Optimizing IFC-structured Data Graph for Code Compliance Checking

Schatz Y., Domer B.
University of Applied Sciences and Arts Western Switzerland, Switzerland
yohann.schatz@hesge.ch

Abstract. Construction code compliance checking requires applying specific computer-interpretable rules on datasets. A proposed solution is to represent IFC data as an RDF graph and perform rule-checking using a rule engine. However, the generated graph has a complicated structure since it follows the IFC data model. Consequently, the definition of compliance rules can be challenging, and rules are sensitive to variations of input graphs structure. A methodology is proposed to optimize graphs by giving them a predefined or "standardized" structure. A case study shows that optimization allows the formulation of more straightforward and easier-to-write compliance rules, applicable to all standardized graphs regardless of the initially used BIM authoring tool. In addition, graph size is significantly reduced.

1. Introduction

Automated rule checking is the process by which software evaluates objects and properties of a digital model against predefined rules (Eastman et al., 2009). In the context of automated code compliance checking, formal computer-interpretable rules are used to validate that the design meets the requirements of a construction standard (e.g., verify if a curved road segment has a sufficient radius considering vehicle speed).

Such a process replaces the long and error-prone manual plan review (Eastman et al., 2009) and optimizes projects in terms of construction time, quality and cost. The task can be handled by a BIM authoring tool or dedicated software (Eastman et al., 2009).

Existing rule-based programs, such as Solibri Model Checker (Solibri, 2021), allow validating models from various sources when data is communicated via an interoperable format. In the AEC domain, the neutral model representation is conveyed by the Industry Foundation Classes (IFC) (buildingSMART, 2021).

The downside of most commercial tools is that they hide checking routines from the user (Burggräf et al., 2021). In addition, the development of custom rules may be limited (Beach et al., 2015; Preidel and Borrmann, 2015). Since code compliance checking requires the system to support multiple different and highly specific rulesets, more flexible solutions are needed.

A promising approach (Pauwels et al., 2011) is based on semantic web technologies. First, IFC data is transformed into a knowledge graph. Second, rulesets are applied to it. In such a data model, also called "RDF graph" (Resource Description Framework) (W3C, 2014), IFC entities, as well as their properties and relationships (i.e., the business data) are expressed as statements called "triples".

IFC-to-RDF (Pauwels, 2017) generates an RDF graph from an IFC-STEP Physical File (SPF). As the service maps every entity to an IfcOWL class and every attribute to an IfcOWL property (Pauwels and Van Deursen, 2012), the RDF graph follows the main structure of IFC (Pauwels and Roxin, 2016).

However, IFC exports are heterogeneous, depending on the software used (Belsky et al., 2016). As a consequence, no unique graph structure for a given model exists. This requires providing software-specific compliance rulesets.

https://doi.org/10.7146/aul.455.c208

In addition, output graphs carry information that is not needed for code compliance checking. This applies particularly to some *IfcResource* elements and instances of *IfcPropertyDefinition* and *IfcRelationship* that do not provide any semantic richness and even complicate access to other relevant data (e.g., a single typed value). Inversely, the limitations of IFC exporters may cause a loss of information required for checking, since they do not integrate program-specific object properties.

Graph optimization, i.e., cleaning (removing unnecessary data) and simplifying its structure to transfer it into a targeted shape (standardized graph), will lead to an easier compliance rule formulation, a better handling of differences in IFC exports, and finally, a maintainable code checking system.

Solutions for graph simplification, such as selective information removal (Fahad et al., 2018; Pauwels and Roxin, 2016) and the addition of straightforward constructs between data (Beach et al., 2015; Bouzidi et al., 2012; de Farias et al., 2015; Pauwels et al., 2017, 2011; Zhang et al., 2018) have been proposed.

This paper aims to develop and complete these principles and integrate them into a coherent workflow, in the context of code compliance checking of IFC models based on a graph representation of data.

2. Methodology

The methodology (Figure 1) has been developed in the context of code compliance checking applied to road infrastructure projects. It combines a set of linked processes, identified by a color code:

- : construction code analysis, leading to a structured representation of construction and regulation vocabularies (ontology).
- : data graph analysis, where information to be added or removed is identified.
- : definition of rulesets to optimize/standardize the data graph.
- : definition of rulesets to check data compliance with construction codes.

