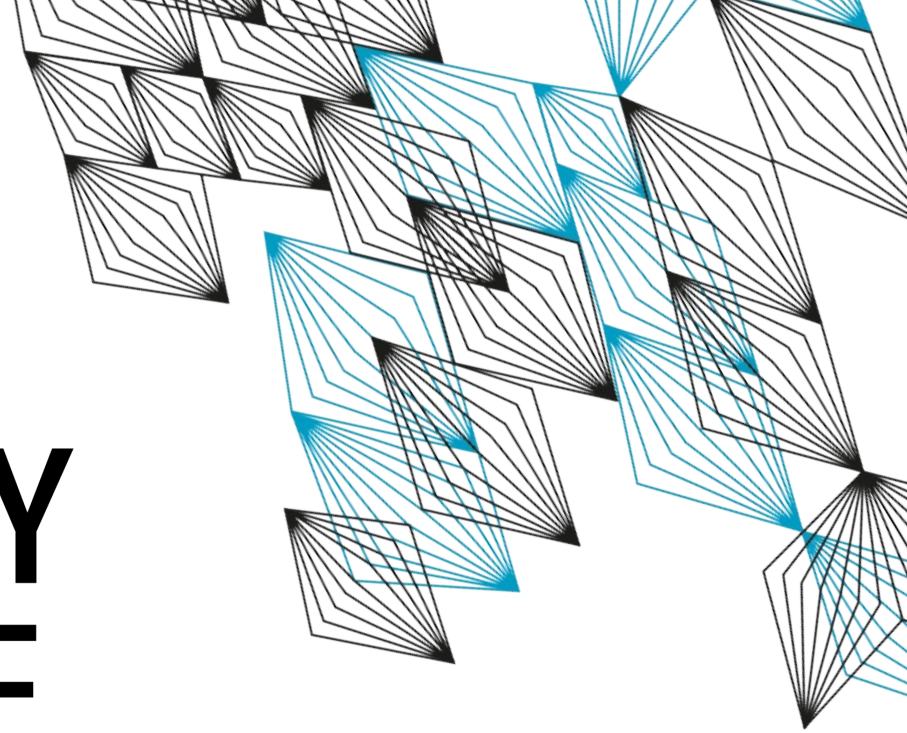
FACULTY OF COMPUTER SCIENCE

SERVICES AND CYBERSECURITY GROUP



UNIVERSITY OF TWENTE.

