Validation of IfcOWL datasets using SHACL

Sander Stolk, MSc MA Head of Innovation Semmtech BV sanderstolk@semmtech.com Kris McGlinn, Ph.D. Research Fellow ADAPT Centre Trinity College Dublin kris.mcglinn@adaptentre.ie

Wouter Lubbers wouterlubbers@semmtech.nl

Outline



Introduction

Validation using OWL and SHACL

Obtaining SHACL shapes for IfcOWL Using two methods

Validating an IFC model

Introduction

- Standardisation is an important part of ensuring interoperability
- Industry Foundation Classes is the current leading standard within the AEC industry
 - Serializations as STEP, XML and RDF (ifcOWL)
- RDF and ifcOWL support interlinking between data sets
 - Major tool vendors (ArchiCAD, Revit) make use of XML and STEP, but do not support if cOWL
- A process of conversion is therefore needed from ifcOWL into STEP or XML formats

Introduction

- Approaches exist to convert if cOWL into STEP [1] but the original if cOWL must correctly support the schema, and this is not guaranteed.
- A method is needed to validate the ifcOWL against the IFC schema
- Here we present a method for validating ifcOWL using SHACL, a standard recommended by the W3C for validating asserted RDF.



[1] <u>https://github.com/BenzclyZhang/IfcSTEP-to-IfcOWL-converters</u>

Case study

- SHACL validation is applied to IfcOWL data, converted from a geospatial data set
- Validation prevents errors in transformation to STEP



Validation using OWL and SHACL

Validation using OWL

- OWL builds on top of RDF to express ontological facts
- Restrictions are used to provide detailed class definitions
- Reasoners exist to infer extra information based on provided information

Validation using OWL



Validation using SHACL

- For many applications, there was a need for validation rather than inference
- SHACL has been published to fill this gap
- While serving different purposes, SHACL and OWL have many similarities
 - A comparison between SHACL and OWL can be found online [2]
- Validation tools exist to check conformance to SHACL constraints

Validation using SHACL



Obtaining SHACL shapes for IfcOWL Method 1: SHACL Rules

11/

Applying rules to IfcOWL

- This method consists of two steps:
 - Extraction of data validation information from IfcOWL
 - Transforming this information to SHACL constraints
- The method applies the querying language SPARQL [3] and SHACL Rules [4]

Applying rules to IfcOWL

```
ifc:IfcDoorPanelProperties
                                                                     rdf:type sh:NodeShape ;
ifc:IfcDoorPanelProperties
                                                                     sh:property
  rdf:type owl:Class ;
  rdfs:subClassOf
                                                                         sh:class ifc:IfcPositiveLengthMeasure ;
                                                                         sh:path ifc:panelDepth_IfcDoorPanelProperties
      rdf:type owl:Restriction ;
      owl:allValuesFrom ifc:IfcPositiveLengthMeasure ;
                                                                     sh:property
      owl:onProperty ifc:panelDepth_IfcDoorPanelProperties
                                                                         sh:qualifiedMaxCount 1 ;
  rdfs:subClassOf
                                                                         sh:path ifc:panelDepth_IfcDoorPanelProperties ;
                                                                         sh:qualifiedValueShape
      rdf:type owl:Restriction ;
      owl:maxQualifiedCardinality "1"^^xsd:nonNegativeInteger ;
                                                                             sh:class ifc:IfcPositiveLengthMeasure
      owl:onProperty ifc:panelDepth_IfcDoorPanelProperties ;
      owl:onClass ifc:IfcPositiveLengthMeasure
   1:
                                                                     sh:property
  rdfs:subClassOf
                                                                         sh:class ifc:IfcDoorPanelOperationEnum ;
      rdf:type owl:Restriction ;
                                                                         sh:path ifc:panelOperation_IfcDoorPanelProperties
      owl:allValuesFrom ifc:IfcDoorPanelOperationEnum ;
                                                                       ];
      owl:onProperty ifc:panelOperation_IfcDoorPanelProperties
                                                                     sh:property
   1:
  rdfs:subClassOf
                                                                         sh:qualifiedMinCount 1 ;
                                                                         sh:qualifiedMaxCount 1 ;
      rdf:type owl:Restriction ;
                                                                         sh:path ifc:panelOperation_IfcDoorPanelProperties ;
      owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger ;
                                                                         sh:qualifiedValueShape
      owl:onProperty ifc:panelOperation_IfcDoorPanelProperties ;
      owl:onClass ifc:IfcDoorPanelOperationEnum
                                                                             sh:class ifc:IfcDoorPanelOperationEnum
```

