W. Terkaj, A. Šojić (2015) Ontology-based representation of IFC EXPRESS rules: An enhancement of the ifcOWL ontology.
Automation in Construction, Volume 57, Pages 188-201, ISSN 0926-5805.
https://doi.org/10.1016/j.autcon.2015.04.010
(https://www.sciencedirect.com/science/article/pii/S0926580515000886)

Ontology-based Representation of IFC EXPRESS rules: an enhancement of the ifcOWL ontology

Walter Terkaj, Aleksandra Sojic

Institute of Industrial Technologies and Automation (ITIA-CNR), Milan, Italy

Abstract

Semantic Web technologies are becoming more and more appealing to the community of the Industry Foundation Classes (IFC) users. The recent actions towards the development of an OWL version of the IFC schema (named ifcOWL) evidence the effort of facing the community request to specify IFC in an ontology language. This paper presents an enrichment of the EXPRESS to OWL conversion patterns with OWL class expressions that specifically capture certain constraints of the IFC standard. The presented class expressions can be used to improve the robustness of ifcOWL and support data integrity, consistency, and applicability across various scenarios of industrial application. In particular, the rules defining the relationships between non-abstract subtypes of `IfcObject` and corresponding subtypes of `IfcTypeObject` are analysed and converted into novel class expressions. The proposed additions to the conversion pattern have been implemented into a software tool and the resulting ontology was tested against a use case to show the benefits of the new solution with respect to a basic ifcOWL. The novel conversion pattern enhances usability of the ifcOWL ontology by enabling its direct instantiation, without necessarily going through the intermediate generation of a STEP physical format that is then converted into an ontology.

Keywords: IFC-OWL conversion, EXPRESS rules, OWL Class Expression, ifcOWL instantiation

1. Introduction

Industry Foundation Classes (IFC) [1] is the accepted standard for Building Information Modelling (BIM). It provides schemas for data management in building and construction, but its applicability extends across numerous other domains that are dealing with industrial data modelling, including factory design, construction, simulation and evaluation of manufacturing processes. The Semantic Web (SW) technologies provide a modelling environment that can deal with heterogeneous data, supporting interoperability across diverse knowledge domains, integration of distributed data and employment of reasoning engines to infer new knowledge automatically [2]. The appealing features of SW motivated work on conversion of the IFC

Email address: `walter.terkaj@itia.cnr.it` (Walter Terkaj, Aleksandra Sojic)

EXPRESS schema [3] into an OWL [4] ontology. In this paper, we take into consideration the existing efforts in the direction of the IFC to OWL conversion. While taking an OWL version of IFC (named *ifcOWL* [5]) as a reference, we propose an enhancement of the conversion pattern in order to optimise its applicability. In particular, the pattern deals with some of the rules defined in the IFC schema. The conversion of these rules have been neglected so far, since the main application scenario addressed in the literature consists in the conversion of IFC files (in the form of STEP physical file [6]) that were generated by IFC-compliant software tools [7]. The conversion of the rules into the OWL version of IFC may enable the direct instantiation of the ontology without going through the generation of STEP files and afterwords checking the consistency of IFC projects, whatever is their origin. Moreover, the enrichment of the ifcOWL ontology allows both to safely use general purpose SW tools and to develop new ifcOWL-based software tools while minimising inconsistencies and erroneous applications of the standard.

In this paper, we address the conversion of the EXPRESS schema [3] representing the IFC standard into an OWL ontology to exploit the features of SW technologies [2]. In particular, an OWL version of IFC (named *ifcOWL*) can be better integrated with other ontologies to support interoperability, while making use of data distribution capabilities; moreover, general purpose reasoning engines can be directly employed to infer new knowledge.

This paper is organized as follows. We first outline the state of the art about the IFC to OWL conversion (Sect.2) and then we briefly present the adopted conversion pattern (Sect.3). Section 4 describes the proposed enhancements to the conversion pattern, whereas its implementation is discussed in Sect.5. Section 6 presents an ontology-based software tool that is able to exploit the results of conversion pattern suggested in the previous section, thus providing additional control for the assurance of the data quality. Finally, we provide an example to demonstrate how the proposed pattern impacts on the results of both closed world assumption (CWA) integrity check and open world assumption (OWA) reasoning (Sect.7).

2. State of the Art

The Industry Foundation classes (IFC) standard by buildingSMART [1] is an object oriented data model provided as an open specification for Building Information Modelling (BIM). IFC aims at supporting data exchange and sharing among the various participants in a building construction or facility management projects. It consists of the data schema, represented as an EXPRESS schema specification [3], and reference data, represented as XML definitions of property and quantity definitions [1]. The IFC model is structured into four layers: Resource layer (i.e. general purpose or low level concepts/objects), Core Layer (where the most abstract concepts of the model are defined), Interoperability Layer defining concepts or objects common to two or more domains, and the Domains/Application Layer [1]. Even if the standard was mainly conceived for the Architecture, Engineering and Construction (AEC) industry domains (e.g. Building Controls, Structural elements, Structural Analysis, etc.), its data structures can be specialised for other industrial domains, such as the manufacturing domain [8].

The emergence of SW technologies led to several initiatives aimed at converting the IFC schema into an ontology language that can provide a semantically rich and platform independent framework to support the integration and interoperability of software tools and exchange of data in a knowledge based system that are human readable and processable by machines. In particular, Beetz and colleagues [9], as well as Krima et al. [10, 11], emphasised the EXPRESS's lack of formal semantics, arguing that a logic-based language such as OWL brings modelling advantages in knowledge representation, semantic data sharing, and reuse of existing ontologies and interoperability with the SW tools.

The first conversion map from EXPRESS to OWL was presented by Schevers and Drogemuller [12], while taking IFC as a reference example and highlighting some of the key issues to be addressed. Beetz et al. [9] proposed a semi-automatic method for converting EXPRESS schemas to OWL files to enhance the applicability and re-usability of the IFC standard. Barbau and colleagues specified the rules for the automated conversion from EXPRESS to OWL within the OntoSTEP initiative [11] and implemented the system as an OWL plug-in for Protégé [13].

Several recent works dealing with the BIM domain have shown how useful SW technologies can be to support the related business processes [14]. Zhang et al. [15] argued that the conversion of IFC into OWL, beside enabling the exploitation of SW technologies for building information models, facilitates the retrieval of information from IFC models. Pauwels et al. [16] showed how Semantic Web Rule Language (SWRL) [17] rules can be exploited to enrich an OWL version of IFC and create a semantic rule checking environment. The use of rules was exemplified by the case of acoustic performance regulations. Similarly, Beach et al. [18] developed a methodology to automatically check the compliance with regulations in the building industry. The authors suggest the development of specific regulation ontologies consisting of SWRL rules that can be linked with a core ontology representing the addressed industrial domain.

Lee et al. [19] proposed to model the work conditions and work items needed during a tiling process by means of ontologies. They made use of the XML version of IFC and reasoning was exploited to infer which are the proper work items according to the work condition.

Terkaj et al. [20] proposed a modular OWL ontology for factory modelling and data sharing between heterogeneous software tools. The ontology, named Virtual Factory Data Model (VFDM), was based on the integration of an OWL version of IFC with other ontologies derived from technical standards (e.g. STEP-NC [21]), or defining basic concepts not included in IFC (e.g. probability distributions), or specializing IFC for the manufacturing domain. The VFDM was exploited to enable interoperability between various software tools, including both commercial (e.g. Plant Simulation by Siemens PLM [22]) and non-commercial (e.g. GIOVE Virtual Factory [23]) applications.

Zhang et al. [24] proposed a construction safety ontology to formalize the safety management knowledge, while considering both construction elements and construction processes. SWRL rules were used to express safety regulations and support the automatic generation of job hazard analysis (JHA) reports.

All the previous cases demonstrate that a recommended and complete ontology version of IFC would be beneficial to ease the integration of data models, support the flow of data and enable the use of ontology-based tools. Indeed, recently Pauwels [5] proposed a conversion pattern that re-elaborates the previous contributions (mainly [9] and [11]) and adds new features, aiming also at producing an officially recommended OWL version of IFC ontology, named ifcOWL. The main conversion criteria were:

(A) keeping the ifcOWL ontology in the OWL DL profile [4] in order to enable reasoning;

(B) enriching the ifcOWL with axioms to match the original EXPRESS schema as closely as possible;

(C) primarily supporting the conversion of IFC STEP physical files into equivalent RDF graphs, without necessarily guaranteeing that an RDF graph can be modelled from scratch by directly instantiating the ifcOWL ontology.

3. IFC to OWL conversion pattern

The previous section mentioned some of the key approaches to the *conversion* of IFC from EXPRESS to OWL that are well documented in the literature (cfr. [12, 9, 11]). This section instead focuses the attention on the most recent development in this research field that was proposed by Pauwels [5]. The key features of the conversion pattern are outlined together with a few examples that are relevant for the analysis carried out in the following sections, especially for the core contribution of Sect. 4. The reference IFC EXPRESS schema is IFC4_ADD1.exp [1].