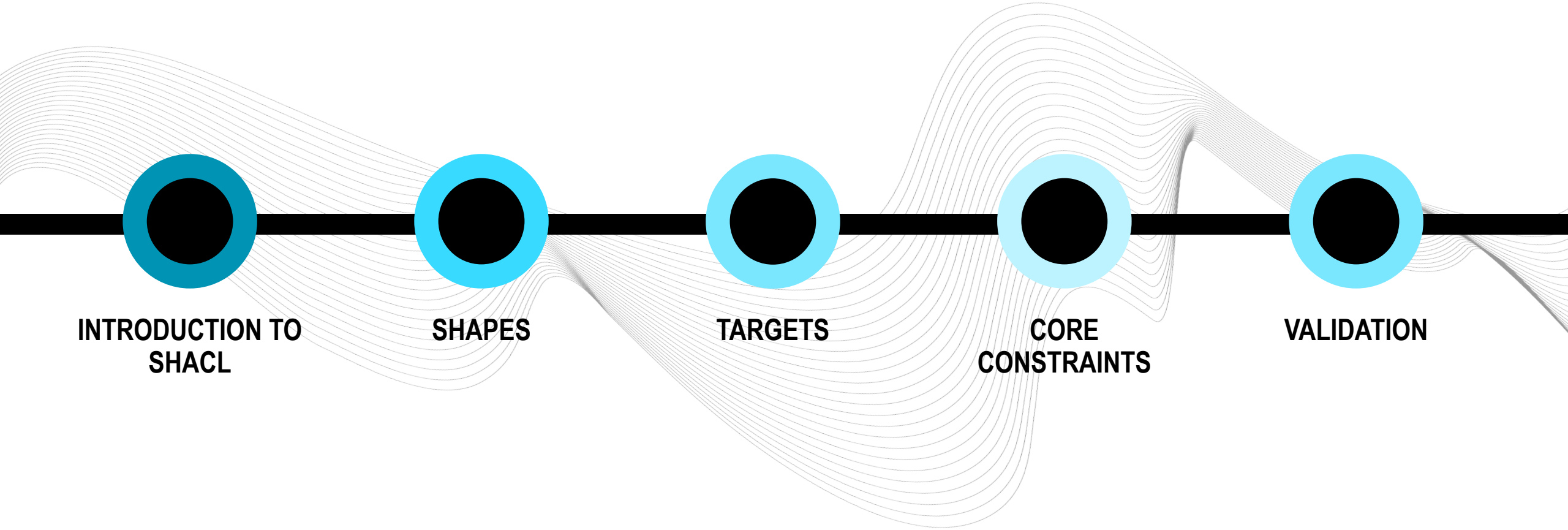
SHAPES CONSTRAINT LANGUAGE – SHACL

LINKED DATA AND SEMANTIC WEB

MARCH 29, 2022

LUIZ BONINO

IN THIS PRESENTATION:



UNIVERSITY OF TWENTE.

SHAPES CONSTRAINT LANGUAGE – SHACL

- W3C recommendation since 20 July 2017 - https://www.w3.org/TR/shacl/
- RDF language;
- Created to allow validation of RDF;
- A “schema” language for RDF;
- SHACL defines a “Shapes Graph” that is used to validate the “Data Graph”;
- Closed World Assumption;

SHACL VALIDATION

Shapes Graph

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix schema: <http://schema.org/> .

schema:Person
  a rdfs:Class, sh:NodeShape ;
  sh:property
    [
      sh:path schema:name ;
      sh:minCount 1 ;
      sh:maxCount 1 ;
    ],
    [
      sh:path schema:age ;
      sh:minCount 1 ;
      sh:minInclusive 18 ;
    ] ;
.

```

Data Graph

```
{
  "@context": {
    "@base": "https://example.com/",
    "@vocab": "http://schema.org/"
  },
  "@id": "John-Doe",
  "@type": "Person",
  "name": [
    "John",
    "Johnny"
  ],
  "age": 18
}

```

Validation Report

Success

No

Errors found

- https://example.com/John-Doe:
  - schema:name:
    - More than 1 values

https://shacl-playground.zazuko.com

RDF VALIDATION

Shapes Graph

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix schema: <http://schema.org/> .

schema:Person
  a rdfs:Class, sh:NodeShape ;
  sh:property
    [
      sh:path schema:name ;
      sh:minCount 1 ;
      sh:maxCount 1 ;
    ],
    [
      sh:path schema:age ;
      sh:minCount 1 ;
      sh:minInclusive 18 ;
    ] ;
.
```

Data Graph

```
{
  "@context": {
    "@base": "https://example.com/",
    "@vocab": "http://schema.org/"
  },
  "@id": "John-Doe",
  "@type": "Person",
  "name": [
    "John",
    "Johnny"
  ],
  "age": 18
}
```

Validation Report

Success

No

Errors found

- https://example.com/John-Doe:
  - schema:name:
    - More than 1 values

https://shacl-playground.zazuko.com

SHACL PROCESSOR

- Two inputs: a data graph (validation target) and a shapes graph (how to validate);
- SHACL processors must not change the graphs, i.e., both data and shapes graphs at the end of the validation must be identical to the graph at the beginning of validation;
- Generates a results graph;
- Can be a SHACL Core processor or SHACL SPARQL processor.