Applying rules to IfcOWL

Benefits of this method

- Reuses readily available SHACL mechanisms and tooling
- Because the input is Linked Data, all relations and restrictions from ifcOWL are easy to query

• However, this method introduces an extra step on top of the existing EXPRESS to IfcOWL conversion

Obtaining SHACL shapes for IfcOWL Method 2: Generating IfcOWL with SHACL

Change to IfcOWL generation

- This method adds SHACL constraints during the generation of IfcOWL
- The EXPRESStoOWL tool [5] functionality has been extended to also output SHACL constraints

Change to IfcOWL generation

Expanding EXPRESStoOWL with SHACL capabilities (OWL Restrictions as SHACL shapes)

← → C (github.com/ssstolk/EXPRES	StoOWL				☆ ○ 🌢		🎗 :
	Ssstolk / EXPRESStoC	DWL			O Watch → 0	tar 0 Y Fork 2		*
	<>Code îî Pull requests 0 ○ Actions III Projects 0 III Wiki ① Security 1			Security III Insights	🔅 Settings			
	EXPRESStoOWL is a set of reusable Java components that allows to parse EXPRESS files and convert them into OWL ontologies in the Ed context of the Industry Foundation Classes (IFC).				ntologies in the Edit]	ł	
	56 commits	រ្រ 2 branches	🗊 0 packages	♥ 4 releases	2 contributors	Ճ∱ য View license		
	Branch: master New pull	request		Create new fi	le Upload files Find file	Clone or download 🗸		
	This branch is even with pipauwel:master.			្រ៉ា Pul	l request 🖹 Compare			
	pipauwel clean changes.md	l record			Latest commit 7a816a7 on Jul 31			
	🖿 doc	Mavenize codebase				3 years ago		
	src/main	added support for IFC4x1				8 months ago		
	.gitignore	Mavenize codebase				3 years ago		
	CHANGES.md	clean changes.md record				4 months ago		
		license change				3 years ago		
	README.md Project description 2			2 years ago				
	pom.xml	[maven-release-plugin] p	epare for next developme	ent iteration		4 months ago		

Change to IfcOWL generation

Expanding EXPRESStoOWL with SHACL capabilities (OWL Restrictions as SHACL shapes)

