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

SEMMTECH

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 ifcOWL

- A process of conversion is therefore needed from ifcOWL into STEP or XML formats

SEMMTECH

Introduction

- Approaches exist to convert ifcOWL into STEP [1] but the original ifcOWL must correctly support the schema, and this is not guaranteed.

- A method is needed to validate the ifcOWL against the IFC schema

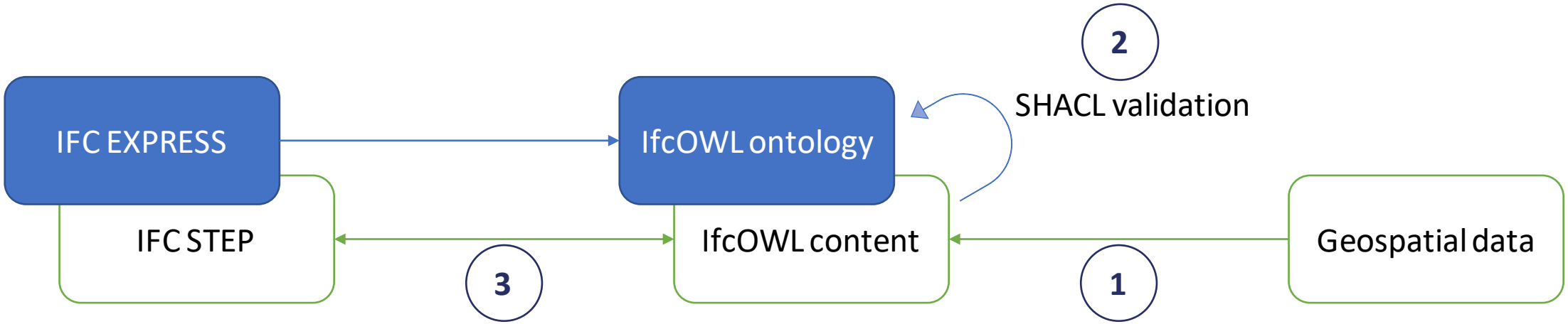
- Here we present a method for validating ifcOWL using SHACL, a standard recommended by the W3C for validating asserted RDF.