Applying the methodology involves four distinct competencies, represented in Figure 1:

1. BIM modeling and data flow management: create the model, configure the IFC export to correspond to information requirements, initiate the IFC-STEP file conversion to RDF graph, run the graph optimization process (based on optimization rules) and launch the rule-checking process (based on compliance rules).
2. Domain knowledge: expertise in construction codes, design and execution of construction. Ability to build a structured representation of domain-specific concepts and formulate compliance rules in natural language.
3. Data science: proficiency in knowledge graph processing using semantic web technologies and automated reasoning systems.
4. IFC: strong knowledge of the IFC schema (including the latest IFC 4.3 version) and related MVDs.

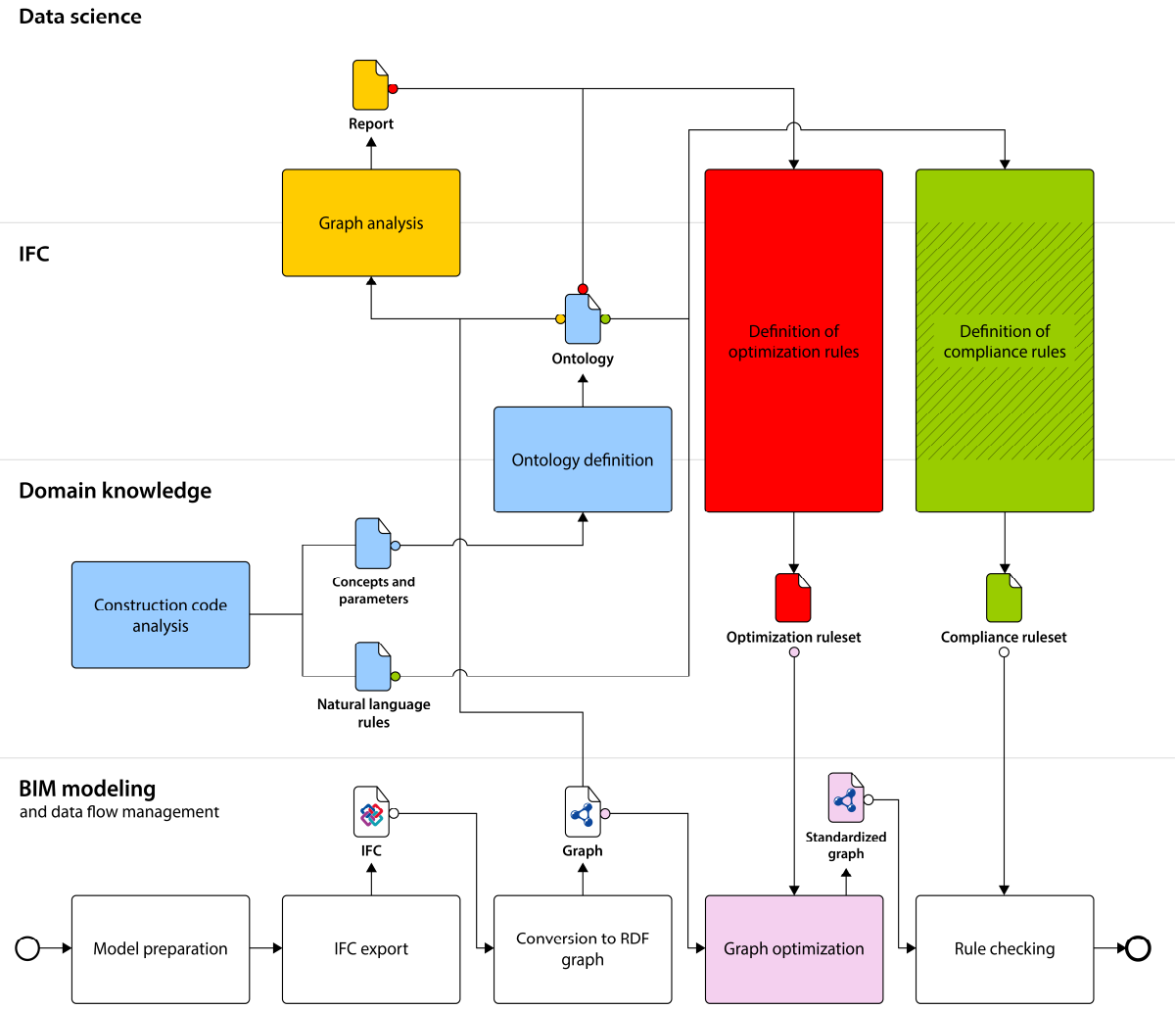

Figure 1: Methodology for optimizing IFC-RDF graphs.