Table 1 reports the summary of the conversion pattern by Pauwels [5] in terms of EXPRESS statements paired with the corresponding OWL axioms. EXPRESS `ENTITY` is converted into `owl:Class`, i.e., the OWL construct that specifies classes in general, including classes of particulars. The EXPRESS taxonomical relationship `SUPERTYPE OF/SUBTYPE OF`, holding between concepts classified by `ENTITY`, has its OWL equivalent in `rdfs:subClassOf`, which holds between `owl:Class` concepts.

The `ABSTRACT` characterization in EXPRESS rules out direct instantiation of certain entities. This constraint does not have a direct counterpart in OWL. The corresponding structural pattern is obtained by declaring that the subclasses form a partition, where the OWL subclasses of the given class cover the extension of the concept. Analogously, the `ONEOF` construct is obtained by declaring that the OWL subclasses of that class are disjoint.

EXPRESS	**OWL**
Schema	Ontology
Simple data type	OWL class with a restriction on an owl:DatatypeProperty
Defined data type	OWL class
Constructed SELECT data type	OWL class with equivalence to a unionOf collection of OWL classes
Constructed ENUMERATION data type	OWL class with equivalence to a oneOf collection of owl:NamedIndividual items
Entity data type	OWL class
Attribute of entity data type	Functional object property with specified domain and range; 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
INVERSE attribute	object property with a specified owl:inverseOf
DERIVE attribute	N/A
WHERE rule	N/A
FUNCTION	N/A
RULE	N/A

Table 1: Summary of the adopted conversion pattern [5], where *owl:* stands for the namespace 'http://www.w3.org/2002/07/owl#'.

In EXPRESS the properties of the entities are modelled via attributes. By default, an attribute listed in the class definition is mandatory for each element of that class. When this is not so, the attribute is labelled `OPTIONAL`: if a class C_1 has optional attribute A_1, then it is a matter of choice whether A_1 should be instantiated by a particular value. EXPRESS attributes are converted into OWL class restrictions by means of object properties, using a universal quantification (`owl:AllValuesFrom`) and, if necessary, cardinality constraints.

One of the novelties introduced in the proposal by Pauwels consists in the automatic conversion of `INVERSE` attributes that were neglected or not thoroughly addressed in previous literature contributions. However, it can be noted that some EXPRESS statements, in particular `WHERE` rules, have not been converted (see Table 1) since they are considered irrelevant according to the conversion criteria mentioned before.

The case of entity `IfcObject` (defined in Fragment 1 using EXPRESS language) can be used as an example to show how the conversion pattern can be applied. An `IfcObject` is "the generalization of any semantically treated thing or process. Objects are things as they appear - i.e. occurrences. Examples of IfcObject include physically tangible items such as wall, beam or covering, physically existing items such as spaces, or conceptual items such as grids or virtual boundaries" [1].

```
ENTITY IfcObject
ABSTRACT SUPERTYPE OF (ONEOF (IfcActor,IfcControl,IfcGroup,IfcProcess,IfcProduct,IfcResource))
SUBTYPE OF (IfcObjectDefinition);
ObjectType : OPTIONAL IfcLabel;
INVERSE
IsDeclaredBy : SET [0:1] OF IfcRelDefinesByObject FOR RelatedObjects;
Declares : SET [0:?] OF IfcRelDefinesByObject FOR RelatingObject;
IsTypedBy : SET [0:1] OF IfcRelDefinesByType FOR RelatedObjects;
IsDefinedBy : SET [0:?] OF IfcRelDefinesByProperties FOR RelatedObjects;
WHERE
UniquePropertySetNames : IfcUniqueDefinitionNames(IsDefinedBy);
END_ENTITY;
```

Fragment 1: Definition of entity `IfcObject`

The conversion of `IfcObject` into OWL is given by the conjunction of the statements from (1) to (11), whereas statements from (12) to (22) define the object properties that are used in the restrictions. These ontology fragments are defined according to the Manchester Syntax [25] that is mapped to the traditional DL Syntax in Table 2.

In addition, for each subclass of `IfcObject` there will be an axiom specifying that such class is disjoint with the other subclasses. The OWL statements capture the EXPRESS abstract construct because every instance of `IfcObject` must be an instance of one and only one of its subclasses, since statement (2) declares `IfcObject` as a subclass of the union of its disjoint subclasses. For example, it cannot be the case that a particular object (e.g. *ObjectX*) instantiates both a subclass of `IfcGroup` and a subclass of `IfcProcess`.

Some of the object properties are named differently from the corresponding EXPRESS attribute (e.g. `ObjectType`) to guarantee their uniqueness in ifcOWL.

```
Class:  IfcObject                                                                        (1)
    SubClassOf:
        IfcActor or IfcControl or IfcGroup or IfcProcess or IfcProduct or IfcResource,  (2)
        IfcObjectDefinition,                                                             (3)
        ObjectType_of_IfcObject only IfcLabel,                                           (4)
        ObjectType_of_IfcObject max 1 IfcLabel,                                          (5)
        IsDeclaredBy only IfcRelDefinesByObject,                                         (6)
        IsDeclaredBy max 1 IfcRelDefinesByObject,                                        (7)
        Declares_of_IfcObject only IfcRelDefinesByObject,                                (8)
        IsTypedBy only IfcRelDefinesByType,                                              (9)
        IsTypedBy max 1 IfcRelDefinesByType,                                            (10)
    DisjointWith:  IfcTypeObject, IfcContext                                            (11)
```

OWL Constructor	**DL Syntax**	**Manchester OWL Syntax**
intersectionOf	$C \cap D$	C **and** D
unionOf	$C \cup D$	C **or** D
complementOf	$\neg C$	**not** C
oneOf	$\{a\} \cup \{b\}$ ...	{a b ...}
someValuesFrom	$\exists R\, C$	R **some** C
allValuesFrom	$\forall R\, C$	R **only** C
minCardinality	$\geq N\, R$	R **min** N
maxCardinality	$\leq N\, R$	R **max** N
cardinality	$= N\, R$	R **exactly** N
hasValue	$\exists R\, \{a\}$	R **value** a

Table 2: Mapping between the *DL Syntax* and *Manchester OWL Syntax* [25]

```
ObjectProperty:  ObjectType_of_IfcObject                                  (12)
    Characteristics:  Functional                                          (13)
ObjectProperty:  IsDeclaredBy                                             (14)
    Characteristics:  Functional                                          (15)
    InverseOf:  RelatingObject_of_IfcRelDefinesByObject                   (16)
ObjectProperty:  Declares_of_IfcObject                                    (17)
    Characteristics:  Functional                                          (18)
    InverseOf:  RelatingObject_of_IfcRelDefinesByObject                   (19)
ObjectProperty:  IsTypedBy                                                (20)
    Characteristics:  Functional                                          (21)
    InverseOf:  RelatedObjects_of_IfcRelDefinesByType                     (22)
```

The OWL statements (9) and (10) involve the class `IfcRelDefinesByType` standing for the IFC objectified relationship entity that is meant to "define the relationship between an object type (see `IfcTypeObject`) and object occurrences (see `IfcObject`). The `IfcRelDefinesByType` is a 1-to-N relationship as it allows for the assignment of one type information to a single or to many objects. Those objects then share the same object type, and the property sets and properties assigned to the object type" [1]. The entity `IfcTypeObject` in turn "defines the specific information about a type, being common to all occurrences of this type. It refers to the specific level of the well recognized generic - specific - occurrance modeling paradigm. The `IfcTypeObject` gets assigned to the individual object instances (the occurrences) via the `IfcRelDefinesByType` relationship. The object type is represented by a set of property set definitions" [1]. Practically, objectified relationships are used in IFC to separate the properties specific to relationships from the object attributes. This allows the development of a separate subtype tree for the relationship semantics [1].

According to the adopted conversion pattern, the core definition of classes `IfcTypeObject` and `IfcRelDefinesByType` is reported in statements (23)-(28) and statements (29)-(34), respectively. Finally, statements (35)-(44) define the object properties that are needed to formulate the restrictions. The complete definitions of classes and properties can be found in [5].

```
Class:  IfcTypeObject                                                      (23)
    SubClassOf:
        IfcObjectDefinition                                                (24)
        HasPropertySets only IfcPropertySetDefinition,                     (25)
        Types only IfcRelDefinesByType,                                    (26)
        Types max 1 IfcRelDefinesByType                                    (27)
    DisjointWith:  IfcObject, IfcContext                                   (28)

Class: IfcRelDefinesByType                                                 (29)
    SubClassOf:
        IfcRelDefines,                                                     (30)
        RelatingType only IfcTypeObject,                                   (31)
        RelatingType exactly 1 IfcTypeObject,                              (32)
        RelatedObjects_of_IfcRelDefinesByType only IfcObject,              (33)
        RelatedObjects_of_IfcRelDefinesByType min 1 IfcObject              (34)

ObjectProperty:  HasPropertySets                                           (35)
    InverseOf:  DefinesType                                                (36)
ObjectProperty:  Types                                                     (37)
    Characteristics:  Functional                                           (38)
    InverseOf:  RelatingType                                               (39)
ObjectProperty:  RelatingType                                              (40)
    Characteristics:  Functional                                           (41)
    InverseOf:  Types                                                      (42)
ObjectProperty:  RelatedObjects_of_IfcRelDefinesByType                     (43)
    InverseOf:  IsTypedBy                                                  (44)
```

The class `IfcPropertySetDefinition` used in statement (25) plays a key role in IFC together with `IfcObject` and `IfcTypeObject`. Indeed it is “a generalization of all individual property sets that can be assigned to an object or type object. The property set definition can be specified either as *dynamically* *extendable property sets* or as *statically defined property sets*. Property set definitions define information that is shared among multiple instances of objects, either object types (via a direct relationship) or object occurrences (via the objectified relationship `IfcRelDefinesByProperties`)" [1].