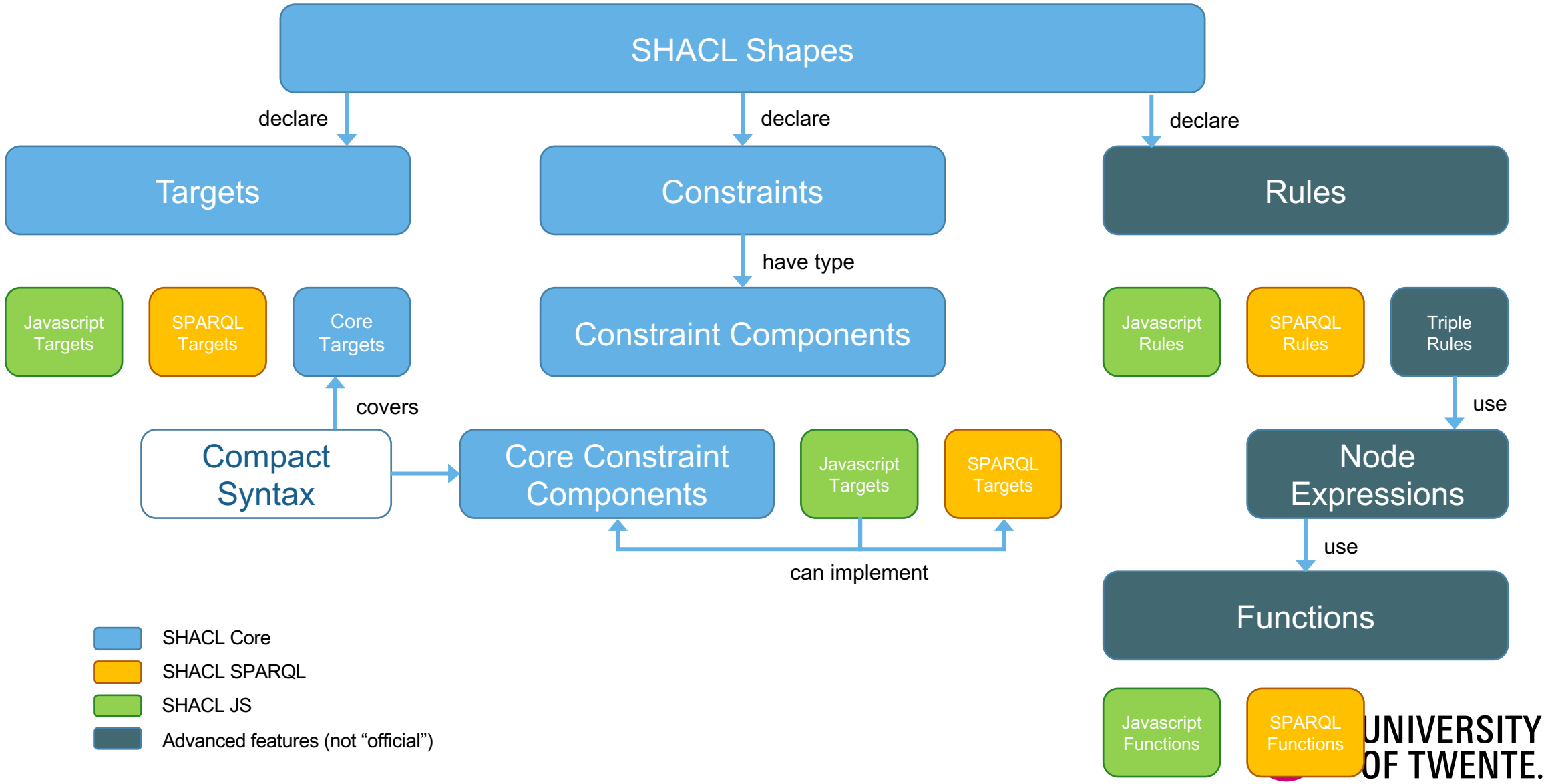
OTHER APPLICATIONS FOR SHACL

- Interface building, e.g., using with DASH;
- Data structure and semantics declaration (semantic data model specification);
- Code generation;
- Data integration;
- Rule-based inferencing;
- …