Combining the above-mentioned competencies allows to optimize the graph generated from the initial BIM model. This results in a standardized graph specifically designed for compliance rule checking (represented in pink in Figure 1).

2.1 Blue - Ontology definition

An ontology is a representation of concepts from a given domain, linked by relationships (Martin et al., 2021). Construction code analysis aims to define a single ontology combining domain and regulation concepts. An excerpt of such an ontology for road infrastructure projects is shown in Figure 2.

Classes correspond to physical and virtual components of the building or infrastructure to be modeled (e.g., road, road segment, carriageway, island, etc.). Data literals correspond to single typed values (e.g., a string, an integer, a boolean, etc.). Object properties relate to object-to-object relationships, while datatype properties assign data values to objects (W3C, 2012).

In this context, datatype properties relate to parameters with values constrained by construction codes.

For example, in the sentence "If the road is of type "A", then the design speed is 100 km/h.", *road type* and *design speed* are implicit parameters related to a *road* concept.

All parameters concerned by code compliance checking are added to the ontology as datatype properties, and the associated concepts are integrated as classes (Figure 2).

Classes and properties are labeled with an IRI (Internationalized Resource Identifier), which is a unique sequence of characters. An IRI comprises a namespace prefix (e.g., "*ont*") and the resource name, separated by a colon.

It is assumed that:

- A class IRI has the form *ont:ClassName*.
- An object property IRI has the form *ont:objectPropertyName*.
- A datatype property IRI has the form *ont:datatypePropertyName*.

As the ontology describes the desired structure of data after the optimization process, it is strongly recommended to ensure that information present in IFC data can be mapped to defined domain classes and properties. Ideally, defining the ontology is a joint effort of domain and IFC experts (Figure 1).