Again according to the adopted conversion pattern, the core definition of classes `IfcPropertySetDefinition` and `IfcRelDefinesByProperties` is reported in statements (45)-(49) and statements (50)-(55), respectively, whereas statements (56)-(63) define the object properties that are needed to formulate their restrictions.

```
Class:  IfcPropertySetDefinition                                                        (45)
    SubClassOf:
       IfcPreDefinedPropertySet or IfcPropertySet or IfcQuantitySet,                    (46)
       IfcPropertyDefinition,                                                           (47)
       DefinesType only IfcTypeObject,                                                  (48)
       DefinesOccurrence only IfcRelDefinesByProperties,                                (49)

Class: IfcRelDefinesByProperties                                                        (50)
    SubClassOf:
       IfcRelDefines,                                                                   (51)
       RelatingPropertyDefinition only IfcPropertySetDefinitionSelect,                  (52)
       RelatingPropertyDefinition exactly 1 IfcPropertySetDefinitionSelect,             (53)
       RelatedObjects_of_IfcRelDefinesByProperties only IfcObjectDefinition,            (54)
       RelatedObjects_of_IfcRelDefinesByProperties min 1 IfcObjectDefinition            (55)

ObjectProperty:  DefinesType                                                            (56)
    InverseOf:  HasPropertySets                                                         (57)
ObjectProperty:  DefinesOccurrence                                                      (58)
    InverseOf:  RelatingPropertyDefinition                                              (59)
ObjectProperty:  RelatingPropertyDefinition                                             (60)
    Characteristics:  Functional                                                        (61)
    InverseOf:  DefinesOccurrence                                                       (62)
ObjectProperty:  RelatedObjects_of_IfcRelDefinesByProperties                            (63)
```

Finally, Fig.1 shows the OWL universal quantification restrictions linking `IfcObjectDefinition`, `IfcObject`, `IfcTypeObject`, `IfcPropertySetDefinition`, `IfcRelDefinesByType` and `IfcRelDefinesByProperties`, which are depicted by dashed lines. `IfcPropertySetDefinitionSelect`, used in statements (52) and (53), is defined as equivalent to the union of `IfcPropertySetDefinition` and

`IfcPropertySetDefinitionSet`; for the sake of simplicity, `IfcPropertySetDefinitionSelect` is not included in Fig.1 and is instead replaced by `IfcPropertySetDefinition`.

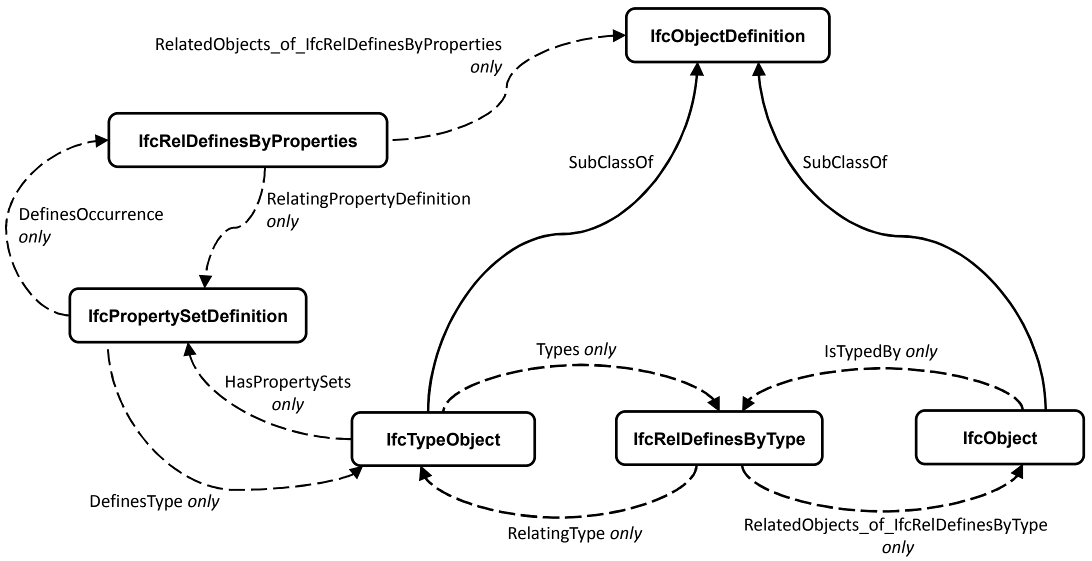


Figure 1: OWL universal quantification restrictions linking `IfcObjectDefinition`, `IfcObject`, `IfcTypeObject`, `IfcPropertySetDefinition`, `IfcRelDefinesByType` and `IfcRelDefinesByProperties`.

4. Conversion of IFC rules into OWL Class Expressions

The conversion pattern presented in the previous section is incomplete because there are still relevant EXPRESS definitions that are not taken in due consideration, as reported in Table 1. In particular, the `WHERE` keyword can be used in the EXPRESS schema to specify rules and complex constraints; Zhao and Liu [26, 27] proposed a mapping between `WHERE` rules and SWRL language, as was already suggested by Beetz et al. [28].

In this section we propose an enhancement of the ifcOWL ontology by partially revising the conversion criteria defined by Pauwels [5]. The criteria *A* and *B* (i.e. keeping ifcOWL in OWL DL profile and enriching the ifcOWL with axioms to match the original EXPRESS schema as closely as possible) are preserved, but criterion *C* (i.e. primarily supporting the conversion of IFC STEP physical files into equivalent RDF files) is excluded and replaced by the following one:

(D) Analysis of rules in the IFC schema and ensuing conversion into the ifcOWL to support the direct instantiation of the ontology and guarantee its consistency, thus recalling the motivations presented in Sect. 1.

If key rules are not properly defined in the ifcOWL ontology, then it is necessary to introduce rule-like constraints at a later time, e.g. during the development of an ontology-based software application. However, this choice leads to at least two important drawbacks:

- The risk of introducing errors and misinterpretations by the software developers.
- The development of rigid (hard-coded) applications that are not able to properly deal with changes in domain ontologies, thus affecting the flexibility of the system and, in turn, the interoperability.

Herein, it is proposed to overcome these problems by enriching the ifcOWL ontology with OWL class expressions[1] representing rules of the IFC schema. Even if the use of class expressions is a standard practice in ontology design, we are going to discuss their significance in the context of IFC conversion pattern, in particular with reference to typing relationships (Sect. 4.1) and pre-defined property sets (Sect. 4.2). We propose the use of OWL class expressions and not SWRL in order to assure compliance with conversion criterion *A*. Indeed, SWRL is not part of OWL and this language has not yet been approved after the submission to the World Wide Web Consortium (W3C).

4.1. Typing relationships

The objectified relationship entity `IfcRelDefinesByType` plays a key role in the IFC schema because it allows to reduce unnecessary data redundancy inside a model, as explained in the definition quoted in Sect.3. The correct use of this typing relationship is guaranteed by `WHERE` rules in the IFC schema that constrain which instances can be linked via `IfcRelDefinesByType`. For example, taking in consideration `IfcWindow` and `IfcWindowType` that are specializations of `IfcObject` and `IfcTypeObject`, respectively, the IFC definition of `IfcWindow` in Fragment 2 shows the `WHERE` rule named `CorrectStyleAssigned`; such rule states that the instances of `IfcWindow` can be related only with an instance of `IfcWindowType`.