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

SEMMTECH

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

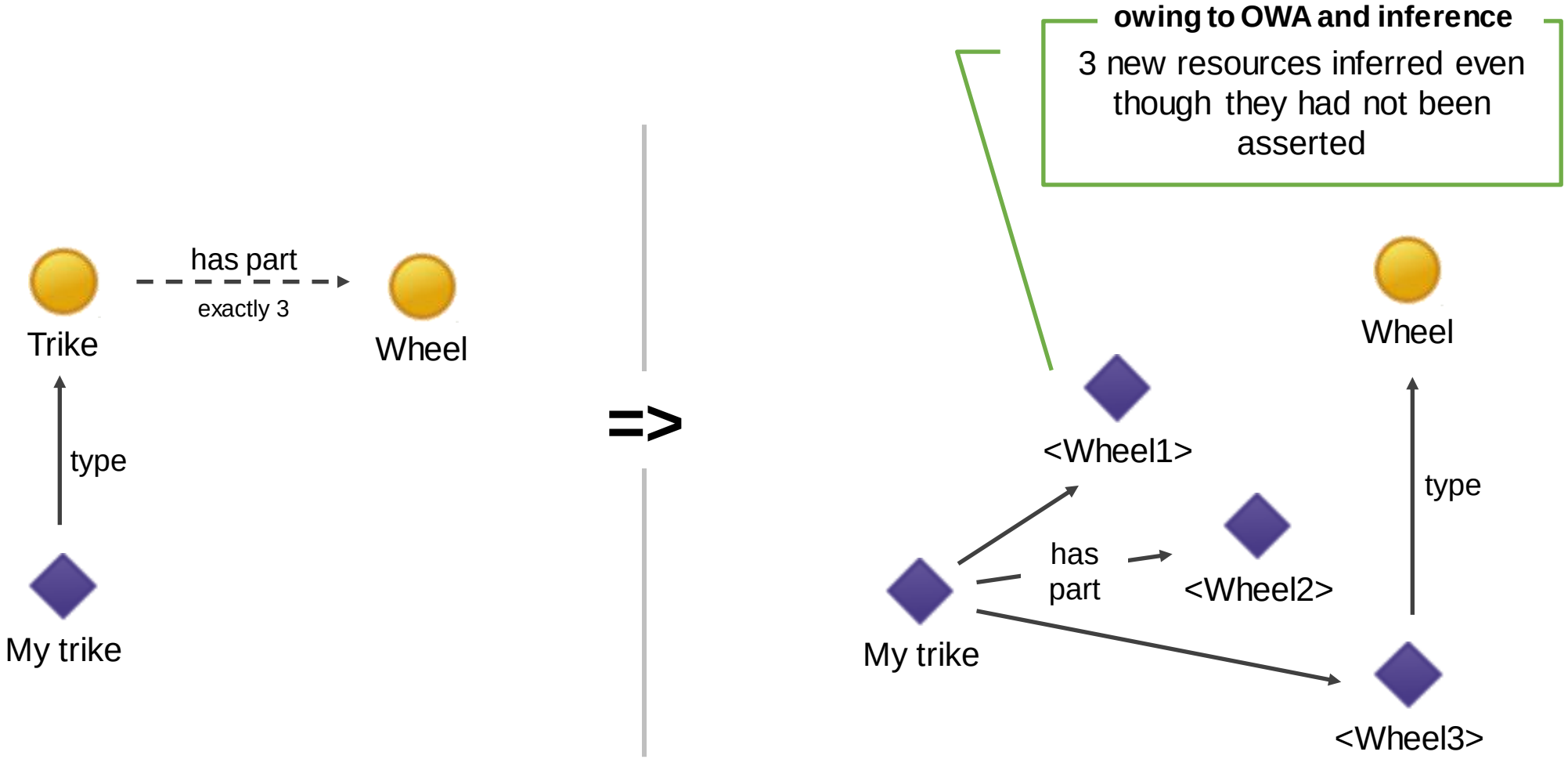
SEMMTECH

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

SEMMTECH

Validation using OWL



SEMMTECH

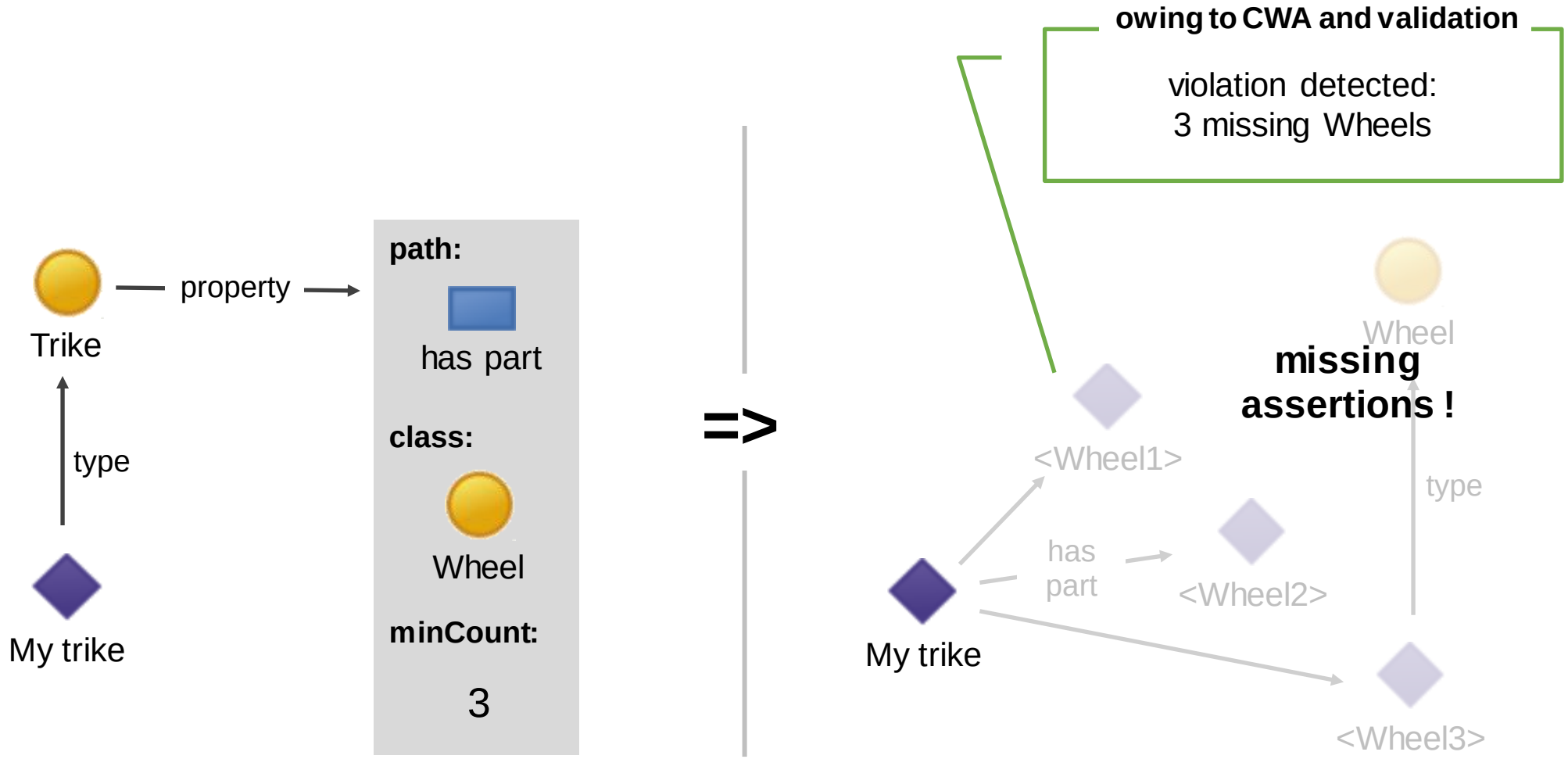
Validation using SHACL

- 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

[2] https://spinrdf.org/shacland-owl.html

SEMMTECH

Validation using SHACL



Obtaining SHACL shapes for IfcOWL

Method 1: SHACL Rules

SEMMTECH

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]

[3] https://www.w3.org/TR/sparql11-query/
[4] https://www.w3.org/TR/shacl-af/

SEMMTECH

Applying rules to IfcOWL

```
ifc:IfcDoorPanelProperties
  rdf:type owl:Class ;
  rdfs:subClassOf
    [
      rdf:type owl:Restriction ;
      owl:allValuesFrom ifc:IfcPositiveLengthMeasure ;
      owl:onProperty ifc:panelDepth_IfcDoorPanelProperties
    ] ;
  rdfs:subClassOf
    [
      rdf:type owl:Restriction ;
      owl:maxQualifiedCardinality "1"^^xsd:nonNegativeInteger ;
      owl:onProperty ifc:panelDepth_IfcDoorPanelProperties ;
      owl:onClass ifc:IfcPositiveLengthMeasure
    ] ;
  rdfs:subClassOf