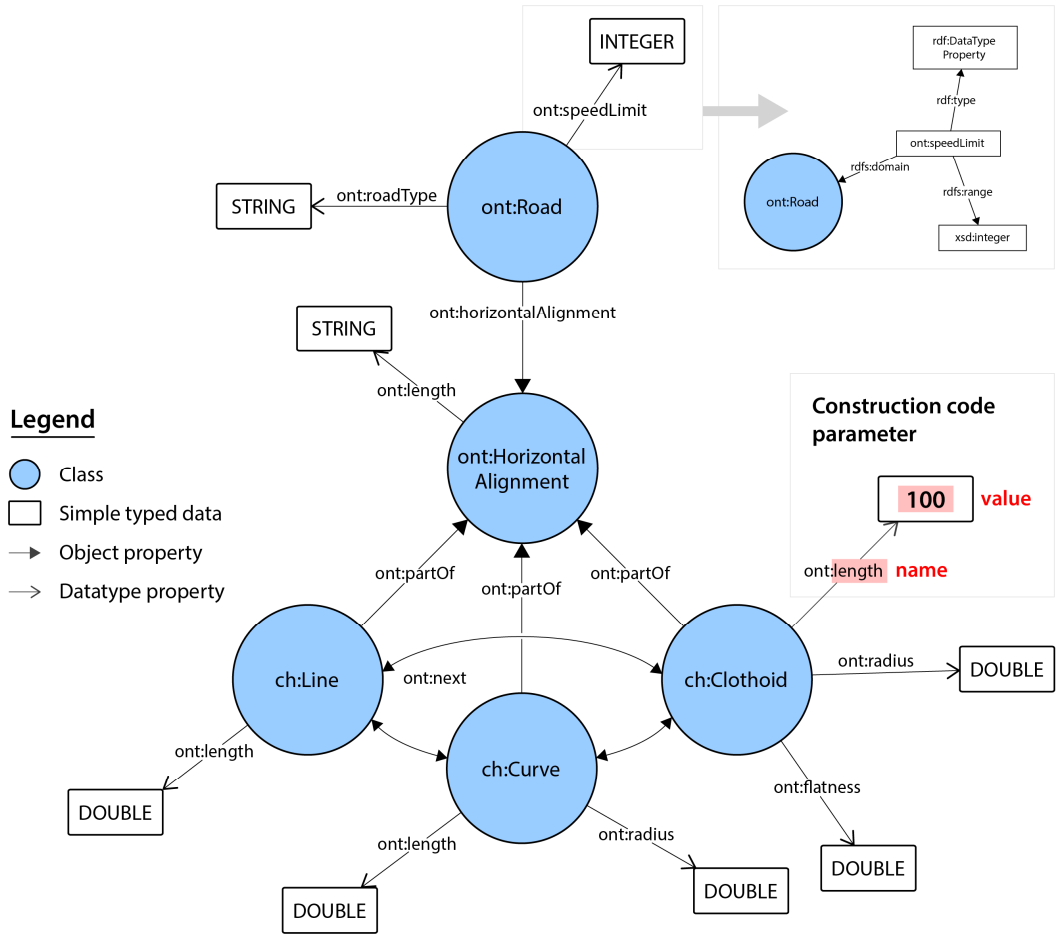

Figure 2: Excerpt of an ontology for road design.

2.2 Orange - Graph analysis, identification of relevant and missing data

The RDF graph is generated from the IFC-SPF file (using IFC-to-RDF). Then, its structure and content are analyzed.

The analysis aims to answer the following questions:

- Which instances (nodes) and relationships (edges), often leading to cumbersome constructs, should be removed from the graph to keep only what is needed for code compliance checking? In other words, what information is outside the scope of the ontology?
- Conversely, which required information is missing and should be provided by other means (e.g., deduced from data)?
- Which constructs should be added so that the graph has the desired structure?
- Finally, which approach to be employed?

Pauwels and Zhang (Pauwels and Zhang, 2015) presented SPARQL query engines and rule engines with dedicated rule languages as solutions for code compliance checking. These technologies are also suitable for graph optimization.

SPARQL (W3C, 2013) is a protocol and a language to query RDF data. SPARQL queries can have different purposes: retrieving information, adding/removing statements, or constructing a new graph. It supports conjunction, disjunction, negation, and aggregation through dedicated functions.

A rule engine (or inference engine) is a tool that produces new information from facts (in this case, all triple statements) and rules (conditional instructions). There are many different rule engines and rule languages (Rattanasawad et al., 2013), which can be selected according to:

- The inference method of the rule engine (iterative, forward chaining, backward chaining, hybrid, etc.).
- The expressiveness, extensibility, and functionality of the rule language.
- The flexibility and openness of the related framework/environment.

2.3 Red - Optimization ruleset definition

Standardizing graph data implies six steps, each one applying different rules:

1. Map IFC instances to ontology classes (Figure 3, a).
2. Create direct constructs (links, object properties from the ontology) between instances (Figure 3, b).
3. Create direct constructs (datatype properties from the ontology) between instances and data typed literals (Figure 3, c).
4. Compute missing parameter values (e.g., the total length of the road) and integrate them (as literals) into the graph with dedicated constructs (Figure 3, d) (optional).
5. Infer missing instances with their corresponding constructs (Figure 3, e) (optional).
6. Delete all undesired information (Figure 3, f) (optional).

