent on the product’s own representation items. They are defined relative to other representation items using IfcLocalPlacement and IfcAxis2Placement3D instances. A placement provides an affine transformation that moves a representation item relative to another representation item. These placements are recursively defined in accordance with the decomposition structure of the IFC model. Furthermore, opening elements can be used to model voids (i.e. boolean subtractions) in related products by means of separate representation items. Figure 10 shows the geometric representation and placement in IFC of a simple IfcWall instance with a nested IfcWindow and Fragment 17 gives the IFC-SPF definition of the model sketched in Figure 10.

![Figure 10: A wall geometry affected by its LocalPlacement and relating void](image)

```
#49=IFCWALL('1wTUq08$GRvR2Sg67pPh9',#5,$,#35,#48,$);
#35=IFCLOCALPLACEMENT(#28,#34);
#34=IFCAXIS2PLACEMENT3D(#33,#7,#10);
#33=IFCCARTESIANPOINT((4.,1.,0.));
#7=IFCDIRECTION((0.,0.,1.));
#10=IFCDIRECTION((0.,1.,0.));
#28=IFCLOCALPLACEMENT($,#27);
#27=IFCAXIS2PLACEMENT3D(#8,#7,#6);
#8=IFCCARTESIANPOINT((0.,0.,0.));
#6=IFCDIRECTION((1.,0.,0.));
#48=IFCPRODUCTDEFINITIONSHAPE($,$,(#38,#47));
#38=IFCSHAPEREPRESENTATION(#11,'Axis','Curve2D',(#37));
#36=IFCCARTESIANPOINT((4.8,0.,0.));
#37=IFCPOLYLINE((#8,#36));
#47=IFCSHAPEREPRESENTATION(#11,'Body','SweptSolid',(#46));
```
If one tessellated POLYHEDRALSURFACE $Z$ needs to be obtained for this example, the absolute coordinates of each representation item in Fragment 17 would need to be obtained and included, thereby including the absolute coordinates of all its voids (for windows, doors, and other penetrations). This initial decision thus has a major impact on the eventual data structure of the BIM model. The resulting WKT literal for the “Body” representation of the wall in Fragment 17 (line 20) is outlined in Fragment 18. GEOMETRYCOLLECTION $Z$ is used as a representation to hold multiple items and POLYHEDRALSURFACE $Z$ is used as a representation for the wall solid volume. The $Z$ signifies the usage of three-dimensional geometry. In total, 10 faces are needed to model the resulting wall volume with genus 1, i.e. a donut from a topological point of view, with a single “hole”. Two of these faces, the first and last in Fragment 18, have inner rings. Each line in Fragment 18 thus represents one face and its absolute coordinates, with the first and last faces also representing the inner holes in them.
The representation in Fragment 17 contains only the rich semantic structure needed to calculate the absolute coordinates of the wall (including all sorts of other semantic properties, such as relationships with other elements in the IFC model); whereas the representation in Fragment 18 contains only the exact absolute coordinates of all the points of the wall. Indeed, none of the lines in Fragment 18 has a direct equivalent in the original IFC-SPF example, simply because all these coordinates are calculated from the initial representation. This calculation (the application of the IfcLocalPlacement iteration) is illustrated for one coordinate in Equation 1 for the underlined point #42 in Fragment 17, resulting in the underlined coordinate in Fragment 18.

\[
\begin{pmatrix}
0 & -1 & 0 & 4 \\
1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{pmatrix}
\times
\begin{pmatrix}
4.8 \\
0.36 \\
0 \\
1 \\
\end{pmatrix}
= 
\begin{pmatrix}
3.64 \\
5.8 \\
0 \\
1 \\
\end{pmatrix}
\]

Equation 1

This serialisation strategy removes more semantic information and flattens geometry definitions into strings that would be succinctly and expressively represented in ifcOWL, such as binary trees of computational solid geometries and sweeps of parametric profile definitions. Considering the loss of semantics and general invasiveness of this second WKT approach, it is not seen as a generally recommendable proposal at this point by the authors.

3.4.5 Impact on ifcOWL ontology and instances

To conclude this section on WKT for ifcOWL, Figure 11 a-c provide an overview of the implications on triple count for the considered conservative WKT serialisation option, applied to the IFC4_ADD1 and IFC2X3_TC1 test set respectively. Note that we have included all sample files online (Pauwels et al., 2016). Faceted geometries, such as basin-faceted-brep, will demonstrate a remarkable reduction in triple count. Counts are decimated after applying WKT to the efficient format for triangulated geometries introduced in IFC4 (e.g. basin-tesselation, in which a single triangulated mesh normally expressed as several lengthy multi-dimensional lists, can suddenly be encoded in a single WKT literal).

Note that the IfcDoc example files (IFC4_ADD1 - Fig. 11) are meant to demonstrate a variety of novel and (at the time of writing) rather esoteric IFC concepts. As such, they are meaningful to demonstrate how the serialisation options perform
on a wide variety of inputs. They are not necessarily representative for the contents of IFC exchanges in practice, in which explicit and polyhedral geometries (e.g. IfcFaceBasedSurfaceModel) at the time of writing are predominant. In this regard, the Schependomlaan data set (IFC2x3_TC1) is far more representative. Indeed, Fig. 11 c shows a decrease in triple count of nearly 50%.