[1]Note that Manchester syntax supports the nesting of class constructors and complex class expressions that can be disambiguated by bracketing, see e.g.[25]

```
ENTITY IfcWindow
SUPERTYPE OF (ONEOF (IfcWindowStandardCase))
SUBTYPE OF (IfcBuildingElement);
OverallHeight : OPTIONAL IfcPositiveLengthMeasure;
OverallWidth : OPTIONAL IfcPositiveLengthMeasure;
PredefinedType : OPTIONAL IfcWindowTypeEnum;
PartitioningType : OPTIONAL IfcWindowTypePartitioningEnum;
UserDefinedPartitioningType : OPTIONAL IfcLabel;
WHERE
CorrectStyleAssigned : (SIZEOF(IsTypedBy) = 0) OR
   ('IFC4.IFCWINDOWTYPE' IN TYPEOF(SELF\IfcObject.IsTypedBy[1].RelatingType));
END_ENTITY;
```

Fragment 2: Definition of entity `IfcWindow`

Based only on statements (9), (10), (26), (27), and (31)-(34), the ifcOWL ontology allows to create links between an instance of a non-abstract `IfcObject` subclass and an instance of any non-abstract `IfcTypeObject` subclass. Thus the conversion criterion *D* is not met. Without restricting relations, it is possible to (erroneously) relate, for example, an instance of `IfcDoor` with an instance of `IfcWindowType`, implying that a door is modelled after a type of window. Therefore, it is relevant to convert the `WHERE` rules defining the correct typing relationship into the OWL version of IFC, otherwise this lack would jeopardize the consistency in the output of an ontology-based software application. As anticipated, it is possible to meet this goal by means of OWL class expressions.

Statement (65) shows how the `WHERE` rule `CorrectStyleAssigned` of `IfcWindow` (see Fragment 2) can be converted into a class expression that consists in a more articulated restriction compared to the restrictions used to convert the entity attributes according to the basic conversion pattern by Pauwels [5].

```
Class:  IfcWindow                                                        (64)
     SubClassOf:  IsTypedBy only (RelatingType only IfcWindowType)       (65)
```

Moreover, we suggest to further constrain the link between non-abstract subclasses of `IfcTypeObject` and corresponding subclasses of `IfcObject` by further characterizing also the `IfcTypeObject` subclass, even if this type of rule is not defined in the IFC schema. Indeed, with reference to the `IfcWindowType` example, Fragment 3 shows that no `WHERE` rule states that it can type only instances of `IfcWindow`.

```
ENTITY IfcWindowType
SUBTYPE OF (IfcBuildingElementType);
PredefinedType : IfcWindowTypeEnum;
PartitioningType : IfcWindowTypePartitioningEnum;
ParameterTakesPrecedence : OPTIONAL IfcBoolean;
UserDefinedPartitioningType : OPTIONAL IfcLabel;
WHERE
CorrectPredefinedType : (PredefinedType <> IfcWindowTypeEnum.USERDEFINED) OR ((PredefinedType =
   IfcWindowTypeEnum.USERDEFINED) AND EXISTS(SELF\IfcElementType.ElementType));
END_ENTITY;
```

Fragment 3: Definition of entity `IfcWindowType`

Statement (67) can be added to guarantee that an instance of `IfcWindowType` can be related only with an instance of `IfcWindow` via a chain of properties including `Types` and `RelatedObjects_of_IfcRelDefinesByType`. Without this additional restriction it would be possible to relate an instance of `IfcWindowType` with an instance of any non-abstract subclass of `IfcObject` that is not characterized by a restriction similar to (65); this is the case of class `IfcBuilding` that is not paired with a corresponding subclass of `IfcTypeObject`.

```
Class:  IfcWindowType                                                              (66)
     SubClassOf:  Types only (RelatedObjects_of_IfcRelDefinesByType only IfcWindow)      (67)
```

Furthermore, it would be beneficial to extend also the EXPRESS specification of IFC, that could be reviewed by adding a specific `WHERE` rule to all non-abstract sub-entities of `IfcTypeObject`. Statement (68) shows an example of this new `WHERE` rule named `CorrectOccurrenceAssigned` for the case of `IfcWindowType`.

```
(SIZEOF(Types) = 0) OR ('IFC4.IFCWINDOW' IN TYPEOF(SELF \IfcObject.Types[1].RelatedObjects)) (68)
```

It must be stressed that the definition of class expressions like (65) and (67) was possible thanks to the conversion of `INVERSE` attributes like `Types` and `IsTypedBy` in the ifcOWL by Pauwels [5].

Figure 2 shows a specialization of Fig.1 and provides a graphical representation of the restrictions (65) and (67) in bold dashed lines. An ontology-based application can also exploit these restrictions to correctly instantiate the class `IfcWindowType` and the class `IfcWindow` as well as to relate them via an instance of the class `IfcRelDefinesByType`.

Overall, the OWL capability to capture class expressions, as sketched in the examples above, represents an advantage over the IFC schema, since it is not needed to split the definitions into separate attributes and rules. More importantly, by rephrasing the system in a logical language, one can make explicit the pattern of reasoning over classes that impacts the inferences about particulars. The following section examines

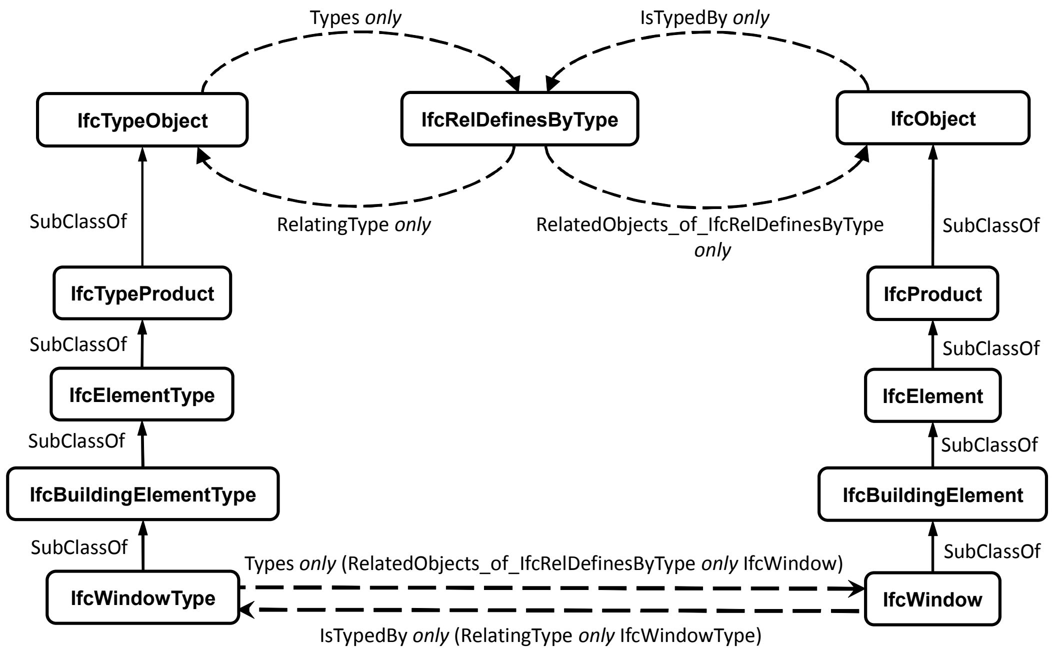


Figure 2: OWL restrictions linking `IfcWindowType` and `IfcWindow`.

the impact of class expressions to the representation and reasoning that involves instances, as Sect.7 will discuss.

4.2. Pre-defined property sets

As already mentioned in Sect.3, the IFC schema allows to attach property sets defined as instances of `IfcPropertySetDefinition` to instances of `IfcTypeObject` and `IfcObject`. `IfcPreDefinedPropertySet` and `IfcPropertySet` are subclasses of `IfcPropertySetDefinition` that can be used to represent *statically defined property sets* (i.e. pre-defined property sets) and *dynamically extendible property sets*, respectively. The IFC schema includes only a few subclasses of `IfcPreDefinedPropertySet` (e.g. `IfcWindowPanelProperties` can be used to characterize a window), whereas most of the property sets included in the IFC specification are not defined in the IFC EXPRESS schema, but are attached as XML files (e.g. Pset_WindowCommon.xml for common properties of a window) that can be used to lead the instantiation of `IfcPropertySet`. Herein only `IfcPreDefinedPropertySet` and its subclasses are considered because the attention is focused on the conversion from EXPRESS to OWL, but future developments could address also the conversion of the XML property sets to OWL.