    [
      rdf:type owl:Restriction ;
      owl:allValuesFrom ifc:IfcDoorPanelOperationEnum ;
      owl:onProperty ifc:panelOperation_IfcDoorPanelProperties
    ] ;
  rdfs:subClassOf
    [
      rdf:type owl:Restriction ;
      owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger ;
      owl:onProperty ifc:panelOperation_IfcDoorPanelProperties ;
      owl:onClass ifc:IfcDoorPanelOperationEnum
    ]
  .
```

```
ifc:IfcDoorPanelProperties
  rdf:type sh:NodeShape ;
  sh:property
    [
      sh:class ifc:IfcPositiveLengthMeasure ;
      sh:path ifc:panelDepth_IfcDoorPanelProperties
    ] ;
  sh:property
    [
      sh:qualifiedMaxCount 1 ;
      sh:path ifc:panelDepth_IfcDoorPanelProperties ;
      sh:qualifiedValueShape
        [
          sh:class ifc:IfcPositiveLengthMeasure
        ]
    ] ;
  sh:property
    [
      sh:class ifc:IfcDoorPanelOperationEnum ;
      sh:path ifc:panelOperation_IfcDoorPanelProperties
    ] ;
  sh:property
    [
      sh:qualifiedMinCount 1 ;
      sh:qualifiedMaxCount 1 ;
      sh:path ifc:panelOperation_IfcDoorPanelProperties ;
      sh:qualifiedValueShape
        [
          sh:class ifc:IfcDoorPanelOperationEnum
        ]
    ]
  .
```