BRIEF HISTORY OF SHACL

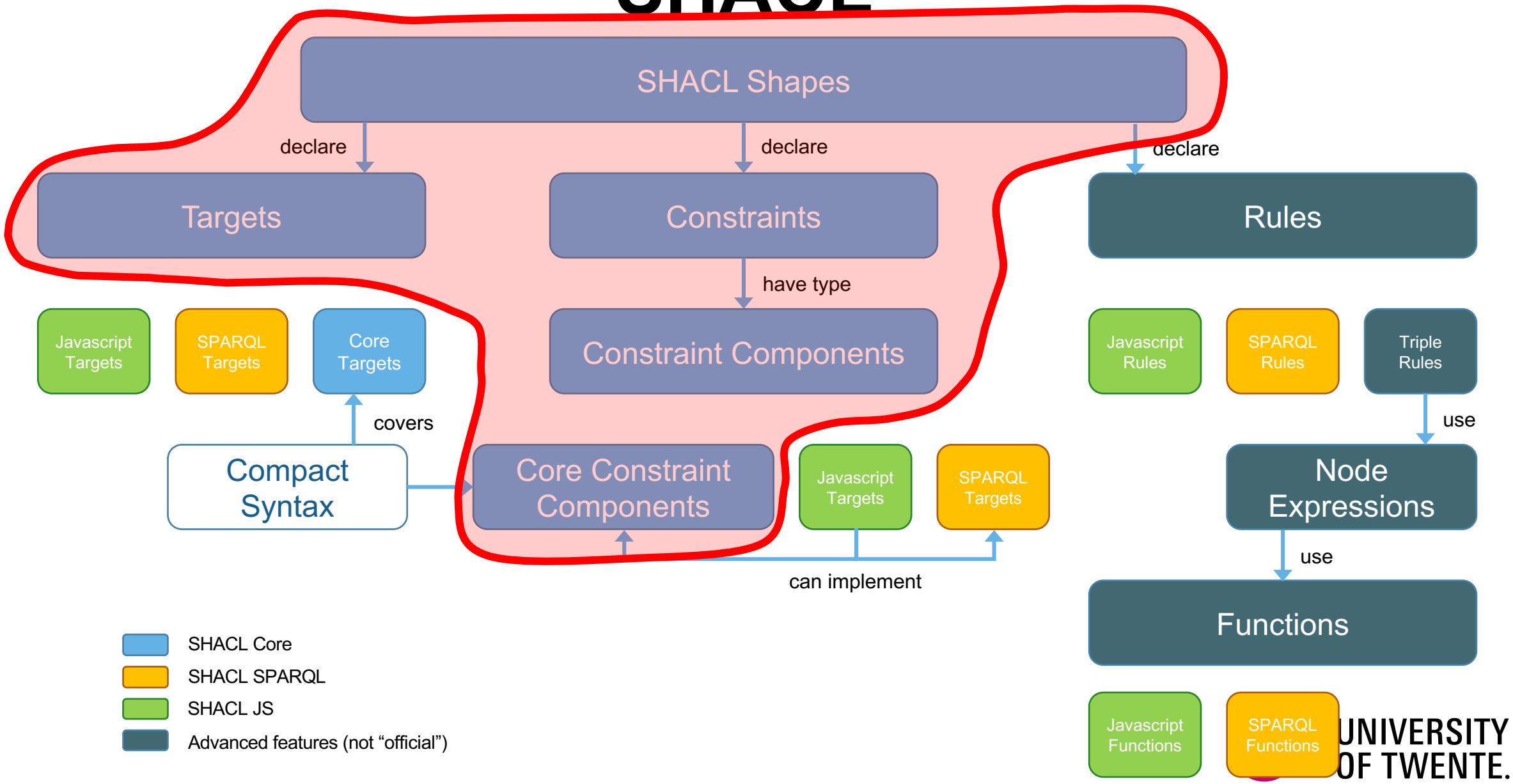
- SHAL Core Vocabulary and Syntax: inspired by IBM Resources Shapes;
- Architecture and extension mechanism: inspired by TopQuadrant's SPIN;
- Compact syntax: inspired by ShEx (Shape Expressions);
- Rules: inspired by SPIN;

SHACL



UNIVERSITY OF TWENTE.