OWLWriter.java

10 out.write(getExpressOwlHeader()); 11 11 12 writePrimaryTypes(out); 13 writeHelperClasses(out); 14	
11	
12 writePrimaryTypes(out); 13 writeHelperClasses(out); 14	
13 writeHelperClasses(out); 14	
14 15 out.close(); 16 17) catch (IOException e) {	
<pre>15 out.close(); 16 17) catch (IOException e) {</pre>	
16 17 } catch (IOException e) {	
<pre>.17 } catch (IOException e) {</pre>	
18 e.printStackTrace();	
19 - }	
20 -	
21	
22 private void outputOWLproperty (BufferedWriter out, PropertyVO property) {	
23 try (
if (property.isList() property.isArray()) {	
<pre>25 out.write("ifc:" + property.getLowerCaseName() + "\r\n");</pre>	
<pre>26 out.write("\trdfs:label \"" + property.getOriginalName()// getOriginalNameLowerCase()</pre>	
27 + "\" ;\z\n");	
<pre>28 out.write("\trdfs:domain ifc:" + property.getDomain().getName() + " ;\r\n");</pre>	
29	
30 // range	
<pre>31 if (property.isListOfList())</pre>	
32 out.write("\trdfs:range " + property.getRangeNS() + ":" + property.getRange() + " List List ;\r\n"));
<pre>33 else if (property.isSet())</pre>	
<pre>34 out.write("\trdfs:range " + property.getRangeNS() + ":" + property.getRange() + " ;\r\n");</pre>	
35 else	
<pre>.36 out.write("\trdfs:range " + property.getRangeNS() + ":" + property.getRange() + "_List ;\r\n");</pre>	
37	
38 // inverse	
<pre>.39 if (property.getInverseProperty() != null)</pre>	
.40 out.write("\towl:inverseOf ifc:" + property.getInverseProperty().getLowerCaseName() + " ;\r\n");	
.41	
42 // typesetting	
<pre>.43 if (!property.isSet())</pre>	
44 out.write("\trdf:type owl:FunctionalProperty, owl:ObjectProperty .\r\n\r\n");	
45 else	
<pre>46 out.write("\trdf:type owl:ObjectProperty .\r\n\r\n"); 47</pre>	
48 dif (!property.getRangeNS().equalsIgnoreCase("expr")) {	
49 // write List range if necessary	
50 c (!property.isSet()) {	
<pre>51 = if (property.isListOfList()) {</pre>	
52 if (!listPropertiesOutput.contains(property.getRange() + "_List")) {	
53 // property not already contained in resulting	
54 // OWL file	
55 // (.TTL) -> no need to write additional	
56 // property	
57	
<pre>58 listPropertiesOutput.add(property.getRange() + "_List");</pre>	
59	
<pre>60</pre>	n");
<pre>61 out.write("\trdf:type owl:Class ;" + "\r\n");</pre>	
62 out.write("\trdfs:subClassOf list:EmptyList, " + property.getRangeNS() + ":" + property	y.getRange() +
63	
<pre>64 out.write(property.getRangeNS() + ":" + property.getRange() + "_List_List" + "\r\n");</pre>	

SHACLWriter.java

	170					
1						
2			<pre>writePrimaryTypes(out);</pre>			
3			writeHelperClasses(out):			
			wittenerpetciasses(out),			
.5			01	it.close();		
.6						
.7			} cate	ch (IOException e) (
8			е	printStackTrace():		
G			1			
~			,			
0	-	1				
1						
2	Ę	priv	ate v	oid outputSHACLproperty (BufferedWriter out, PropertyVO property) {		
3	-1		try {			
4	_		1	f (property.isList() property.isArray()) {		
5	5/*			out write ("if r : " + property getLower(aseName() + "\r\n").		
c	T			out units (1/1 vdfsidomin ifsill + nyopatu gatDomin () gatName () + ll (vhr))		
0				out.write((truis.domain fic. + property.getbomain().getwame() + ,(in),		
8				// range		
9				if (property.isListOfList())		
0				<pre>out.write("\trdfs:range " + property.getRangeNS() + ":" + property.getRange() + " List List ;\r\n");</pre>		
1				else if (property.isSet())		
2				ut write("trdferrance " + property getDangeNS() + "." + property getDange() + " .\r\n").		
2						
3						
4				<pre>out.write("\trdis:range " + property.getkangens() + ":" + property.getkange() + "_List ;\r\n");</pre>		
5						
6				// typesetting		
7				if (!property.isSet())		
8				out.write("\trdf:type_owl:FunctionalProperty, owl:ObjectProperty ,\r\n\r\n");		
G						
0				aut units (U)traffitume auti-ObjectDreporty ()r/n/r/nU); #/		
0				out.write('truit.type owr.objectroperty .(Inn'i'), "/		
1						
2				if (!property.getRangeNS().equalsIgnoreCase("expr")) {		
3				// write List range if necessary		
4	-			if (!property.isSet()) {		
5	-			if (property.isListOfList()) {		
6	_			if (!listPropertiesOutput.contains(property.getBange() + " List")) {		
7	T			// property not already contained in regulting		
0				// property not different in resulting		
0						
9				// (.TL) -> no need to write additional		
0				// property		
1						
2				<pre>listPropertiesOutput.add(property.getRange() + " List");</pre>		
3						
4				out.write(property.getRangeNS() + ":" + property.getRange() + " List List" + " $(r)n$ "):		
5				out write ("\trdf:type sh:NodeShape :" + "\r\p") .		
6						
-						
				out.write("\tsn:property" + "\r\n");		
8				<pre>out.write("\t\t[" + "\r\n");</pre>		
9				<pre>out.write("\t\t\tsh:path list:hasContents ;" + "\r\n");</pre>		
0				<pre>out.write("\t\tsh:class " + property.getRangeNS() + ":" + property.getRange() + " List" + "\r\n");</pre>		
1				out.write("/t/t];" + "/r/n");		
2						
3				out write ("/teb.property" + "/y/p").		
3				Out.wirde((com.property + (r(m),		