SEMMTECH

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

SEMMTECH

Change to IfcOWL generation

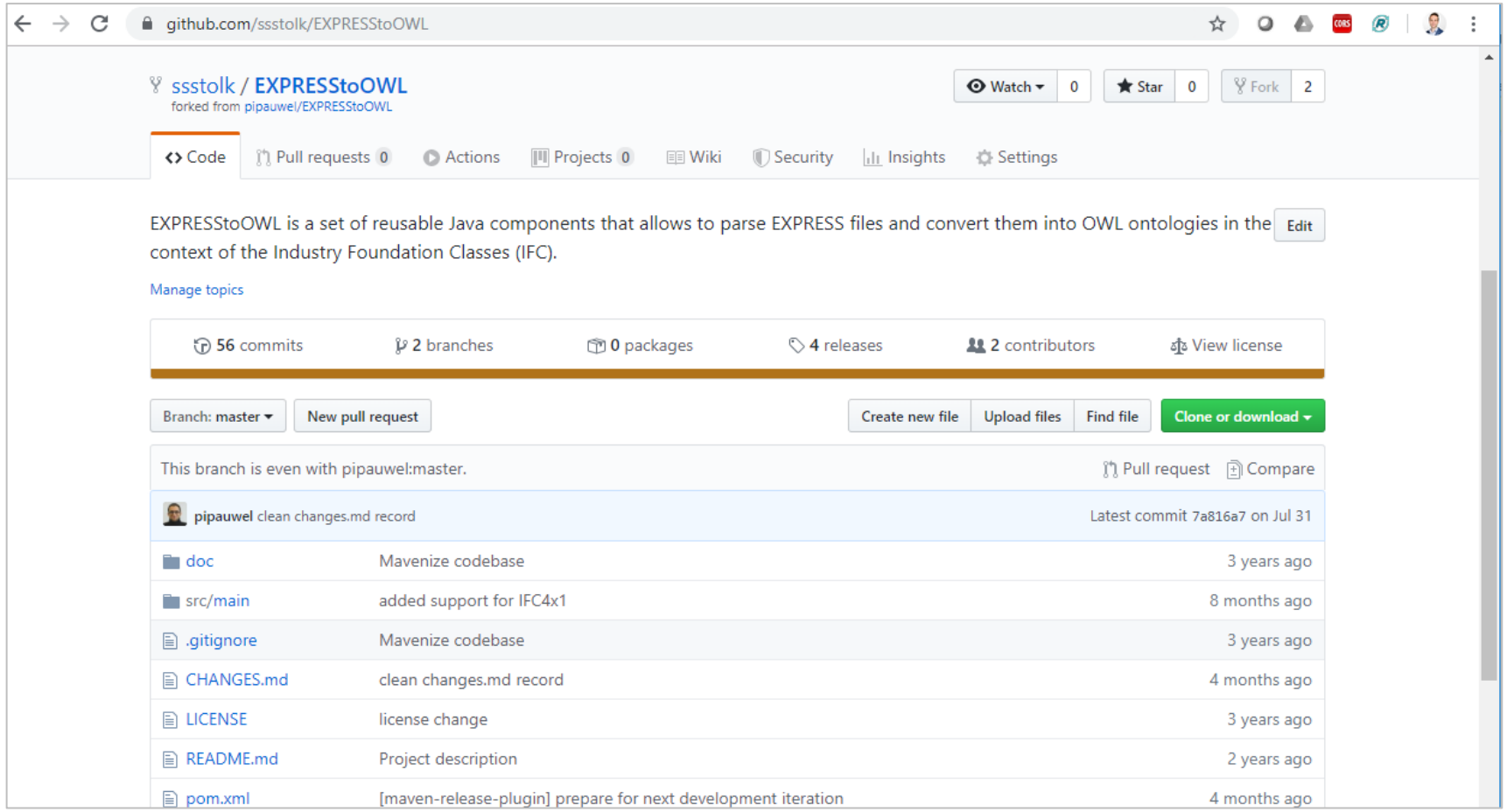
- This method adds SHACL constraints during the generation of IfcOWL
- The EXPRESStoOWL tool [5] functionality has been extended to also output SHACL constraints

