Shapes Constraint Language Masterclass

Veronika Heimsbakk

Shapes Constraint Language Masterclass

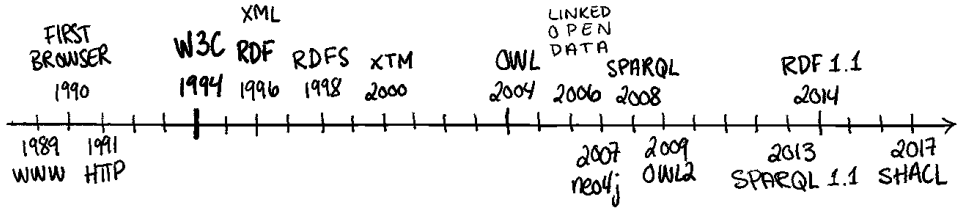
Veronika Heimsbakk

Knowledge Graph Lead

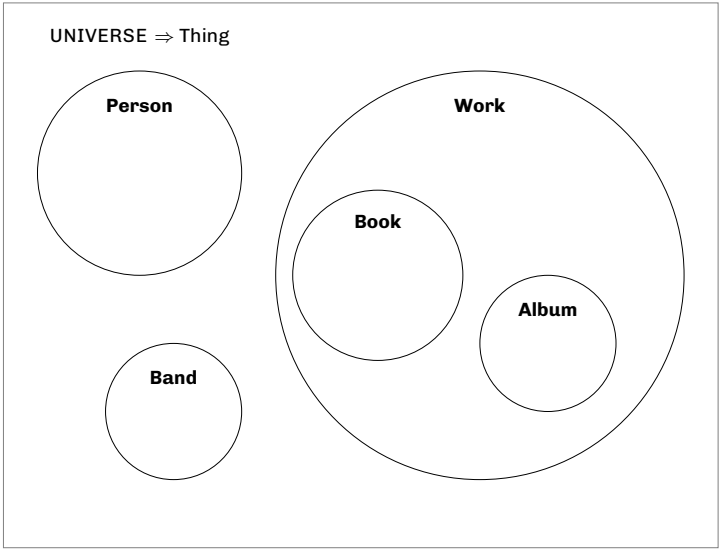
veronika.heimsbakk@capgemini.com

- vheimsbakk
- veleda
- veronikaheim
- veronahe.no

Once upon a time...



Things!



Classes
Work
Book
Album
Band
Person

Sub-classes
Book $\subseteq$ Work
Album $\subseteq$ Work

Axiom
Book $\cap$ Album $= \emptyset$

Classes

Work

```
ex:Work rdf:type owl:Class .
```

Book

```
ex:Book rdf:type owl:Class .
```

Album

```
ex:Album rdf:type owl:Class .
```

Band

```
ex:Band rdf:type owl:Class .
```

Person

```
ex:Person rdf:type owl:Class .
```

Sub-classes

Book $\subseteq$ Work

```
ex:Book rdfs:subClassOf ex:Work .
```

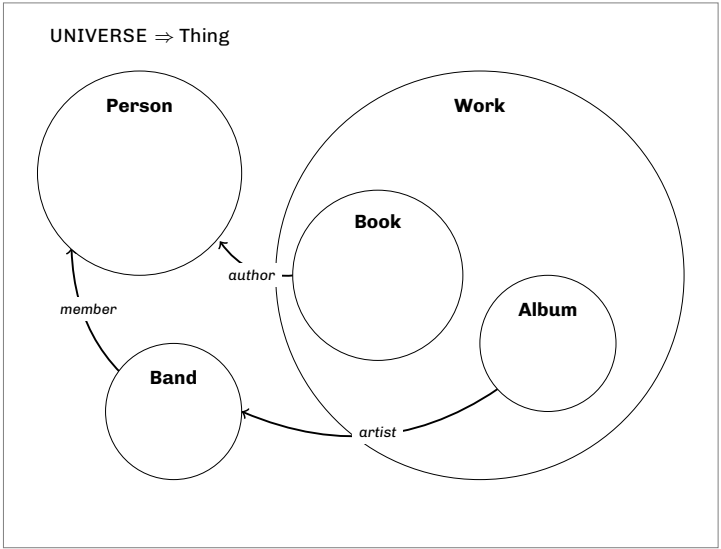
Album $\subseteq$ Work

```
ex:Album rdfs:subClassOf ex:Work .
```