Change to IfcOWL generation

- This implementation proved to be more challenging
- Considering existing tooling to transform IFC data from STEP to IfcOWL, there is still a flaw

IfcOWL validation

Currently in lfcOWL2x3TC1

a owl:Restriction owl:onProperty ifc:lengthValue_IfcQuantityLength ; owl:allValuesFrom ifc:IfcLengthMeasure]



IfcOWL validation (SHACL)

Proposed SHACL shape

[sh:path ifc:lengthValue_IfcQuantityLength ; sh:class <u>express:REAL</u>]



Validating an IFC model using SHACL

Validating an IFC model using SHACL

- Recent work has seen the publication of Ordnance Survey Ireland (OSi, Ireland's national mapping agency) Prime2 data set
 - Contains information of over 50 million spatial objects including road segments, buildings, fences, etc.
- Using R2RML over 200 thousand buildings are now available as RDF using the GeoSPARQL vocabulary.
 - Geolocation, form, function, etc.
- R2RML has been demonstrated to enable the generation of 3D "walls" represented as ifcOWL from the simple 2D geospatial footprint.
- These mappings and resulting ifcOWL model are complex, and the SHACL approach provides a method for validation, the results are available here:
 - https://www.scss.tcd.ie/~mcglink/r2rml/



Validating an IFC model using SHACL

14	eprerix rur.	NILCO.//WWW.WJ.OIU/1999/02/22-1	ur-syntax-ns#/ .
13	<pre>@prefix inst:</pre>	< <u>http://linkedbuildingdata.net/</u>	ifc/resources20191018_103307/> .
14	<pre>@prefix expr:</pre>	< <u>https://w3id.org/express#</u> > .	_
15	<pre>@prefix vann:</pre>	< <u>http://purl.org/vocab/vann/</u> > .	
16			
17	[a	<pre>sh:ValidationReport ;</pre>	
18	sh:conforms	false ;	
19	sh:result	[a	sh:ValidationResult ;
20		sh:focusNode	inst:IfcCartesianPoint 62 ;
21		sh:resultMessage	"Value does not have shape :eca3dadf9cad748c7855f4e703c8eeee" ;
22		sh:resultPath	ifc:coordinates IfcCartesianPoint ;
23		sh:resultSeverity	sh:Violation ;
24		sh:sourceConstraintComponent	<pre>sh:NodeConstraintComponent ;</pre>
25		sh:sourceShape	:b0 ;
26		sh:value	
27		1 ;	
28	sh:result	[a	<pre>sh:ValidationResult ;</pre>
29		sh:focusNode	inst:IfcSIUnit 36 ;
30		sh:resultMessage	"Less than 1 values have shape :70f8f1f92af443851f6aeaabec3fb655" ;
31		sh:resultPath	ifc:dimensions IfcNamedUnit ;
32		sh:resultSeverity	sh:Violation ;
33		sh:sourceConstraintComponent	<pre>sh:QualifiedMinCountConstraintComponent ;</pre>
34		sh:sourceShape	_:b1
35		1 ;	
36	sh:result	[a	sh:ValidationResult ;
37		sh:focusNode	<pre>inst:IfcLengthMeasure_List_296 ;</pre>
38		sh:resultMessage	"Value must be an instance of ifc:IfcLengthMeasure" ;
39		sh:resultPath	list:hasContents ;
40		sh:resultSeverity	sh:Violation ;
41		sh:sourceConstraintComponent	<pre>sh:ClassConstraintComponent ;</pre>
42		sh:sourceShape	_:b2 ;
43		sh:value	inst:REAL_247
44];	
45	sh:result	í a	sh:ValidationResult :



Conclusion

- SHACL provides benefits for data validation over OWL
 - Designed specifically for data validation
 - Supported with readily available tooling
- Therefore, it is recommended to gradually adopt SHACL in IfcOWL for data validation