[5] https://github.com/pipauwel/EXPRESStoOWL/

SEMMTECH

Change to IfcOWL generation

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



SEMMTECH

Change to IfcOWL generation

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

OWLWriter.java

```
            out.write("@prefix expr: <" + Namespace.EXPRESS + "#> .\r\n");
            out.write(getExpressOwlHeader());

            writePrimaryTypes(out);
            writeHelperClasses(out);

            out.close();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void outputOWLproperty(BufferedWriter out, PropertyVO property) {
        try {
            if (property.isList() || property.isArray()) {
                out.write("ifc:" + property.getLowerCaseName() + "\r\n");
                out.write("\trdfs:label \"" + property.getOriginalName()// getOriginalNameLowerCase()
                        + "\" ;\r\n");
                out.write("\trdfs:domain ifc:" + property.getDomain().getName() + " ;\r\n");

                // range
                if (property.isListOfList())
                    out.write("\trdfs:range " + property.getRangeNS() + ":" + property.getRange() + "_List_List ;\r\n");
                else if (property.isSet())
                    out.write("\trdfs:range " + property.getRangeNS() + ":" + property.getRange() + " ;\r\n");
                else
                    out.write("\trdfs:range " + property.getRangeNS() + ":" + property.getRange() + "_List ;\r\n");

                // inverse
                if (property.getInverseProperty() != null)
                    out.write("\towl:inverseOf ifc:" + property.getInverseProperty().getLowerCaseName() + " ;\r\n");

                // typesetting
                if (!property.isSet())
                    out.write("\trdf:type owl:FunctionalProperty, owl:ObjectProperty .\r\n\r\n");
                else
                    out.write("\trdf:type owl:ObjectProperty .\r\n\r\n");

                if (!property.getRangeNS().equalsIgnoreCase("expr")) {
                    // write List range if necessary
                    if (!property.isSet()) {
                        if (property.isListOfList()) {
                            if (!listPropertiesOutput.contains(property.getRange() + "_List")) {
                                // property not already contained in resulting
                                // OWL file
                                // (.TTL) -> no need to write additional
                                // property

                                listPropertiesOutput.add(property.getRange() + "_List");

                                out.write(property.getRangeNS() + ":" + property.getRange() + "_List_EmptyList" + "\r\n");
                                out.write("\trdf:type owl:Class ;" + "\r\n");
                                out.write("\trdfs:subClassOf list:EmptyList, " + property.getRangeNS() + ":" + property.getRange() +

                                out.write(property.getRangeNS() + ":" + property.getRange() + "_List_List" + "\r\n");
```

SHACLWriter.java

```
                writePrimaryTypes(out);
                writeHelperClasses(out);

                out.close();

            } catch (IOException e) {
                e.printStackTrace();
            }
        }

    private void outputSHACLproperty(BufferedWriter out, PropertyVO property) {
        try {
            if (property.isList() || property.isArray()) {
/*              out.write("ifc:" + property.getLowerCaseName() + "\r\n");
                out.write("\trdfs:domain ifc:" + property.getDomain().getName() + " ;\r\n");

                // range
                if (property.isListOfList())
                    out.write("\trdfs:range " + property.getRangeNS() + ":" + property.getRange() + "_List_List ;\r\n");
                else if (property.isSet())
                    out.write("\trdfs:range " + property.getRangeNS() + ":" + property.getRange() + " ;\r\n");
                else
                    out.write("\trdfs:range " + property.getRangeNS() + ":" + property.getRange() + "_List ;\r\n");

                // typesetting
                if (!property.isSet())
                    out.write("\trdf:type owl:FunctionalProperty, owl:ObjectProperty .\r\n\r\n");
                else
                    out.write("\trdf:type owl:ObjectProperty .\r\n\r\n"); */

                if (!property.getRangeNS().equalsIgnoreCase("expr")) {
                    // write List range if necessary
                    if (!property.isSet()) {
                        if (property.isListOfList()) {
                            if (!listPropertiesOutput.contains(property.getRange() + "_List")) {
                                // property not already contained in resulting
                                // OWL file
                                // (.TTL) -> no need to write additional
                                // property

                                listPropertiesOutput.add(property.getRange() + "_List");

                                out.write(property.getRangeNS() + ":" + property.getRange() + "_List_List" + "\r\n");
                                out.write("\trdf:type sh:NodeShape ;" + "\r\n");

                                out.write("\tsh:property" + "\r\n");
                                out.write("\t\t[" + "\r\n");
                                out.write("\t\t\tsh:path list:hasContents ;" + "\r\n");
                                out.write("\t\t\tsh:class " + property.getRangeNS() + ":" + property.getRange() + "_List" + "\r\n");
                                out.write("\t\t] ;" + "\r\n");

                                out.write("\tsh:property" + "\r\n");
```