The IFC schema defines the correct use of a pre-defined property set by means of `WHERE` rules that constrain which specialization of `IfcObject` and `IfcTypeObject` can be linked to the property set. For example, the IFC schema definition of `IfcWindowPanelProperties` in Fragment 4 shows the `WHERE` rule named ApplicableToType. This rule states that an instance of IfcWindowPanelProperties must be related with at least one instance of IfcWindowType or IfcWindowStyle via the attribute DefinesType. The rule can be converted into the OWL class expressions shown in statements (70) and (71).

```
ENTITY IfcWindowPanelProperties
SUBTYPE OF (IfcPreDefinedPropertySet);
OperationType : IfcWindowPanelOperationEnum;
PanelPosition : IfcWindowPanelPositionEnum;
FrameDepth : OPTIONAL IfcPositiveLengthMeasure;
FrameThickness : OPTIONAL IfcPositiveLengthMeasure;
ShapeAspectStyle : OPTIONAL IfcShapeAspect;
WHERE
ApplicableToType : (EXISTS(SELF\IfcPropertySetDefinition.DefinesType[1])) AND
   ( ('IFC4.IFCWINDOWTYPE' IN TYPEOF(SELF\IfcPropertySetDefinition.DefinesType[1]))
     OR ('IFC4.IFCWINDOWSTYLE' IN TYPEOF(SELF\IfcPropertySetDefinition.DefinesType[1])) );
END_ENTITY;
```

Fragment 4: Definition of entity IfcWindowPanelProperties

```
Class:  IfcWindowPanelProperties                                          (69)
     SubClassOf:
        DefinesType only (IfcWindowStyle or IfcWindowType),               (70)
        DefinesType min 1 (IfcWindowStyle or IfcWindowType),              (71)
```

Moreover, even if not defined in the IFC schema, also the class expression (72) could be added to guarantee that an instance of IfcWindowPanelProperties can be related only with an instance of IfcWindow (and no other subclass of IfcObject) via a chain of properties including DefinesOccurrence and RelatedObjects_of_IfcRelDefinesByProperties.

```
SubClassOf:  DefinesOccurrence only (RelatedObjects_of_IfcRelDefinesByProperties only IfcWindow) (72)
```

Further specifications of class expressions do not include superclasses of IfcWindowType and IfcWindow because of maintainability concerns. Indeed, other class expressions would need to be updated whenever a new property set for IfcWindowType and/or IfcWindow is created in a domain ontology that extends ifcOWL, thus creating a conflict between the desired extendibility and stability of a recommended ifcOWL.

5. Implementation of the conversion patterns

The conversion patterns presented in the previous section have been implemented into a C++ program that makes use of programming libraries built on the Redland RDF libraries [29]. This program implements a set of general purpose routines for the automatic generation of OWL class expressions that are similar in structure to statements (65), (67), (70)-(72) defined for `IfcWindow`, `IfcWindowType` and `IfcWindowPanelProperties`.

The input data needed to run the C++ program consists in a list of paired classes being non-abstract subclasses of `IfcObject`, `IfcTypeObject`, or `IfcPreDefinedPropertySet`. Having adopted a notation where `OccX`, `TypeY`, and `PsetZ` stand for a generic non-abstract subclass of `IfcObject`, `IfcTypeObject`, and `IfcPreDefinedPropertySet`, respectively, the input list can be obtained by parsing the `WHERE` rules in the IFC EXPRESS schema and applying the following rules:

- a pair `OccX` - `TypeY` is added to the list if a `WHERE` rule containing the expression (73) is found in the definition of `OccX`;
- a pair `PsetZ` - `TypeY` is added to the list if a `WHERE` rule containing the expression (74) is found in the definition of `PsetZ`;
- a pair `PsetZ` - `OccX` is added to the list if a `WHERE` rule containing the expression (75) is found in the definition of `PsetZ`.

```
(TypeY IN TYPEOF(SELF\IfcObject.IsTypedBy[1].RelatingType))                          (73)
(TypeY IN TYPEOF(SELF\IfcPropertySetDefinition.DefinesType[1]))                      (74)
(OccX IN TYPEOF(SELF\IfcPropertySetDefinition.DefinesOccurrence[1].RelatedObjects))  (75)
```

Alternatively, the user can directly provide an input list, if the class expressions will be used to enrich a domain ontology that extends ifcOWL. The routines of the program provide the following functionalities:

- for each pair `OccX` - `TypeY`, the statements (76) and (77) are added to the definition of class `OccX` and `TypeY`, respectively.
- for each pair `PsetZ` - `TypeY`, the statement (78) is added to the definition of class `TypeY`.
- for each pair `PsetZ` - `OccX`, the statement (79) is added to the definition of class `OccX`.

```
SubClassOf:  IsTypedBy only (RelatingType only TypeY)                                  (76)
SubClassOf:  Types only (RelatedObjects_of_IfcRelDefinesByType only OccX)              (77)
SubClassOf:  DefinesType only TypeY                                                    (78)
SubClassOf:  DefinesOccurrence only (RelatedObjects_of_IfcRelDefinesByProperties only OccX) (79)
```

The program allows to generate and store the additional class expressions in an ontology module that is separated from the ontology containing the basic definitions of the classes. This is particularly useful if the program has to enrich the ifcOWL ontology, because in this case the additional class expressions can be stored in a new ontology module. This module, named *ifcOWL_rules*, imports ifcOWL and exploits the advantages of data distribution and linked-data paradigm enabled by the SW approach. When an Abox ontology must be created, the user decides to directly import *ifcOWL_rules* or ifcOWL ontology according to his/her needs.

6. Ontology-based Software Tool exploiting additional OWL class expressions

This section presents OntoGUI, an ontology-based software tool that can be employed in the management and instantiation of Abox ontology modules. This tool mainly aims at supporting:

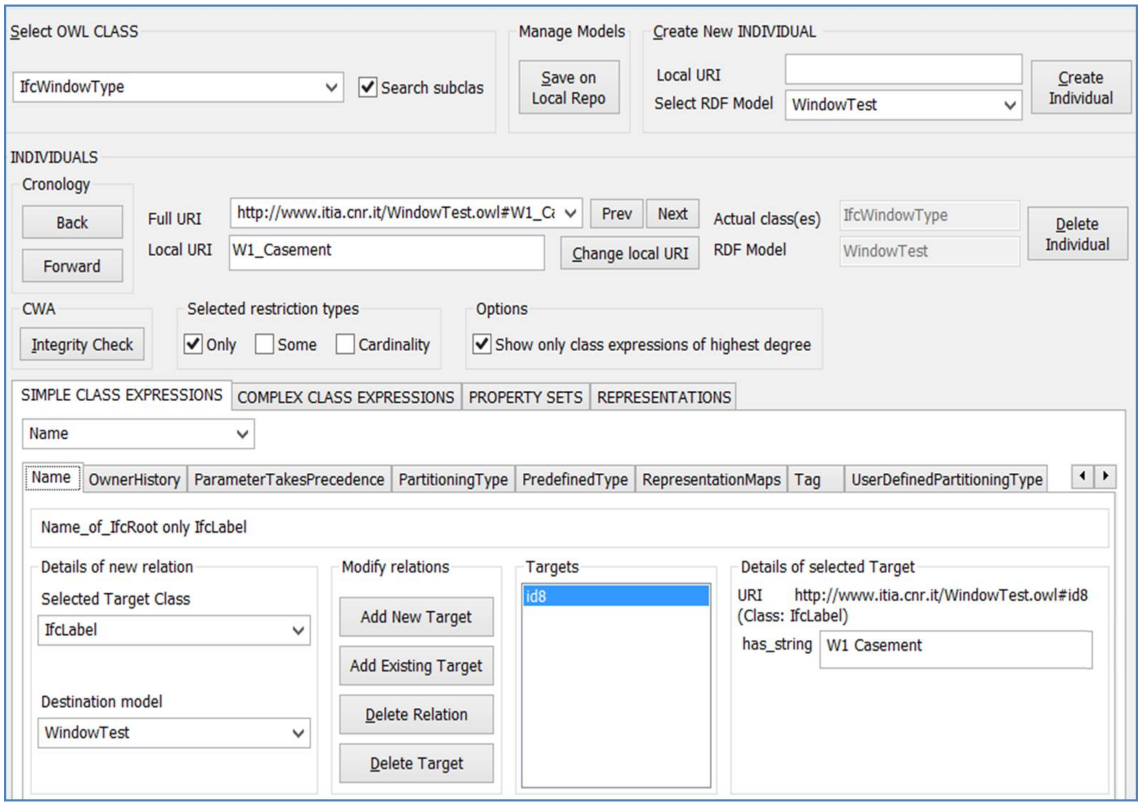
- The fast evaluation of ontology Tbox by concurrently instantiating a corresponding Abox, following a Test-driven development approach.
- The generation of RDF data sets to be used as input for other ontology-based applications, without needing to develop complex customized graphical user interfaces or data converters.

OntoGUI was developed in the C++ language making use of wxWidgets Cross-Platform GUI Library and another programming library named *VfConnectorLib*, based on the Redland RDF libraries [29]. The *VfConnectorLib* library includes the following key features:

- A mapping between class definitions in the ifcOWL ontology and C++ classes and methods.
- Support for different solutions of data repository that can be implemented as file-based systems, relational databases or native triple stores [30].

The user interface of OntoGUI consists of a control panel for the creation and loading of ontologies from a file-based repository or other more performing RDF stores. The control panel provides also access to a set of functional modules, including *Individuals Manager* that is a general purpose tool for the exploration, generation and characterization of OWL individuals. The user interface of *Individuals Manager* (Fig.3) is dynamically reconfigured every time a different OWL class is selected since the tool is able to extract information related to the following axioms defined in the Tbox ontology:

- Equivalent classes, both defined as single classes or union of classes.
- Subclasses, both defined as single classes or union of classes.
- Restrictions of any degree if they involve universal quantifier (i.e. `owl:allValuesFrom`), existential quantifier (i.e. `owl:someValuesFrom`), or cardinality constraints (i.e. `owl:qualifiedCardinality`, `owl:maxQualifiedCardinality`, `owl:minQualifiedCardinality`).