For reference, we also include a triple count indication for another sample IFC2X3 file (Fig. 11 b), namely the “2011-09-14-Duplex IFC 2x3 Coordination MVD” file that is made available at BuildingSMART International (2011). Also here, the improvement in terms of triple count is around 50%, which is considerably higher than the relative improvements shown towards the left side in Fig. 11. This is caused by the fact that a regular real-world sample typically includes a lot more tessellated items and thus eventually also considerably more polygons, polylines and Cartesian points.

We do not include a comparison with the corresponding IFC-SPF files in terms of file size. Whereas an IFC model has only one serialisation option (IFC-SPF), an RDF graph has multiple serialisation options (TTL, RDF/XML, N3, ...), none of which is line-based. File sizes can thus vary significantly depending on the chosen serialisation mechanism for RDF graphs. Making a comparison in terms of file size between SPF file and serialised RDF graph is thus not a fair or correct comparison. As can be seen in Pauwels and Roxin (2016), however, serialised RDF graphs are always bigger and more verbose compared to SPF files, unless content is cut from the file. One can also clearly see in Fragment 4 that it is close to impossible to be less verbose than what is included in an IFC-SPF representation.
4 Discussion and conclusion

The inherent multi-domain nature of building projects, the high degree of collaboration between stakeholders involved in them and the large amount of heterogeneous data processed make the use of modularized data distributable over network structures and across file boundaries highly desirable. With the Semantic Web and Linked Data efforts and their well-accepted technological enablers including RDF and OWL, the building domain has the opportunity to address these needs using established technologies. However, the sheer size of commonly used model schema and resulting instance data sets make optimizations necessary to successfully employ these concepts. In particular, as indicated at the outset of this paper, information models for the domain of architectural design and construction typically rely on a lot of aggregate data and ordered collections for the representation of geometric structures. An example is the intensive use of LIST data types in IFC files. These data types do not map well on formal representations in OWL, simply because there is limited support for ordered aggregation types in mostly set-based information modelling scenarios supported by OWL. This particularly affects the representation of geometry in the ifcOWL ontology, which is automatically generated from the IFC EXPRESS schema using a generic procedure. Because of the difficult mapping of LISTS to an equivalent OWL representation, the resulting ifcOWL representation for geometric data does not allow efficient and intuitive usage of the data.

4.1 Four representation approaches

In this paper, we therefore investigated four different alternative representations that can be used to represent aggregation types (LISTs) in OWL for the specific case of the IFC. From the outline of these four representation approaches, it is clear that a trade-off needs to be made between semantic precision and computational efficiency. The procedures that result in the semantically most precise representations (Procedure 1, 2a, and 2b) also result in lengthy and overly complex representations, yet they are the closest equivalents to the original EXPRESS schema. The procedures that result in computationally more efficient representations (Procedure 3a, 3b, and 4), on the other hand, result in semantically less strict data representations, which could in turn result in errors, flaws and mistakes in the applications relying on that data. Yet, as many have concluded in the geospatial domain, this might be the only option to make the use of RDF graphs and OWL ontologies feasible for the architectural design and construction industry.

The differences between the considered representation alternatives can be better appreciated by reporting some statistics, as shown in Table 7. As indicated in the table, the number of triples and individuals generated drastically decreases when applying the conversion procedures 1 to 4, with a maximum number of triples
for representation method 2b (4+8k triples) and a minimum for representation method 4 (1 triple).

<table>
<thead>
<tr>
<th>Procedure</th>
<th>IfcCartesianPoint instance example (Fragment 4)</th>
<th>Generic ordered aggregation attribute with k items</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>triples</td>
<td>individuals</td>
</tr>
<tr>
<td>1</td>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>2a</td>
<td>24</td>
<td>7</td>
</tr>
<tr>
<td>2b</td>
<td>28</td>
<td>7</td>
</tr>
<tr>
<td>3a</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>3b</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 7: The number of triples and individuals generated when applying the various conversion procedures for the IfcCartesianPoint example and a generic ordered aggregation attribute.

When a defined data type is involved as a type range (e.g. IfcLengthMeasure, IfcInteger, ...), then Procedure 4 is clearly the most efficient in terms of storage size and we recommend to employ it whenever a corresponding WKT term is available (see Table 6). Procedure 3b could be applied when a corresponding WKT term is missing, but it must be stressed that this procedure further requires to specify conventions for the automatic generation of the new object properties, so that also the IFC-SPF to RDF graph (and vice versa) conversion can be automated. Moreover, Procedure 3b can be reasonably applied only to upper bounded aggregation data types that represent a minor case in IFC (see Table 3).