SEMMTECH

Change to IfcOWL generation

- This implementation proved to be more challenging

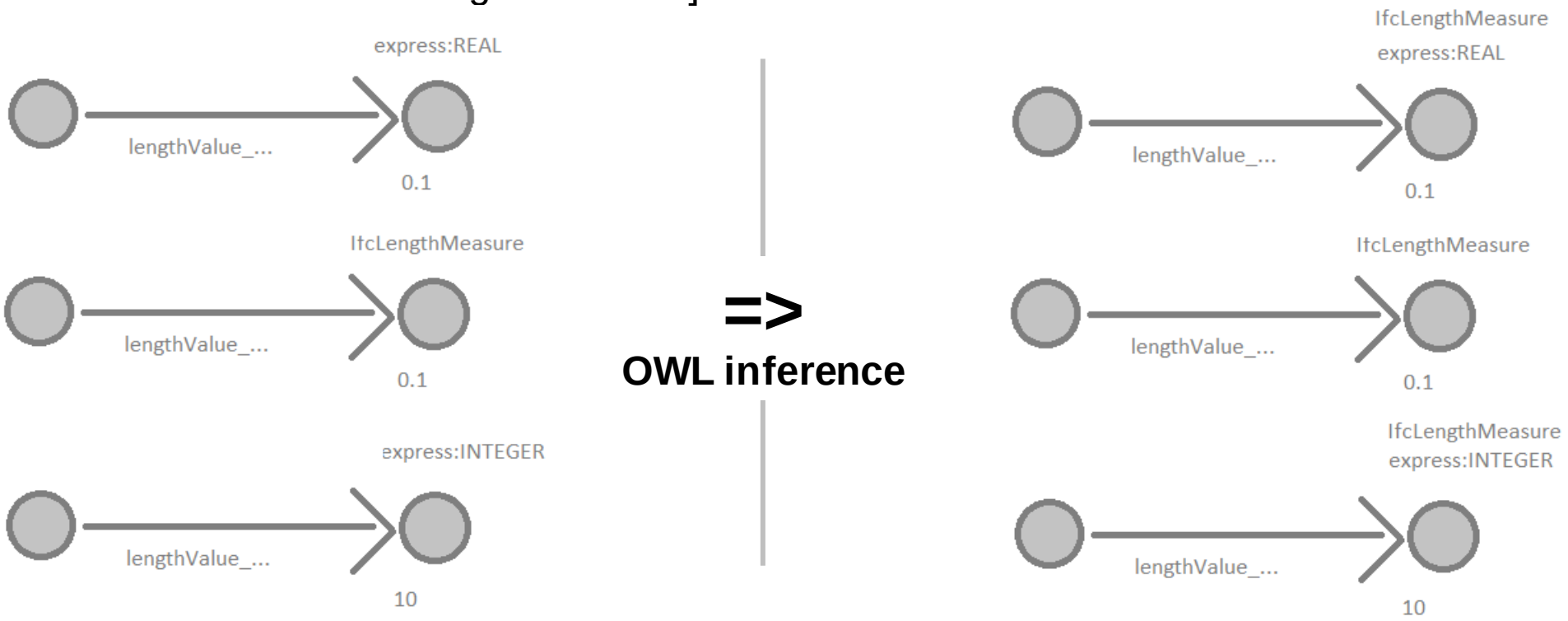
- Considering existing tooling to transform IFC data from STEP to IfcOWL, there is still a flaw