Axiom

Book $\cap$ Album $= \emptyset$

```
ex:Book owl:disjointWith ex:Album .
```



Relationships
artist⟨Album, Band⟩
member⟨Band, Person⟩
author⟨Book, Person⟩

Relationships

artist⟨Album, Band⟩

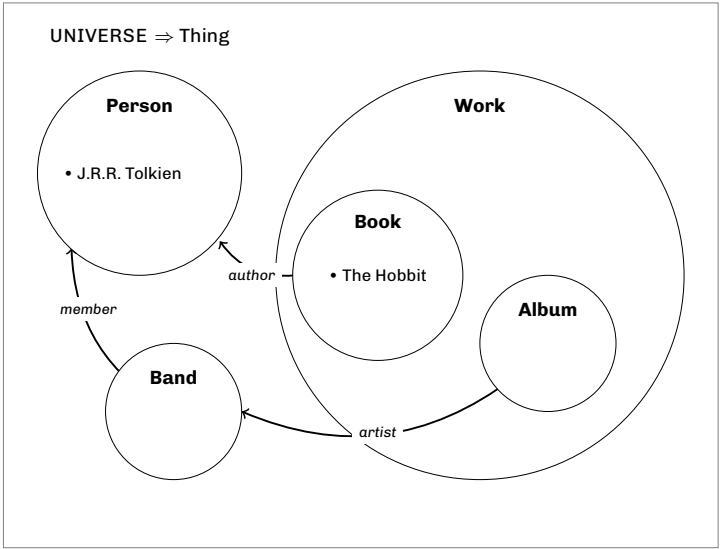
```
ex:artist rdf:type owl:ObjectProperty ;
          rdfs:domain ex:Album ;
          rdfs:range ex:Band .
```

member⟨Band, Person⟩

```
ex:member rdf:type owl:ObjectProperty ;
          rdfs:domain ex:Band ;
          rdfs:range ex:Person .
```

author⟨Book, Person⟩

```
ex:author rdf:type owl:ObjectProperty ;
          rdfs:domain ex:Book ;
          rdfs:range ex:Person .
```



Elements
The Hobbit ∈ Book
J.R.R. Tolkien ∈ Person

Elements

The Hobbit $\in$ Book

```
ex:TheHobbit rdf:type owl:NamedIndividual ,
                      ex:Book .
```

J.R.R. Tolkien $\in$ Person

```
ex:JRRTolkien rdf:type owl:NamedIndividual ,
                       ex:Person .
```

World assumptions

Open world assumption (OWA)

- Admits incomplete knowledge.
- Ontologies with Web Ontology Language (OWL).

The assumption that the truth value of a statement may be true irrespective of whether or not it is known to be true.

Example

Statement: In a hole in the ground there lived a hobbit.
Question: Do Gandalf live in a hole in the ground?
OWA: Unknown

Closed world assumption (CWA)

- Shape constraints with Shape Constraint Language (SHACL).

Any statement that is true is known to be true. What is not currently known to be true is false.

Example

Statement: In a hole in the ground there lived a hobbit.
Question: Do Gandalf live in a hole in the ground?
CWA: No

Shape Constraint Language

A language for describing and validating RDF graphs

SPIN & ShEx

SPIN

- Both are backed by SPARQL.
- SHACL Constraint Components are more flexible than SPIN Templates due to the possibility of combining multiple constraint types into same shape definition.
- SPIN Templates require new instances and multiple `spin:constraint` triples.

ShEx

- Both have shapes on nodes and properties.
- ShEx intends to be a grammar or schema for RDF.
- SHACL aims to provide a constraint language for RDF.
- ShEx returns an annotated data graph.
- SHACL return a validation report as RDF.