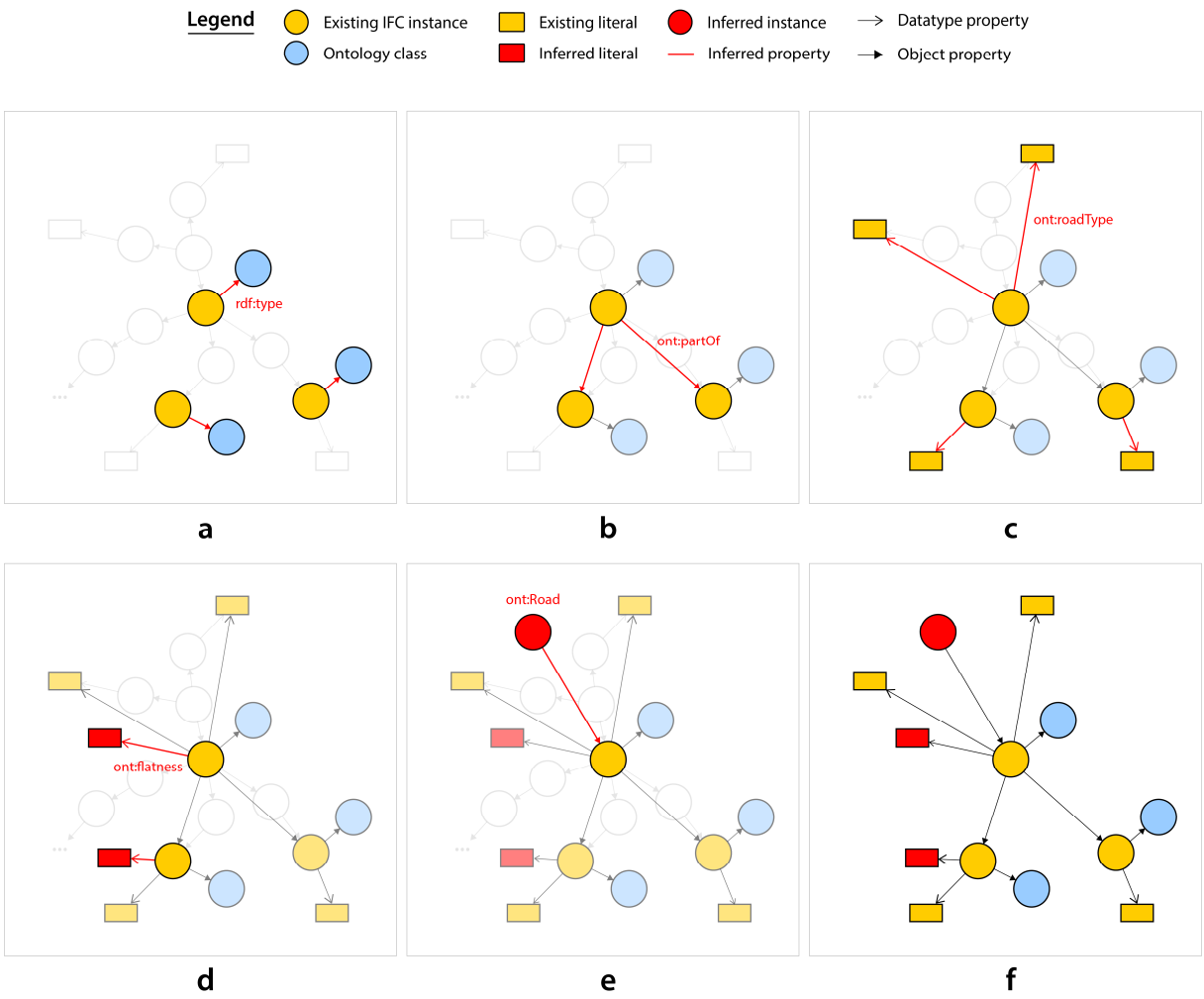

Figure 3: Operations to optimize and standardize IFC-RDF graphs.

Identification of correspondence between graph instances and ontology classes (Figure 3, a) is based on:

- The IFC class of the instance: *if O is an instance of Ifc(...), then O is an instance of C, where C is a class from the ontology* (e.g., IfcRoad → ch:Road).
- The type, property set, or classification associated with the instance (e.g., *IfcRoadPart* instance has type "Road section" → ch:RoadSection).

Note that computing parameter values (Figure 3, d) may require specific functions that are not supported by all rule languages. The solution for data processing must therefore be carefully chosen.

3. Case study

The procedure presented in Figure 1 is applied to check the compliance of a road infrastructure model with the construction code extract given in Figure 4. The data graph will be standardized beforehand, following the methodology described in Section 2. Figure 2 shows the desired data structure.

For roads of type RGD and RP, a curve preceded by a straight section shall have a minimum radius as defined in **Table 16**.