Select OWL CLASS: IfcWindowType [✓ Search subclas]

Local URI:

Select RDF Model: WindowTest

Full URI: http://www.itia.cnr.it/WindowTest.owl#W1_Ca

Local URI: W1_Casement

Actual class(es): IfcWindowType

RDF Model: WindowTest

Selected restriction types: [✓ Only] Some Cardinality

Options: [✓ Show only class expressions of highest degree]

Name

Name_of_IfcRoot only IfcLabel

Selected Target Class: IfcLabel

Destination model: WindowTest

Targets: id8

URI http://www.itia.cnr.it/WindowTest.owl#id8 (Class: IfcLabel)

has_string: W1 Casement

Figure 3: A view of Individuals Manager tool in OntoGUI

The characterisation of the classes with their restrictions is exploited by the *Individuals Manager* tool both for exploring and generating relations between individuals and also for checking the consistency of the individuals. After loading an ontology, *Individuals Manager* allows to select any OWL class defined in the corresponding Tbox and with respect to this class the following operations can be performed:

- Generation of a new individual belonging to the selected class.
- Listing of the individuals belonging to the selected class (and its subclasses if this option is flagged) and selection of one of them.
- Navigation through all the possible relations involving the selected individual according to the restrictions defined in the Tbox for the class(es) it belongs to. For each restriction it is possible to visualise the target individuals or literals that can be found at the end of the property chain defined by the restriction itself. The target individuals can be in turn explored by double clinking.
- Creation of new relations between the selected individual and already existing individuals or individuals/literals generated on demand.
- Integrity check of the selected individual by analysing one by one all the restrictions associated with

the classes the individual belongs to and checking if the restrictions are violated when interpreted as Integrity Constraints according to the Closed World Assumption (see Sect. 7.3).

In particular, after querying which are the target individuals or literals at the end of the property chain specified by the analysed restriction, the *integrity check* is able to identify the following violations:

- At least one target does not belong to the class/datatype specified by a restriction using a universal quantifier.
- There is no target belonging to the class/datatype specified by a restriction using an existential quantifier.
- The number of targets belonging to the class/datatype specified by a restriction using a qualified cardinality does not respect the cardinality constraint.
- A functional property is used to link a specific individual with more than one individual/literal.

The basic functionalities of *Individuals Manager* can be provided for any ontology Tbox without requiring specific adaptations. However, *Individuals Manager* can also be enhanced by introducing customizations depending on a specific Tbox. In particular, taking into consideration the ifcOWL ontology, the following functionalities have been added:

- Customized window for the quick definition of an object placement.
- Automatic replication of the aggregation structure defined for an individual of type object class if this individual is used to generate an instance of the corresponding occurrence object class that is identified thanks to a class expression like (77).
- Specific interface for the management of pre-defined property sets. If a non-abstract subclass of `IfcObject` (e.g. `OccX`) or `IfcTypeObject` (e.g. `TypeY`) is selected, then the non-abstract subclasses of `IfcPreDefinedPropertySet` are explored searching for class expressions like (78) and (79). If a match is found in a subclass of `IfcPreDefinedPropertySet` (e.g. `PsetZ`), then this class is added to the list of selectable pre-defined property sets.
- Visualization of inherited property sets. If an instance of a non-abstract subclass of `IfcObject` is typed by an instance of the corresponding type object class according to a class expression like (76), then the property sets of the typing instance are inherited. The inheritance is disabled if the property set is overridden by the object occurrence instance making use of `IfcRelDefinesByProperties` (see Fig.1).
- Visualization of the proper unit of measurement when numeric values are explored according to definitions in the IFC project or library.

7. Benefits of additional OWL class expressions under CWA and OWA

7.1. Test case

The test case consists of an excerpt taken from an IFC model available on an open access repository [31] and it involves the instantiation of IFC entities `IfcWindowType`, `IfcWindowPanelProperties`, `IfcWindowLiningProperties`, and `IfcWindow`, thus recalling the examples presented in Sect.4. Table 3 reports the characteristics of the `IfcWindowType` instance named `W1_Casement`; two property sets are attached to the definition of the window type, namely instances of `IfcWindowPanelProperties` and `IfcWindowLiningProperties`.

IFC Entity	Attribute	Value
IfcWindowType	Name	W1 Casement
	Construction Type	NOTDEFINED
	Operation Type	SINGLE_PANEL
	Parameter Takes Precedence	0
IfcWindowLiningProperties	Lining Depth	50
	Lining Thickness	50
	Transom Thickness	0
	Mullion Thickness	0
	First Transom Offset	0
	Second Transom Offset	0
	First Mullion Offset	0
	Second Mullion Offset	0
IfcWindowPanelProperties	Operation Type	NOTDEFINED
	Panel Position	MIDDLE
	Frame Depth	50
	Frame Thickness	50

Table 3: Definition of `W1_Casement` as an instance of `IfcWindowType` [31]

The original IFC model required a limited re-factoring since it was based on a previous version of IFC, namely IFC2x3. For example, `IfcWindowType` is used in place of `IfcWindowStyle` because it is deprecated in IFC4_ADD1.

The excerpt of the IFC model was converted into an ontology module, named `WindowTest`, that instantiates a fragment of the ifcOWL ontology. The instantiation can be carried out using any OWL editor (e.g. Protégé [13], OntoGUI, etc.) or an automatic converter of IFC files into RDF graphs (e.g. [32]). Furthermore, individuals `W01` and `D01` were generated by instantiating classes `IfcWindow` and `IfcDoor`, respectively. Finally, a typing relationship with individual `W1_Casement` of class `IfcWindowType` was added to both `W01` and `D01`. The typing relation of `D01` is a modelling error that was introduced on purpose

485 to check the relevance of the additional class expressions in the following subsections. The graphical representation of the key individuals defined in the ontology module `WindowTest` is shown in Fig.4, where the individuals are represented by boxes with grey background.

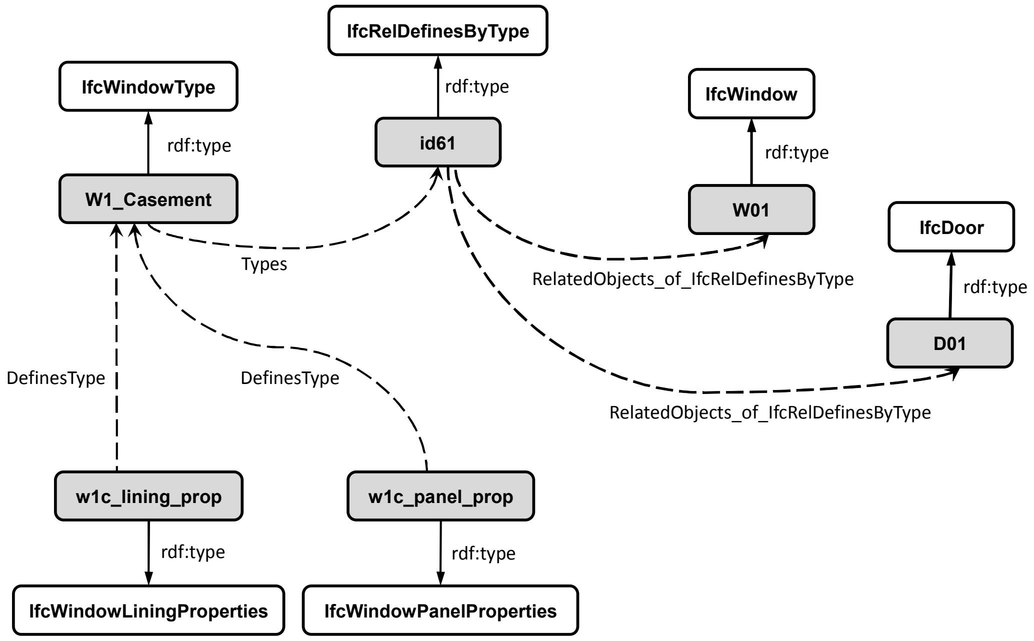


Figure 4: Graphical representation of the key individuals within the ontology module WindowTest, where *rdf:* stands for the namespace 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'.

It is possible to design two configurations of the test case (`WinTest1` and `WinTest2`), even using the same ontology module `WindowTest`, as shown also in Fig.5:

490 `WinTest1.` The ontology `WindowTest` imports only ifcOWL.

`WinTest2.` The ontology `WindowTest` imports ifcOWL and also ifcOWL_rules that contains the definition of additional class expressions like (76)-(79).

The test case configurations[2] will be analysed in two possible scenarios; the former is defined by the Open World Assumption (OWA, see Sect.7.2), whereas the latter considers the context of the Closed World 495 Assumption (CWA, see Sect.7.3). CWA and OWA are two distinct approaches to knowledge representation [33].