SHACL & OWL

Common

- RDF & URIs
- Rely on RDF Schema (RDFS)

Difference

Inference Validation

When to use SHACL?

It depends on the use case!

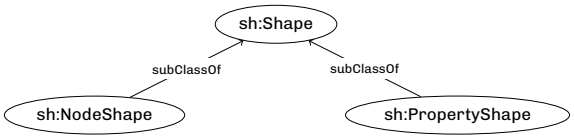
- Validation, before or after reasoning (or both)
- Automate certain parts of a data pipeline
- Acceptance testing ontologies and/or shapes
- Information modelling
- Generate user interface
- As a schema

We'll get back to this!

SHACL Shape

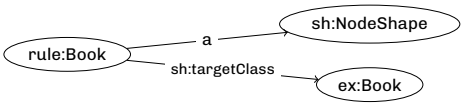
A collection of constraints for given RDF resource.

- Shapes about focus nodes (**sh:NodeShape**).
- Shapes about values of a property or path for the focus node (**sh:PropertyShape**).



sh:NodeShape

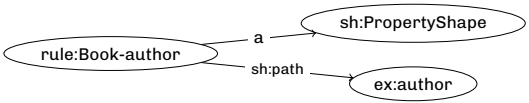
A *node shape* is a shape that describes focus nodes (subject)!




```
rule:Book
  a sh:NodeShape ;
  sh:targetClass ex:Book .
```

sh:PropertyShape

A *property shape* is a shape that describes properties and their values (and that is the subject of a triple that has *sh:path* as its predicate).




```
rule:Book-author
  a sh:PropertyShape ;
  sh:path ex:author .
```

Books with authors

```
rule:Book
  a sh:NodeShape ;
  sh:targetClass ex:Book ;
  sh:property rule:Book-author .

rule:Book-author
  a sh:PropertyShape ;
  sh:path ex:author .
```

SHACL Core Constraint Components

SHACL Core Constraint Components

Value type

sh:class	Each value node is an instance of a given type.
sh:datatype	Datatype of each value node.
sh:nodeKind	Node kind (IRI, blank node etc.) of each value node.

```
rule:Book
  a sh:NodeShape ;
  sh:targetClass ex:Book ;
  sh:property [
    sh:path ex:author ;
    sh:class foaf:Person ;
  ] ;
  sh:property [
    sh:path ex:published ;
    sh:datatype xsd:date ;
  ] .
```

SHACL Core Constraint Components — example

Example of conforming data

```
ex:TheHobbit a ex:Book ;
 ex:author ex:JRRTolkien ;
 ex:published "1937-09-21"^^xsd:date .

ex:JRRTolkien a foaf:Person .
```

Example of non-conforming data

```
ex:TheHobbit a ex:Book ;
 ex:author ex:JRRTolkien ;
 ex:published "1937"^^xsd:gYear .

ex:JRRTolkien a ex:Person .
```

SHACL Core Constraint Components

Cardinality

sh:minCount	Minimum cardinality as xsd:integer.
sh:maxCount	Maximum cardinality as xsd:integer.

Value range

sh:minExclusive	$x <$ $value
sh:minInclusive	$x <=$ $value
sh:maxExclusive	$x >$ $value
sh:maxInclusive	$x >=$ $value

```
rule:Book
  a sh:NodeShape ;
  sh:targetClass ex:Book ;
  sh:property [
    sh:path ex:mainTitle ;
    sh:datatype xsd:string ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
  ] ;
```

```
  sh:property [
    sh:path ex:pages ;
    sh:datatype xsd:integer ;
    sh:minInclusive 10 ;
    sh:maxExclusive 10000 ;
  ] .
```

SHACL Core Constraint Components — example

Example of conforming data

```
ex:TheHobbit a ex:Book ;
  ex:mainTitle "The Hobbit, or There and Back Again" ;
  ex:pages "310"^^xsd:integer .
```