Straight section length (m)	Minimum radius (m)
≤ 100	100
100 < L ≤ 500	200
> 500	300

Table 16. Minimum curve radius

Figure 4: Excerpt of a road design standard.

Concepts related to the excerpt above are "road", "straight section", and "curved section". "Road" corresponds to *ont:Road* class (Figure 2). "Straight section" and "curved section" each describe a geometrical component of the road's horizontal alignment and will be mapped to *ont:Line* and *ont:Curve* classes, respectively. Required properties for the checking are "road type" (*ont:roadType*), "straight section length" (*ont:length*), and "radius" (*ont:radius*) (Figure 2).

A road section, composed of straight segments and curves, is modeled in Autodesk Civil 3D. The IFC model provides required geometric information such as segments length and radius. Road type values are defined in custom property sets.

The model is exported in IFC 4.1 (since Civil 3D does not yet support release 4.3) and converted with IFC-to-RDF. At this stage, the graph has about 690,000 triples (file size: 50.3 MB, Turtle syntax).

Apache Jena is used to perform both tasks: optimization and compliance checking. Jena is an "all-in-one" framework for processing RDF graphs, including a SPARQL query engine (Jena ARQ) and a general-purpose rule engine with a dedicated rule language (The Apache Software Foundation, 2021). Additional functions are implemented in the Jena rule language to enhance its expressivity and make it more functional.

Graph analysis leads to the following conclusions:

- Only instances of the *IfcAlignment* class, combined with their associated property sets and geometric information, contain data indicated as relevant by the ontology and have to be preserved.
- Objectified relationships and other undesired information such as *IfcPropertyDefinition* concepts, instances without semantics (*IfcBuildingElementProxy*), and unnecessary resources can be removed from the graph. This is not mandatory but recommended when the graph is used exclusively for code compliance checking to improve its readability and reduce data quantity.
- Direct constructs must be created, according to the ontology.
- Properties such as "flatness", which may be required for another checking, are missing and should be computed from existing data.

Then, optimization rules (Jena rules and SPARQL CONSTRUCT queries) are formulated:

```
# Step 1: Class mapping (Jena rules)

[optimization_01: (?i rdf:type ifc:IfcAlignment2DHorizontal) -> (?i rdf:type
ont:HorizontalAlignment) ]

...
```

```
# Step 6: Undesired information removal (SPARQL CONSTRUCT query)
CONSTRUCT {?i ?j ?k} WHERE
{
	SELECT ?i ?j ?k WHERE {
		?i rdf:type ?type
		FILTER (
			?type = ont:Road ||
			?type = ont:HorizontalAlignment ||
			...
				)
		?i ?j ?k
	}
	GROUP BY ?i ?j ?k
}
```

Once the rules were applied, the graph size dropped to about 4000 triples (file size: 0.3 MB). Data representation has been modified to comply with the ontology. Compliance rules are shorter and more intelligible since they are only composed of meaningful and straightforward triple statements based on ontology's vocabulary:

```
# Checking compliance with Figure 4 (Jena rule)
[compliance_with_Figure_4: (?line rdf:type ont:Line), (?curve rdf:typeont:Curve), (
?alg rdf:type ont:HorizontalAlignment), (?road rdf:type ont:Road),(?line ont:partOf
?alg), (?line ont:next ?curve), (?road ont:horizontalAlignment ?alg), (?line ont:le
ngth ?length), (?curve ont:radius ?radius),(?road ont:roadType ?type), filter(?type
= RGD|RP), check((?length<=100&radius>100)|?length=]100;500]&radius>200)|?length>50
0&radius>300), ?result) -> print(?result) ]

--> PASS
```