SEMMTECH

IfcOWL validation

Currently in IfcOWL2x3TC1

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

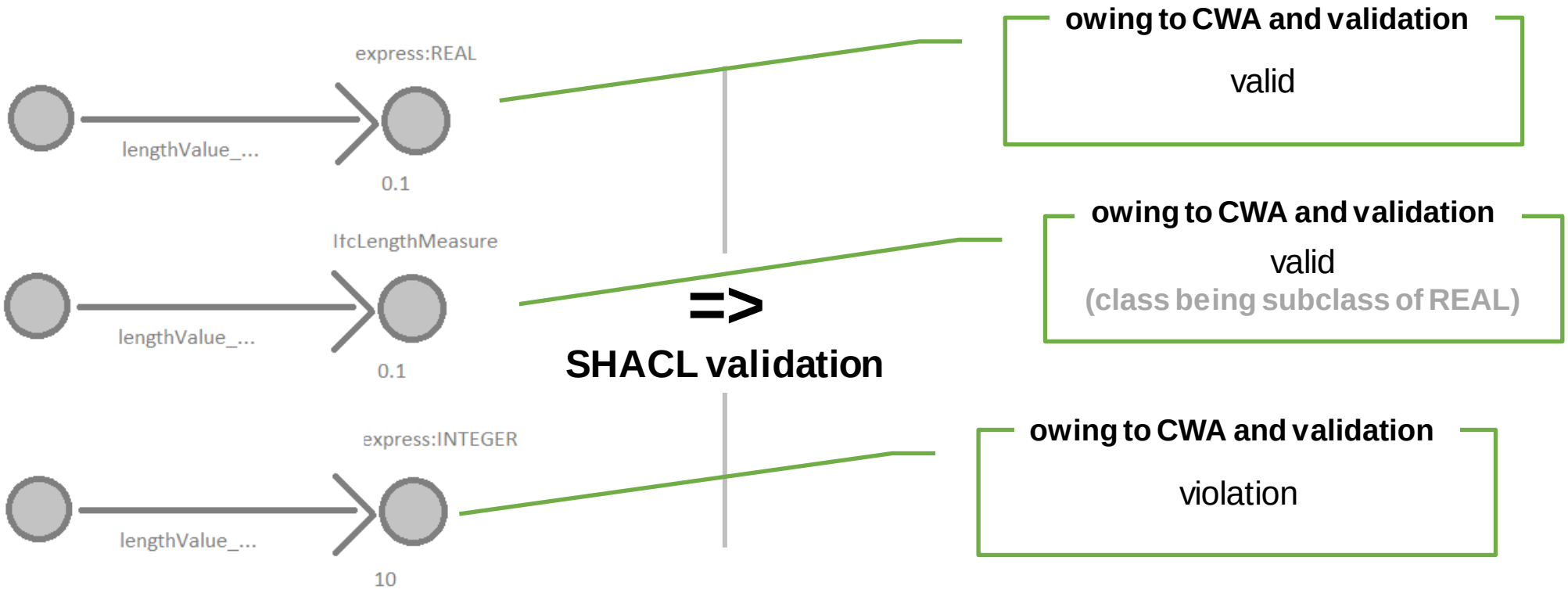


SEMMTECH

IfcOWL validation (SHACL)

Proposed SHACL shape

[sh:path ifc:lengthValue_IfcQuantityLength ;
sh:class express:REAL]