Example of non-conforming data

```
ex:TheHobbit a ex:Book ;
  ex:mainTitle "The Hobbit, or There and Back Again" ;
  ex:mainTitle "Hobbiten, eller ditut og attende" ;
  ex:pages "5"^^xsd:integer .
```

SHACL Core Constraint Components

String-based

sh:minLength	Minimum length as xsd:integer.
sh:maxLength	Maximum length as xsd:integer.
sh:pattern	Regular expression.
sh:languageIn	A list of languages.
sh:uniqueLang	One unique tag per language.

```
rule:Book
 a sh:NodeShape ;
 sh:targetClass ex:Book ;
 sh:property [
  sh:path ex:mainTitle ;
  sh:datatype rdf:langString ;
  sh:minLength 2 ;
  sh:maxLength 100 ;
  sh:uniqueLang true ;
 ] ;
 sh:property [
   sh:path ex:isbn ;
   sh:pattern "^(?=(?:\D*\d){10}(?:(?:\D*\d){3})?$)[\d-]+$" ;
 ] .
```

SHACL Core Constraint Components — example

Example of conforming data

```
ex:TheHobbit a ex:Book ;
  ex:mainTitle "Hobbiten, eller ditut og attende"@nn ;
  ex:mainTitle "Hobbiten, eller Fram og tilbake igjen"@nb ;
  ex:mainTitle "The Hobbit, or There and Back Again"@en ;
  ex:isbn "978-0-261-10221-7" .
```

Example of non-conforming data

```
ex:TheHobbit a ex:Book ;
  ex:mainTitle "The Hobbit"@en ;
  ex:mainTitle "The Hobbit, or There and Back Again"@en ;
  ex:isbn "123" .
```

SHACL Core Constraint Components

Property pair	Compare two IRIs where,
sh:equals	$x \equiv y$
sh:disjoint	$x \cap y = \emptyset$
sh:lessThan	$x < y$
sh:lessThanOrEquals	$x <= y$

```
rule:Person
 a sh:NodeShape ;
 sh:targetClass ex:Person ;
 sh:property [
  sh:path ex:birthDate ;
  sh:lessThanOrEquals
    ex:deathDate ;
 ] .
```

SHACL Core Constraint Components — example

Example of conforming data

```
ex:JRRTolkien a ex:Person ;
 ex:birthDate "1892-01-03"^^xsd:date ;
 ex:deathDate "1973-09-02"^^xsd:date .
```

Example of non-conforming data

```
ex:JRRTolkien a ex:Person ;
 ex:birthDate "1992-01-03"^^xsd:date ;
 ex:deathDate "1973-09-02"^^xsd:date .
```

SHACL Core Constraint Components

Logical	List of value nodes that,
sh:not	Cannot conform to given shape.
sh:and	Conforms to all provided shapes.
sh:or	Conforms to at least one of the provided shapes.
sh:xone	Conforms to exactly one of the provided shapes.

```
rule:Person
  a sh:NodeShape ;
  sh:targetClass ex:Person ;
  sh:xone (
    [
      sh:path ex:birthDate ;
      sh:datatype xsd:date ;
      sh:minCount 1 ;
    ]
    [
      sh:path ex:birthYear ;
      sh:datatype xsd:gYear ;
      sh:minCount 1 ;
    ]
  ) .
```

SHACL Core Constraint Components — example

Example of conforming data

```
ex:JRRTolkien a ex:Person ;
 ex:birthDate "1982-01-03"^^xsd:date .
```

Example of non-conforming data

```
ex:JRRTolkien a ex:Person .
```

SHACL Core Constraint Components

Shape-based	Each value node,
sh:node	Conforms to the given node shape.
sh:property	Has a given property shape.
Other	
sh:closed	Boolean signalising a complete shape.
sh:ignoredProperties	List of properties to ignore.
sh:hasValue	At least one value node is equal to the given term.
sh:in	Value node is member of given list.

```
rule:TheHobbit
 a sh:NodeShape ;
 sh:targetNode ex:TheHobbit ;
 sh:closed true ;
 sh:ignoredProperties (rdf:type) ;
 sh:property [
  sh:path ex:author ;
  sh:hasValue ex:JRRTolkien ;
 ] .
```