When an entity data type is involved as a type range (e.g. IfcGridAxis, IfcCartesianPoint, ...), then Procedure 2a remains the most viable and general purpose option to stay close to the original EXPRESS schema and guarantee reasoning capabilities. Procedure 2a is better than Procedure 2b in terms of storage size and because of the limitations discussed in Sect. 3.2.2. Procedure 3a is promising, but it faces the same disadvantages mentioned before for Procedure 3b. In particular, the solution based on WKT (i.e. Procedure 4) looks most promising and it will be further tested in future research.

With the WKT serialization, a concept of folding multiple triplets into concise string literals is proposed. We have elaborated WKT in particular as it strikes a balance between human readability, expressiveness, efficiency, geometric fidelity and implementation support. In addition, the semantics with relation to spatial reasoning are well understood. Support for geometrical querying on top of WKT literals is implemented in software, but efficient high performance indexes tend to be limited to geospatial data and point features in particular (Battle and Kolas, 2012). Another approach would be to disregard geometrical information altogether or keep them as pointers to (binary) data streams. The merit of our approach is that we keep the door open to include geometrical information in querying and

4The recently proposed glTF standard employs such a mechanism using a text-based JSON hierarchy paired with a binary stream. https://www.khronos.org/gltf
reasoning processes. By employing a solution entirely within the domain of RDF, no further processing is required to compute subgraphs and facilitate updates. WKT has a binary encoding called WKB which can be used under the hood in triple stores, similar to how an `xsd:double` can be stored as a binary IEEE 754 encoded floating point number, rather than as its string serialization and datatype URI, for performance reasons.

4.2 Modularisation

As indicated in the introduction (Sect. 1.3), a second way in which ifcOWL graphs can be made less complex and smaller in size, is by modularising the ifcOWL ontology. This modularisation track would be a considerable achievement, especially if it aligns with the above work regarding the representation of geometric data. The most interesting (in terms of efficiency and size) of the outlined alternative representations for geometric data are those WKT representations that provide a string serialisation of the absolute coordinates of geometric elements. These representations are entirely different from the existing ifcOWL structure, as was clearly shown in Fragments 17 and 18. In fact, one might actually want to choose the desired representation alternative (ifcOWL or WKT-based ifcOWL) depending on the use case. If appropriate modularisation of ifcOWL would be achieved, this choice would likely be much easier to implement, as both representation alternatives could be offered as separate ontology modules to an end user.

In any case, it will be necessary in future work to design how the new WKT-based OWL axioms can be integrated with the existing ifcOWL ontology and properly considered in automated conversion routines. Such integration can be eased by a modularisation approach to enhance the ifcOWL without needing to revise the core conversion pattern (see Table 1). Although this needs to be further investigated in combination with the actual modularisation of the entire ifcOWL, we tentatively propose a modular structure as shown in Figure 12.

![Figure 12: The tentatively proposed modular structure for ifcOWL.](image)

Five modules can be recognised: a LIST module, an EXPRESS module, an ifcOWL module, an ifcOWL_WKT module, and a WKT module. The LIST, EX-
PRESS, and ifcOWL modules already exist. The WKT module would define the required WKT strings, starting with those WKT strings already defined for the geospatial domain. It could for example represent https://w3id.org/WKT#POINT as a distinct datatype, so that it can replace ifcowl4:wktPoint in Fig. 9. The ifc-OWL_WKT module then imports the WKT module, and it defines those ifcOWL properties that can be improved using WKT-based axioms. For example, the property ifcowl4:coordinates_IfcCartesianPoint (see Fig. 9) could be declared and restricted to only point to a literal of datatype https://w3id.org/WKT#POINT. This modular structure needs to be tested in future research in order to find out to what extent this ifcOWL_WKT module conflicts with requirements represented in the imported ifcOWL ontology. Finally, the modular structure proposed in Fig. 12 would allow an end user to choose:

- to use the full standard ifcOWL representation for geometry (LISTs);
- or to leave out all geometry;
- or to use a semantically less complex WKT representation of geometry;
- or to use an even entirely different representation for geometry.

5 Acknowledgments

The first author gratefully acknowledges the financial support provided by the Special Research Fund (BOF) of Ghent University. The research of the second author has been partially funded by MIUR under the Italian flagship project “La Fabbrica del Futuro”, Subproject 2, research project “Product and Process Co-Evolution Management via Modular Pallet configuration”, and by the EU 7th FP under the grant agreements No: 314156, “Engineering Apps for advanced Manufacturing Engineering”.

References


Franz Baader and Werner Nutt. Basic description logics. In Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, Description Logic Handbook: Theory, Implementation, and


Nicholas Drummond, Alan Rector, Robert Stevens, Georgina Moulton, Matthew Horridge, Hai Wang, and Julian Sedenberg. Putting OWL in order: Patterns for sequences in OWL. In OWL Experiences and Directions (OWLED 2006), 2006.


Information Management Group, University of Manchester. list.owl - OWL research at the University of Manchester, 2015. Available online: http://owl.cs.manchester.ac.uk/?attachment_id=1296 (Last accessed on 2 August 2016).