Without optimization, the same rule (compliance_with_Figure_4) would have had a far more complex and lengthy content and could not have been applied without a prior inference rule:

```
[prior_inference: (?algh ifc:segments_IfcAlignment2DHorizontal ?list1), (?list1 lis
t:hasNext ?list2) -> (?algh ifc:segments_IfcAlignment2DHorizontal ?list2) ]

# Checking compliance with Figure 4 (Jena rule)
[compliance_with_Figure_4: (?line rdf:type ifc:IfcLineSegment2D), (?hseg1 ifc:curve
Geometry_IfcAlignment2DHorizontalSegment ?line), (?list1 list:hasContents ?hseg1),
(?list1 list:hasNext ?list2), (?list2 list:hasContents ?hs2), (?hseg2 ifc:curveGeom
etry_IfcAlignment2DHorizontalSegment ?curve), (?curve rdf:type ifc:IfcCircularArcSe
gment2D), (?algh ifc:segments_IfcAlignment2DHorizontal ?list1), (?algh ifc:segments
_IfcAlignment2DHorizontal ?list2), (?algc ifc:horizontal_IfcAlignmentCurve ?algh),
(?alg ifc:axis_IfcLinearPositioningElement ?algc), (?alg rdf:type ifc:IfcAlignment)
, (?rdbp ifc:relatedObjects_IfcRelDefinesByProperties ?alg), (?rdbp ifc:relatingPro
pertyDefinition_IfcRelDefinesByProperties ?pset), (?pset ifc:name_IfcRoot ?label1),
(?label1 express:hasString ?psetname), filter(psetname="Pset_Road"), (?pset ifc:has
Properties_IfcPropertySet ?prop), (?prop ifc:name_IfcProperty ?label2), (?label2 ex
press:hasString ?propname), filter(propname="roadType"), (?prop ifc:nominalValue_If
cPropertySingleValue ?label3), (?label3 express:hasString ?roadType), filter(roadTy
pe=RGD||RP), (?line ifc:segmentLength_IfcCurveSegment2D ?pos1), (?pos1 express:hasD
ouble ?length), (?curve ifc:radius_IfcCircularArcSegment2D ?pos2), (?pos2 express:h
asDouble ?radius), check((?length<=100&radius>100)|?length=]100;500]&radius>200)|?l
ength>500&radius>300), ?result) -> print(?result) ]

--> PASS
```

4. Conclusion and future work

Code compliance checking based on graphs is a promising approach since graph data structures can easily be transformed to represent information in the desired way. It is proposed to optimize (standardize) IFC-RDF graphs before using them in a checking system. Standardization leads to a manageable formulation of rules.

Results of a case study showed that the graph size is considerably reduced in addition to simplifying and clarifying the content of formal compliance rules. In case the resulting standardized graph has to be stored (for example, in a triplestore) for further compliance checking, this allows significant storage space savings.

Moreover, the advantage of giving the data graph a predefined structure is that compliance rules become independent of the initially employed software and can be reused for all previously standardized graphs. Consequently, maintenance efforts are equally distributed among domain and software experts.

Without standardization, compliance rules must be modified when building codes or input data structure change. With standardization, compliance rules are adjusted only when regulations change. As indicated, formulating optimization rules requires multiple competencies, implicitly a collaboration between several experts.

The next stage is to enhance and further automate the process of graph simplification and standardization by using graph theory and advanced techniques such as machine learning. In the same way, generating compliance rules directly from paper documents, using methods such as NLP (Natural Language Processing), is a subject of interest.

Acknowledgment

This research is supported by Innosuisse in the framework of the Innovation project 34406.1 IP-ENG: Digital method for efficient analysis and benchmarking of road infrastructure projects "digiMABS".

References

Beach, T.H., Rezgui, Y., Li, H., Kasim, T. (2015). A rule-based semantic approach for automated regulatory compliance in the construction sector. Expert Syst. Appl. 42, pp. 5219–5231. DOI: https://doi.org/10.1016/j.eswa.2015.02.029

Belsky, M., Sacks, R., Brilakis, I. (2016). Semantic Enrichment for Building Information Modeling. Comput.-Aided Civ. Infrastruct. Eng. 31, pp. 261–274. DOI: https://doi.org/10.1111/mice.12128

Bouzidi, K.R., Fies, B., Faron-Zucker, C., Zarli, A., Thanh, N.L. (2012). Semantic Web Approach to Ease Regulation Compliance Checking in Construction Industry. Future Internet 4, pp. 830–851. DOI: https://doi.org/10.3390/fi4030830

buildingSMART (2021). IFC Formats - buildingSMART Technical. Available at: https://technical.buildingsmart.org/standards/ifc/ifc-formats/, accessed December 2021.