SHACL Core Constraint Components — example

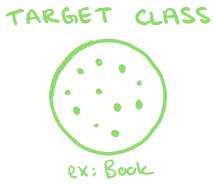
Example of conforming data

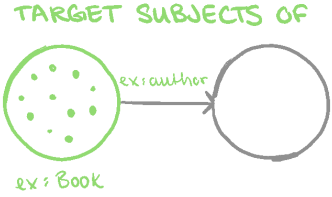
```
ex:TheHobbit a ex:Book ;
  ex:author ex:JRRTolkien .
```

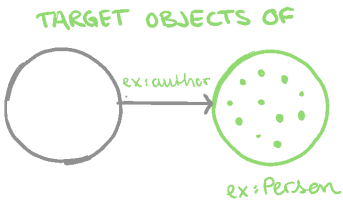
Example of non-conforming data

```
ex:TheHobbit a ex:Book ;
  ex:genre ex:Fantasy ;
  ex:author ex:JRRTolkien .
```

Oooops! Targets!







Quirks!

```
rule:Book
  a sh:NodeShape, owl:Class ;
  sh:property [
    sh:path ex:author ;
    sh:or (
      [ sh:class foaf:Person ; ]
      [ sh:datatype xsd:string ; ]
    )
  ] .
```

Other nice to knows about SHACL

For property shapes:

`sh:name` A label. Can have multiple values, one per language tag.

`sh:description` Description or comment. Can have multiple values, one per language tag.

`sh:order` Relative order as a decimal number.

`sh:group` Have range to an URI instance of `sh:PropertyGroup`.

`sh:defaultValue` Contains no fixed semantics. Should align with `sh:datatype` or `sh:class` for given shape.

SHACL-SPARQL

sh:message	Annotation used for validation result.
sh:prefixes	IRI to prefix mapping.
sh:select	String containing the SPARQL query.

```
ex:Book
  a sh:NodeShape ;
  sh:message "The mainTitle must be in Nynorsk."@en ;
  sh:sparql [
      a sh:SPARQLConstraint ;
      sh:prefixes ex: ;
      sh:select """
        SELECT $this ?mainTitle WHERE {
          $this ex:mainTitle ?mainTitle .
          FILTER(LANG(?mainTitle) = "nn")
        }
        """ ;
    ] ;
.
```

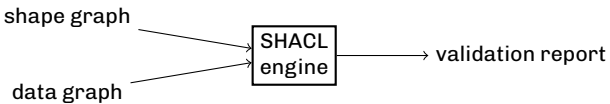
SHACL-SPARQL

```
ex:
 a owl:Ontology ;
 owl:imports sh: ;
 sh:declare [
  sh:prefix "ex" ;
  sh:namespace "http://data.veronahe.no/collection/"^^xsd:anyURI ;
 ] ;
 sh:declare [
  sh:prefix "rule" ;
  sh:namespace "http://rules.veronahe.no/collection/"^^xsd:anyURI ;
 ] .
```

SHACL-SPARQL

- Queries cannot contain a `MINUS` clause.
- Queries cannot be a federated query (`SERVICE`).
- Queries cannot contain a `VALUES` clause.
- Queries cannot use the syntax form `AS ?x` for any prebound variable.

SHACL engine



Validation report

Each instance of *sh:ValidationReport* has exactly one value of *sh:conforms*.

sh:conforms is true iff the validation did not produce any **validation results**, and false otherwise.