[2]`WinTest1` and `WinTest2` are available at `https://ontohub.org/repositories/test-ifc`, under tab 'File Browser'

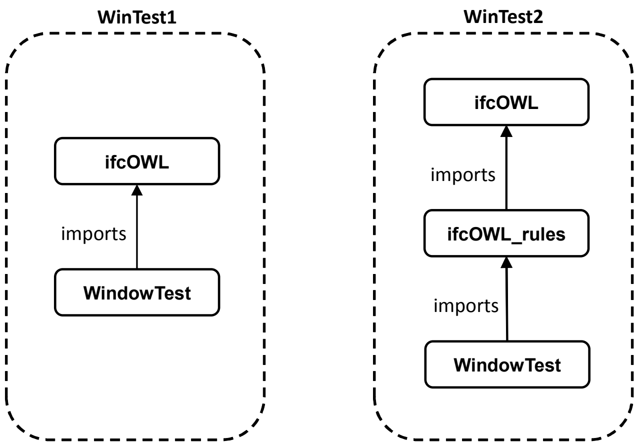


Figure 5: Graphical representation of the ontology modules in test case configurations `WinTest1` and `WinTest2`.

7.2. Reasoning under OWA

The Open World Assumption includes the concept of incomplete information. For instance, if a fact (or an atomic sentence) is not present in the database it is not *false* by default, but rather it is considered under the assumption that it is *unknown* if the sentence is true or false. An advantage of using OWA is that it allows expansion of knowledge in the ABox through the reasoning and inference over the existing facts in the ABox and the axioms defined in the TBox. SW and OWL support the implementation of OWA, that can drive to the discovery of new facts that have not been explicitly declared.

Test case configurations `WinTest1` and `WinTest2` can be considered in the context of OWA, which leaves open the possibility that some facts are unknown. In this case `WinTest1` passes the parsing test performed via reasoners that are available in Protégé (FaCT++, Pellet, HermiT), without capturing any inconsistency. Unlike `WinTest1`, the class expressions available in `WinTest2` function as the closure axioms, explicitly restricting the intended meaning of the typing relationship between the class-pairs `IfcWindowType` and `IfcWindow`, and `IfcDoorType` and `IfcDoor`. Thus, an inappropriate assignment of the typing relationship leads to inconsistency of `WinTest2`.

One of the possible explanations of the inconsistency is given by considering together statements (67), (80), (81), (82), (83), and (84).

`D01 Type IfcDoor` (80)

`W1_Casement Types id61` (81)

`id61 RelatedObjects_of_IfcRelDefinesByType D01` (82)

`W1_Casement Type IfcWindowType` (83)

`IfcDoor DisjointWith IfcWindow` (84)

The individual D01 of class IfcDoor (80) is involved in a typing relationship (81-82) with individual W1_Casement of class IfcWindowType (83). According to statement (67), an individual of class IfcWindowType can be in a typing relationship only with individuals of class IfcWindow, therefore it can be inferred that individual D01 belongs to class IfcWindow as well. However, IfcWindow and IfcDoor are disjoint classes (84) and their intersection is an empty set, thus leading to inconsistency because individual D01 cannot be a member of both classes.

The presence of additional class expressions in WinTest2 provides a relevant advantage over WinTest1 because it enables the detection of inconsistencies caused by modelling errors. This is particularly useful during reasoning, because it reduces the risk of generating and using the fact inferred from a poorly built ontology. For examples, the works by Pauwels et al. [16], Beach et al. [18], and Lee et al. [19] mentioned in Sect.2 could benefit from the inclusion of the proposed class expressions that can increase the quality and confidence of inferences obtained via reasoning.

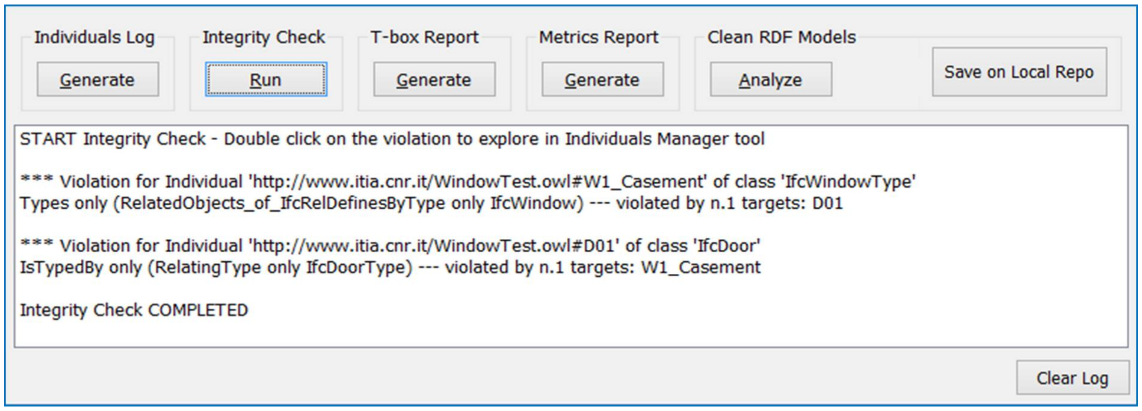
7.3. Integrity Constraint Validation under CWA

The Closed World Assumption states that *unless an atomic sentence is known to be true, it can be assumed to be false* (see [33], p. 210). This assumption is frequently used in the database practice as it allows to act as if a Knowledge Base (KB) represents complete knowledge. In addition, the CWA provides a useful approach that simplifies the representational scope by capturing only a small fraction of the large number of sentences that can be declared in a KB, i.e. the fraction of the atomic sentences that are *true by default*. It also assumes that any unmentioned atomic sentence is *false*. Moreover, the typical KB that we consider employs the CWA with domain closure, including the additional assumption that no object exists apart from the named constants (see [33], p. 214).

The test case configurations WinTest1 and WinTest2 can be re-evaluated in the context of CWA under the domain closure where everything that exists must be explicitly declared and only the declared sentences are considered to be *true*. The fact that instance W1_Casement is a member of the class IfcWindowType, and the fact that D01 is a member of the class IfcDoor are stated to be true. If an integrity check based on CWA[3] is run over WinTest1, then no violation is detected and its interpretation under CWA is satisfiable. On the other hand, certain statements about the typing relationships are recognised as impermissible, i.e. *false*, within WinTest2 (see Fig.6). The typing relation between W1_Casement and D01 is *false* in WinTest2 because an individual of class IfcWindowType is allowed to have a typing relationship only

[3]The integrity check under CWA can be run with different tools, e.g. Pellet Integrity Constraint Validator (http://clarkparsia.com/pellet/icv/) and OntoGUI (see Sect. 6).

with individuals of class `IfcWindow`, whereas `D01` belongs to class `IfcDoor`, thus leading to a violation of integrity check under CWA.



```
START Integrity Check - Double click on the violation to explore in Individuals Manager tool

*** Violation for Individual 'http://www.itia.cnr.it/WindowTest.owl#W1_Casement' of class 'IfcWindowType'
Types only (RelatedObjects_of_IfcRelDefinesByType only IfcWindow) --- violated by n.1 targets: D01

*** Violation for Individual 'http://www.itia.cnr.it/WindowTest.owl#D01' of class 'IfcDoor'
IsTypedBy only (RelatingType only IfcDoorType) --- violated by n.1 targets: W1_Casement

Integrity Check COMPLETED
```

Figure 6: Explanation of integrity constraint violation for `WinTest2` under CWA as generated by OntoGUI tool (see Sect. 6).

Also in the case of CWA the presence of additional class expressions in `WinTest2` is beneficial because it provides tighter constraints to be used to detect unintended statements and to lead the generation of new instances and relationships.

8. Concluding Remarks

Motivated by the relevance of the IFC standard and the growing interest towards SW technologies, we provided an enhancement of the state-of-the-art IFC to OWL conversion patterns by adding class expressions that explicate the links between IFC object occurrence, object type and pre-defined property sets.

The analysis of a test case under CWA and OWA scenarios has demonstrated that the additional class expressions allow the detection of impermissible relationships between individuals, thus showing how different conversion strategies can lead to more accurate (and even more reliable) ontological representations of the IFC standard. These strategies rely on the structural constructs of the ontological language and are thus suitable to verify the logical consistency and coherence of the captured information.

The relevance of the new class expressions has been discussed also in the scope of software development by addressing the case of the ontology-based software tool OntoGUI. The class expressions allow this tool to correctly create relations between object types, object occurrences, and pre-defined property sets while avoiding hard-coded implementations and enhancing the consistency and integrity of data on the level of application. The generation of class expressions like (76)-(79) can be applied to the core ifcOWL ontology, but also to domain ontologies that extend ifcOWL. Therefore, any ontology-based application that is able to handle such class expressions will enhance its flexibility and re-usability.