Burggräf, P., Dannapfel, M., Ebade-Esfahani, M., Scheidler, F. (2021). Creation of an expert system for design validation in BIM-based factory design through automatic checking of semantic information. Procedia CIRP 99, pp. 3–8. DOI: https://doi.org/10.1016/j.procir.2021.03.012

de Farias, T.M., Roxin, A., Nicolle, C. (2015). IfcWoD, Semantically Adapting IFC Model Relations into OWL Properties. DOI: https://doi.org/10.48550/ARXIV.1511.03897

Eastman, C., Lee, Jae-min, Jeong, Y., Lee, Jin-kook (2009). Automatic rule-based checking of building designs. Autom. Constr. 18, pp. 1011–1033. DOI: https://doi.org/10.1016/j.autcon.2009.07.002

Fahad, M., Bus, N., Fies, B. (2018). Semantic topological querying for compliance checking, in: Karlshøj, J., Scherer, R. (Eds.), EWork and EBusiness in Architecture, Engineering and Construction. CRC Press.

Martin, S., Szekely, B., Allemand, D. (2021). The Rise of the Knowledge Graph. O'Reilly Media Inc.

Pauwels, P. (2017). GitHub page of IFCtoRDF. GitHub. Available at: https://github.com/pipauwel/IFCtoRDF, accessed November 2021.

Pauwels, P., de Farias, T.M., Zhang, C., Roxin, A., Beetz, J., De Roo, J., Nicolle, C. (2017). A performance benchmark over semantic rule checking approaches in construction industry. Adv. Eng. Inform. 33, pp. 68–88. DOI: https://doi.org/10.1016/j.aei.2017.05.001

Pauwels, P., Roxin, A. (2016). SimpleBIM : from full ifcOWL graphs to simplified building graphs. Ework Ebusiness Archit. Eng. Constr. pp. 11–18.

Pauwels, P., Van Deursen, D. (2012). IFC/RDF: Adaptation, Aggregation and Enrichment.

Pauwels, P., Van Deursen, D., Verstraeten, R., De Roo, J., De Meyer, R., Van de Walle, R., Van Campenhout, J. (2011). A semantic rule checking environment for building performance checking. Autom. Constr. 20, pp. 506–518. DOI: https://doi.org/10.1016/j.autcon.2010.11.017

Pauwels, P., Zhang, S. (2015). Semantic Rule-checking for Regulation Compliance Checking: An Overview of Strategies and Approaches.

Preidel, C., Borrmann, A. (2015). Automated Code Compliance Checking Based on a Visual Language and Building Information Modeling. Presented at the 32nd International Symposium on Automation and Robotics in Construction, Oulu, Finland. DOI: https://doi.org/10.22260/ISARC2015/0033

Rattanasawad, T., Saikaew, K.R., Buranarach, M., Supnithi, T. (2013). A review and comparison of rule languages and rule-based inference engines for the Semantic Web, in: 2013 International Computer Science and Engineering Conference (ICSEC). Presented at the 2013 International Computer Science and Engineering Conference (ICSEC), IEEE, Nakorn Pathom, Thailand, pp. pp. 1–6. DOI: https://doi.org/10.1109/ICSEC.2013.6694743

Solibri (2021). Webpage of Solibri. Available at: https://www.solibri.com/, accessed November 2021.

The Apache Software Foundation (2021). Webpage of Jena inference support. Available at: https://jena.apache.org/documentation/inference/#extensions, accessed November 2021.

W3C (2014). RDF 1.1 Primer. Available at: https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/, accessed October 2021.

W3C (2013). SPARQL 1.1 Overview. Available at: https://www.w3.org/TR/sparql11-overview/, accessed October 2021.

W3C (2012). OWL 2 Web Ontology Language Primer (Second Edition). Available at: https://www.w3.org/TR/2012/REC-owl2-primer-20121211/, accessed October 2021.

Zhang, C., Beetz, J., de Vries, B. (2018). BimSPARQL: Domain-specific functional SPARQL extensions for querying RDF building data. Semantic Web 9, pp. 829–855. DOI: https://doi.org/10.3233/SW-180297