Iff validation conforms false, the report will contain an instance of **sh:ValidationResult**.

```
[
  a sh:ValidationReport ;
  sh:conforms true ;
] .
```

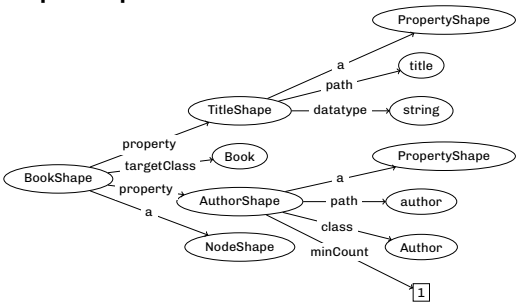
Validation result

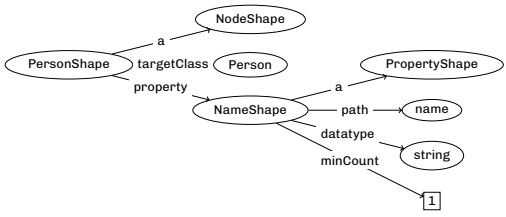
All properties described can be specified in a validation result.

sh:focusNode	Node that caused the result.
sh:resultPath	Pointing to value of **sh:path**
sh:value	Value node that violated constraint.
sh:sourceShape	Shape that given focus node validated against.
sh:sourceConstraintComponent	Constraint component that caused the result.
sh:detail	Parent result containing more details about the violation.
sh:message	Annotation property with textual details.
sh:severity	Default **sh:Violation**.

Validation example

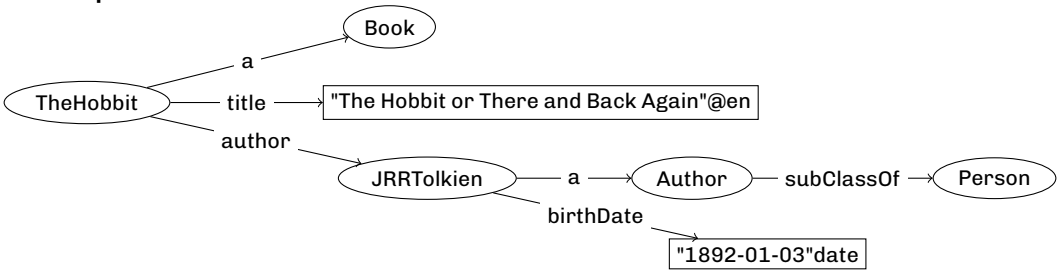
Shapes Graph





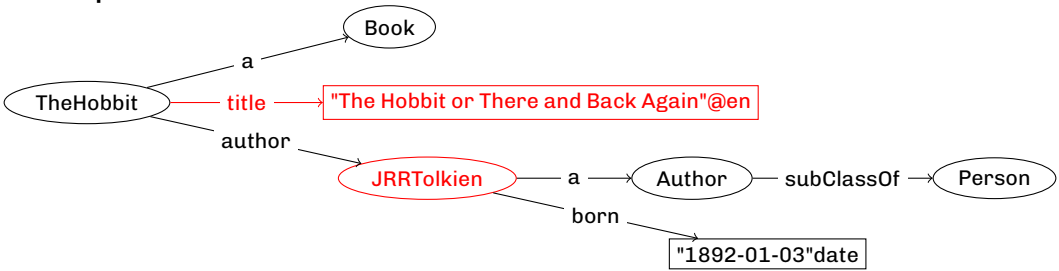
Validation example

Data Graph



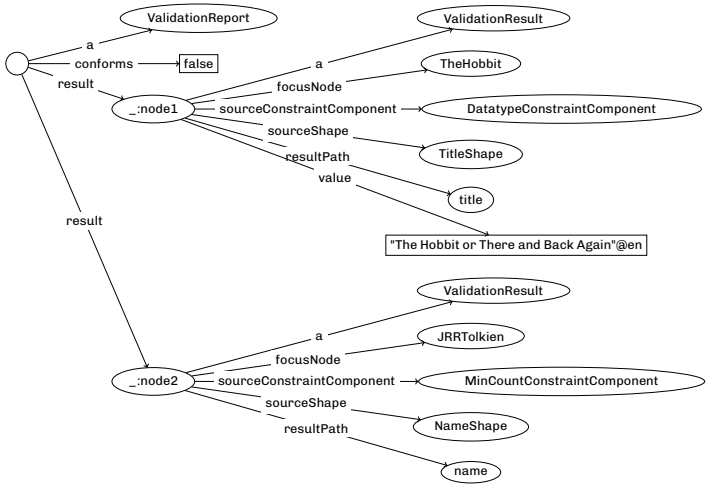
Validation example

Data Graph



Validation example

Validation Result (RDF4J)