It remains to be determined to what extent OWL class expressions can be employed to convert further explicit and implicit rules in the IFC schema. Further developments are foreseen to address the following open issues:

- IFC property and quantity sets have not yet been considered in the literature about EXPRESS to OWL conversion because they are not included in the EXPRESS schema, but attached to the IFC specification as XML files. However, the XML definition can be converted to OWL by creating new pre-defined property sets. In this case, class expressions like (78) and (79) would complete the characterization of the property sets.
- `WHERE` rules that constrain the feasible values of EXPRESS defined data types (e.g. `IfcPositiveInteger` and `IfcTextAlignment`) can be converted to class expressions consisting in restrictions on OWL datatype properties (`owl:DatatypeProperty`). This conversion pattern would be general purpose and not limited to the IFC schema.
- Further `WHERE` rules declarations in the IFC schema may be successfully converted into customized class expressions. Attention will be paid to declarations involving objectified relationship classes (i.e. subclasses of `IfcRelationship`).
- Implicit rules in the IFC standard could be made explicit in its OWL version. For example, class expressions can be exploited to enforce that a process (see `IfcProcess`) can be nested only by another process.

Ontology can help to shed light into other aspects of the IFC standard as well. On the one hand, it helps to understand classes and relationships, raising the awareness of the modeller toward a coherent use of these elements [34]. On the other hand, it provides guidelines to choose the appropriate representation schema depending on the information to be modelled. It remains to verify whether this further analysis may lead to improve the conversion results here discussed as suggested by previous experiences in applied ontology, e.g. [35, 36, 37].

9. Acknowledgments

This research has been partially funded by the European Union 7th FP (FP7/20072013) under the grant agreement No: 314156, Engineering Apps for advanced Manufacturing Engineering (Apps4aME) and by MIUR under the Italian flagship project Fabbrica del Futuro, Subproject 2, research project Product and Process Co-Evolution Management via Modular Pallet configuration (PRO2EVO) and Smart Manufacturing 2020 of the Cluster Tecnologico Nazionale Fabbrica Intelligente.

References

[1] T. Liebich, Y. Adachi, J. Forester, J. Hyvarinen, S. Richter, T. Chipman, M. Weise, J. Wix, Industry Foundation Classes official release - Introduction, online.
URL http://www.buildingsmart-tech.org/ifc/IFC4/final/html/introduction.htm

[2] T. Berners-Lee, J. Hendler, O. Lassila, et al., The semantic web, Scientific American 284 (5) (2001) 28–37.

[3] International Organization for Standardization, ISO 10303-11:2004, Industrial Automation Systems and Integration – Product Data Representation and Exchange – Part 11: Description Methods: The EXPRESS Language Reference Manual, online.
URL http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38047

[4] P. Hitzler, M. Krtzsch, B. Parsia, P. F. Patel-Schneider, S. Rudolph, OWL 2 Web Ontology Language Primer (Second Edition), online.
URL http://www.w3.org/TR/owl2-primer/

[5] P. Pauwels, ifcOWL: the EXPRESS to OWL conversion pattern, online (2015).
URL http://www.w3.org/community/lbd/ifcowl/

[6] International Organization for Standardization, ISO 10303-21:2002, Industrial Automation Systems and Integration – Product Data Representation and Exchange – Part 21: Implementation Methods: Clear Text Encoding of the Exchange Structure, online.
URL http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38047

[7] buildingSMART, Certified Software, online.
URL http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38047

[8] W. Terkaj, M. Urgo, Ontology-based Modeling of Production Systems for Design and Performance Evaluation, in: 12th IEEE International Conference on Industrial Informatics, Porto Alegre, 2014, 2014.

[9] J. Beetz, J. Van Leeuwen, B. De Vries, IfcOWL: A case of transforming EXPRESS schemas into ontologies, Artificial Intelligence for Engineering Design, Analysis and Manufacturing 23 (01) (2009) 89–101.

[10] S. Krima, R. Barbau, X. Fiorentini, R. Sudarsan, R. Sriram, OntoSTEP: OWL-DL ontology for STEP, National Institue of Standards and Technology, NISTIR 7561.

[11] R. Barbau, S. Krima, S. Rachuri, A. Narayanan, X. Fiorentini, S. Foufou, R. D. Sriram, OntoSTEP: Enriching product model data using ontologies, Computer-Aided Design 44 (6) (2012) 575–590.

[12] H. Schevers, R. Drogemuller, Converting the Industry Foundation Classes to the Web Ontology Language, in: Semantics, Knowledge and Grid, 2005. SKG'05. First International Conference on, IEEE, 2005, pp. 73–73.

[13] H. Knublauch, R. W. Fergerson, N. F. Noy, M. A. Musen, The Protégé OWL plugin: An Open Development Environment for Semantic Web Applications, in: The Semantic Web–ISWC 2004, Springer, 2004, pp. 229–243.

[14] P. Pauwels, R. De Meyer, J. Van Campenhout, Interoperability for the Design and Construction Industry through

Semantic Web Technology, in: T. Declerck, M. Granitzer, M. Grzegorzek, M. Romanelli, S. Rger, M. Sintek (Eds.), Semantic Multimedia, Vol. 6725 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2011, pp. 143–158.

[15] L. Zhang, R. R. Issa, Development of IFC-based construction industry ontology for information retrieval from IFC models, in: Proceedings of the 2011 EG-ICE Workshop, University of Twente, The Netherlands, July, 2011, pp. 6–8.

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

[17] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, et al., SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Member submission 21 (2004) 79.

[18] T. Beach, Y. Rezgui, H. Li, T. Kasim, A rule-based semantic approach for automated regulatory compliance in the construction sector, Expert Systems with Applications 42 (12) (2015) 5219 – 5231.

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

[20] W. Terkaj, G. Pedrielli, M. Sacco, Virtual Factory Data Model, in: Proceedings of the Workshop on Ontology and Semantic Web for Manufacturing, CEUR Workshop Proceedings, Vol. 886, 2012, pp. 29–43.

[21] International Organization for Standardization, ISO 14649-1:2003, Industrial automation systems and integration – Physical device control – Data model for computerized numerical controllers – Part 1: Overview and fundamental principles, online.
URL `http://www.iso.org/iso/catalogue_detail.htm?csnumber=34743`

[22] B. Kádár, W. Terkaj, M. Sacco, Semantic virtual factory supporting interoperable modelling and evaluation of production systems, CIRP Annals-Manufacturing Technology 62 (1) (2013) 443–446.

[23] M. Colledani, G. Pedrielli, W. Terkaj, M. Urgo, Integrated Virtual Platform for Manufacturing Systems Design, Procedia CIRP 7 (2013) 425 – 430.

[24] S. Zhang, F. Boukamp, J. Teizer, Ontology-based semantic modeling of construction safety knowledge: Towards automated safety planning for job hazard analysis (JHA), Automation in Construction 52 (2015) 29 – 41.

[25] M. Horridge, N. Drummond, J. Goodwin, A. L. Rector, R. Stevens, H. Wang, The Manchester OWL Syntax, in: OWLed, Vol. 216, 2006.

[26] W. Zhao, J. Liu, OWL/SWRL representation methodology for EXPRESS-driven product information model: Part I. Implementation methodology, Computers in Industry 59 (6) (2008) 580 – 589.

[27] W. Zhao, J. Liu, OWL/SWRL representation methodology for EXPRESS-driven product information model: Part II: Practice, Computers in Industry 59 (6) (2008) 590 – 600.

[28] J. Beetz, J. van Leeuwen, B. de Vries, An ontology web language notation of the industry foundation classes, in: Proceedings of the 22nd CIB W78 Conference on Information Technology in Construction, Vol. 2006, 2005.

[29] D. Beckett, Redland RDF Libraries, Online.
URL `http://librdf.org/`

[30] G. Modoni, M. Sacco, W. Terkaj, A survey of RDF store solutions, in: International ICE Conference on Engineering, Technology and Innovation (ICE), 2014, pp. 1–7.

[31] Open IFC Model Repository, Iai: 6-01window_brep_ac_1, online.
URL `http://openifcmodel.cs.auckland.ac.nz/Model/Details/199`

[32] P. Pauwels, D. Van Deursen, IFC-to-RDF: adaptation, aggregation and enrichment, in: First International Workshop on Linked Data in Architecture and Construction, Abstracts, 2012, pp. 1–3.

[33] R. Brachman, H. Levesque, Knowledge representation and reasoning, Elsevier, 2004.

[34] S. Borgo, E. M. Sanfilippo, A. Sojic, W. Terkaj, Ontological Analysis and Engineering Standards: an initial study of IFC, in: V. Ebrahimipour, S. Yacout (Eds.), Ontology Modeling in Physical Asset Integrity Management, Springer, 2015.

[35] N. Guarino, C. A. Welty, An overview of OntoClean, in: Handbook on ontologies, Springer, 2009, pp. 201–220.

[36] M. Grüninger, Using the PSL ontology, in: Handbook on Ontologies, Springer, 2009, pp. 423–443.

[37] M. West, Developing high quality data models, Elsevier, 2011.

W. Terkaj, A. Šojić (2015) Ontology-based representation of IFC EXPRESS rules: An enhancement of the ifcOWL ontology.
Automation in Construction, Volume 57, Pages 188-201, ISSN 0926-5805.
https://doi.org/10.1016/j.autcon.2015.04.010
(https://www.sciencedirect.com/science/article/pii/S0926580515000886)