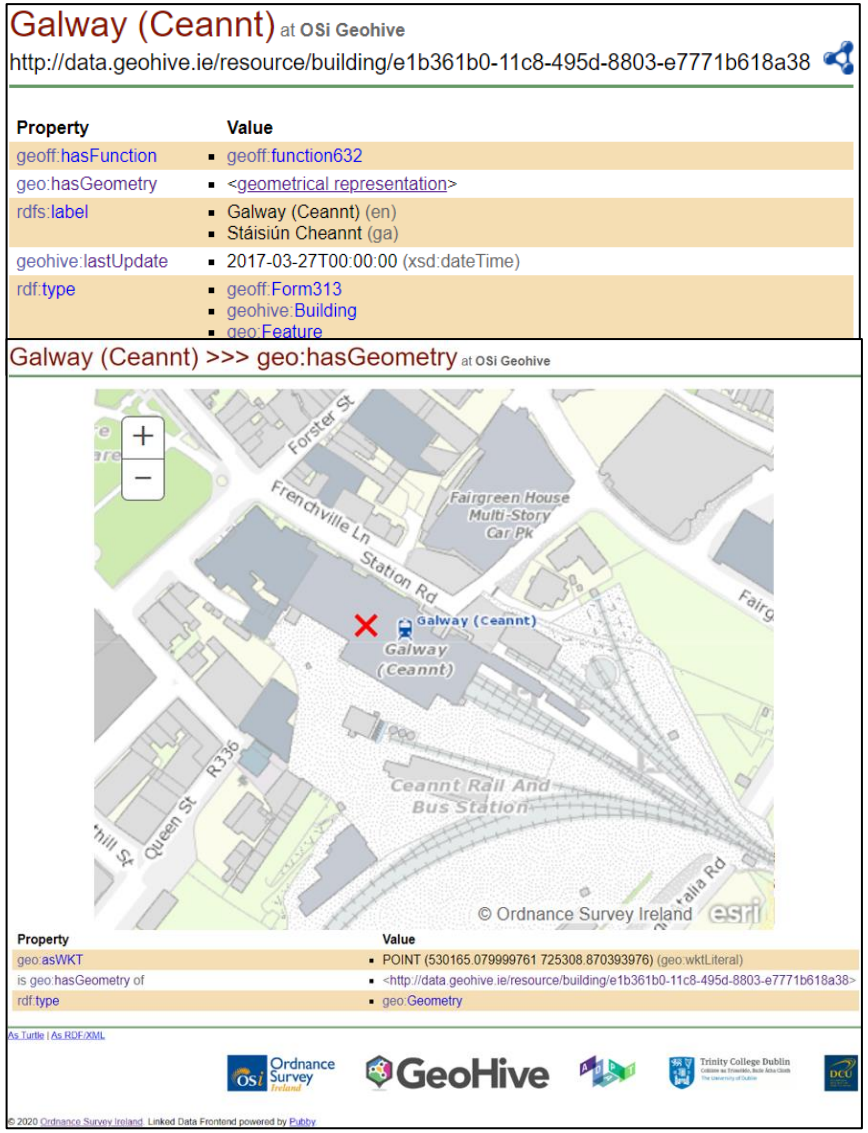


Validating an IFC model using SHACL

SEMMTECH

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/



SEMMTECH

Validating an IFC model using SHACL

```
@prefix idr:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix inst:  <http://linkedbuildingdata.net/ifc/resources20191018_103307/> .
@prefix expr:  <https://w3id.org/express#> .
@prefix vann:  <http://purl.org/vocab/vann/> .

[ a                 sh:ValidationReport ;
  sh:conforms       false ;
  sh:result         [ a                             sh:ValidationResult ;
                      sh:focusNode                  inst:IfcCartesianPoint_62 ;
                      sh:resultMessage              "Value does not have shape _:eca3dadf9cad748c7855f4e703c8eeee" ;
                      sh:resultPath                 ifc:coordinates_IfcCartesianPoint ;
                      sh:resultSeverity             sh:Violation ;
                      sh:sourceConstraintComponent  sh:NodeConstraintComponent ;
                      sh:sourceShape                _:b0 ;
                      sh:value                      inst:IfcLengthMeasure_List_251
                    ] ;
  sh:result         [ a                             sh:ValidationResult ;
                      sh:focusNode                  inst:IfcSIUnit_36 ;
                      sh:resultMessage              "Less than 1 values have shape _:70f8f1f92af443851f6aeaabec3fb655" ;
                      sh:resultPath                 ifc:dimensions_IfcNamedUnit ;
                      sh:resultSeverity             sh:Violation ;
                      sh:sourceConstraintComponent  sh:QualifiedMinCountConstraintComponent ;
                      sh:sourceShape                _:b1
                    ] ;
  sh:result         [ a                             sh:ValidationResult ;
                      sh:focusNode                  inst:IfcLengthMeasure_List_296 ;
                      sh:resultMessage              "Value must be an instance of ifc:IfcLengthMeasure" ;
                      sh:resultPath                 list:hasContents ;
                      sh:resultSeverity             sh:Violation ;
                      sh:sourceConstraintComponent  sh:ClassConstraintComponent ;
                      sh:sourceShape                _:b2 ;
                      sh:value                      inst:REAL_247
                    ] ;
  sh:result         [ a                             sh:ValidationResult ;
```

Conclusion

SEMMTECH

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