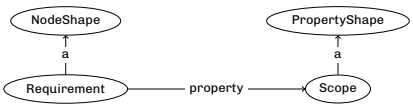
SHACL Implementations

Framework	
ruby-rdf/shacl	`https://github.com/ruby-rdf/shacl`
dotNetRDF	`https://dotnetrdf.org/docs/stable/api/VDS.RDF.Shacl.html`
pySHACL	`https://github.com/RDFLib/pySHACL`
RDF4J	`https://rdf4j.org/`
Jena	`https://jena.apache.org/`
Vendors	
TopQuadrant	`https://www.topquadrant.com/`
Stardog	`https://www.stardog.com/`
Cambridge Semantics	`https://cambridgesemantics.com/anzograph/`
Franz	`https://allegrograph.com`
Web playground	
SHACL Playground	`https://shacl.org/playground/`

...and more: `https://github.com/w3c-cg/awesome-semantic-shapes`

SHACL Stories

Regulatory Requirements



- The **Requirement** is the core at *every* activity by the Norwegian Maritime Authority.
- SHACL as verbose vocabulary for describing machine-readable requirements.
- CWA for the domain of law.

More on reasons why in *Using the Shapes Constraint Language for modelling regulatory requirements* by Veronika Heimsbakk and Kristian Torkelsen, `https://arxiv.org/abs/2309.02723`

Requirement

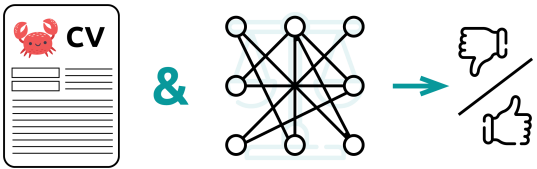
```
sdir:REG20140701955S1P1v0
  a sh:NodeShape ;
  a sdir:Requirement ;
  rdfs:label "Virkeområde"@no ;
  sh:property scope:PS_CargoShip ;
  sh:property scope:PS_ForeignTrade_PassengerShip ;
  sh:property scope:PS_MobileOffshoreUnit ;
  sh:property scope:PS_minLOA_24_LeisureBoat ;
  sdir:chapterTitle "Forskrift om radiokommunikasjonsutstyr for norske skip og flyttbare innretninger"@no ;
  sdir:eliReference "regulation/2014/07/01/955/part/1/chapter/1/section/1/subsection/1" ;
  sdir:generalScopes sdir:REG20140701955S1v0 ;
  sdir:regulationReference "https://lovdata.no/dokument/SF/forskrift/2014-07-01-955/§1" ;
  sdir:regulationTitle "Forskrift om radiokommunikasjonsutstyr for norske skip og flyttbare innretninger"@no ;
  sdir:takeEffect "2024-02-14"^^xsd:date ;
.
```

Scope

```
scope:PS_CargoShip
  a sh:PropertyShape ;
  a scope:Scope ;
  sh:path sdir:vesselType ;
  sh:description "Scope of vessel type cargo ship"@en ;
  sh:description "Virkeområde fartøytype lasteskip"@no ;
  sh:group scope:Scopes ;
  sh:hasValue vesseltype:CargoShip ;
  sh:name "Fartøytype lasteskip"@en ;
  sh:name "Fartøytype lasteskip"@no ;
  sh:name "Vessel type cargo ship"@en ;
.

scope:PS_minLOA_24_LeisureBoat
  a scope:CompoundScope ;
  sh:description "Compound scope of minLOA 24 LeisureBoat"@en ;
  sh:group scope:CompoundScopes ;
  sh:name "Length overall more than 24 m; vessel type leisure boat"@en ;
  sh:name "Største lengde større enn 24 m; fartøytype fritidsbåt"@no ;
  sh:property scope:PS_LeisureBoat ;
  sh:property scope:PS_minLOA_24 ;
.
```

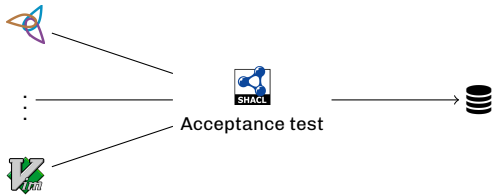
Issue Certificates




```
...
sh:or (
  [ sh:and ( # first alternative
    [ sh:or ( cert:D2A0 cert:D2B0 cert:D3A0
              cert:D3B0 cert:D4B0 cert:D4F0 ) ]
    [ sh:path nma:hasSeagoingServiceRequirement ;
      sh:hasValue nma:SGS_500_1080_D0 ; ]
  )]

  [ sh:and ( # second alternative
    [ sh:or ( cert:D2A0 cert:D2B0
              cert:D3A0 cert:D3B0 ) ]
    [ sh:path nma:hasSeagoingServiceRequirement ;
      sh:hasValue nma:SGS_500_720_D0 ; ]
    [ sh:path nma:hasSeagoingServiceRequirement ;
      sh:hasValue nma:SGS_500_360_C0 ; ]
  )]
) ;
...
```

Acceptance Testing

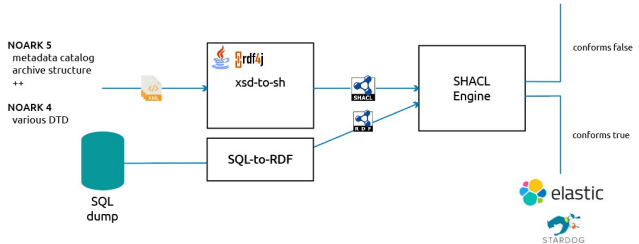



```
:RDFSClassShape a sh:NodeShape ;
  sh:targetClass rdfs:Class ;
  sh:property :RDFSLabelShape .

:RDFSLabelShape a sh:PropertyShape ;
  sh:path rdfs:label ;
  sh:minCount 1 ; sh:maxCount 1 ;
  sh:datatype rdf:langString .
```

- Shapes to validate the *structure* of the TBox.
- Included in the commit-pipeline, or outside if git is not used.
- Does not validate the *content* of the graph.

Schema



Journal post snippet

```
<jp/123> a :JournalPost ;
  :numAttachements 1 ;
  :documentMedium :ElectronicArchive ;
  :journalPostNumber "12/123-4"^^xsd:string ;
  ... .
```

SHACL snippet

```
[
  sh:datatype xsd:integer ;
  sh:maxCount 1 ;
  sh:predicate :numAttachements ;
]
```

Fun fact! The SHACL Engine implemented at eInnsyn led to the SHACL Engine for rdf4j.

Demo and data 👇

`https://github.com/veleda/shacl-masterclass`

References & resources

Images

My toy box & bookshelf	freepik.com

Around the web

W3C Recommendation	*Shape Constraint Language*	`https://www.w3.org/TR/shacl/`
Ivo Velitchkov and Veronika Heimsbakk	*SHACL Wiki*	`https://kvistgaard.github.io/shacl/`
Holger Knublauch	*SHACL and OWL Compared*	`https://spinrdf.org/shacl-and-owl.html`
W3C Working Group Note	*SHACL Advanced Features*	`https://w3c.github.io/shacl/shacl-af/`
TopQuadrant	*DASH Data Shapes*	`http://datashapes.org/`
`$this`		`https://github.com/veleda/shacl-masterclass`

Book

Jose Emilio labra Gayo, Eric Prud'hommeaux, Iovka Boneva, Dimitris Kontokostas, *Validating RDF Data*, 2018.