SHACL



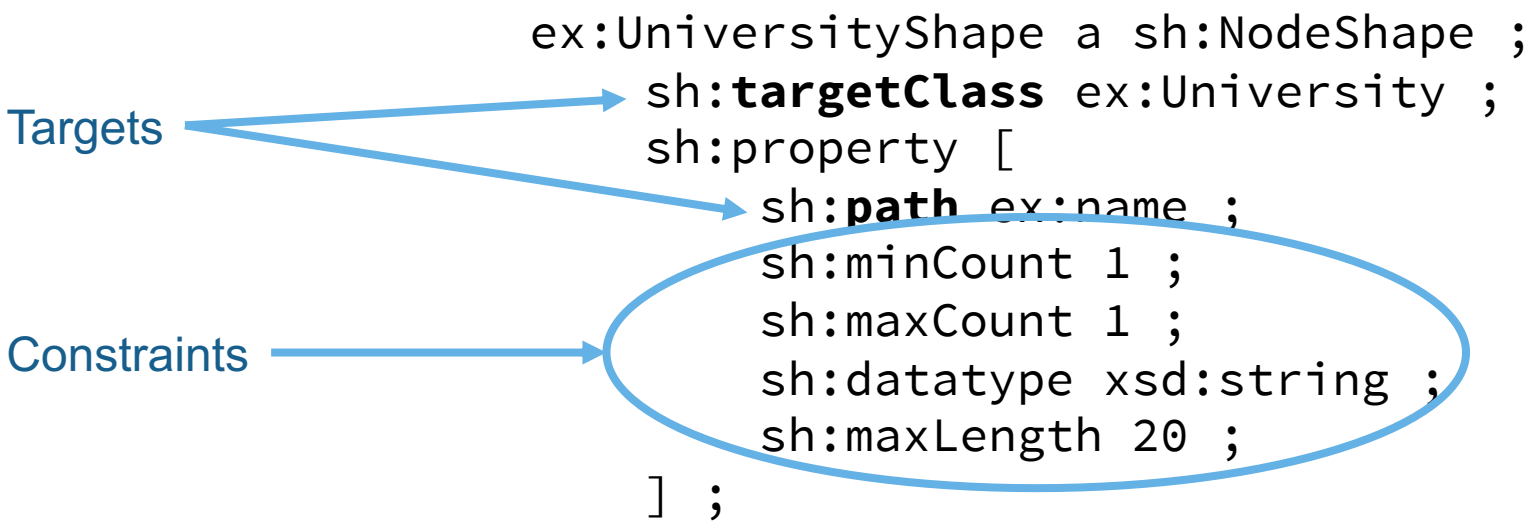
UNIVERSITY OF TWENTE.

WHAT IS A SHAPE?

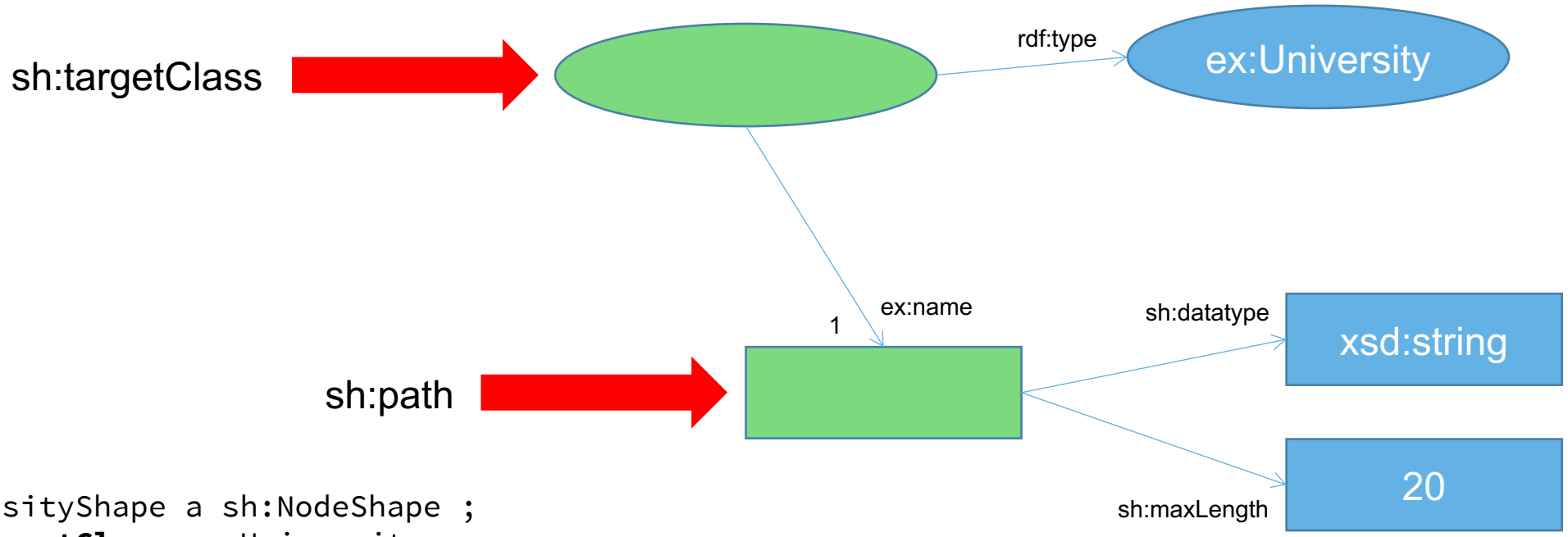
A collection of targets and contraints:

- Targets: define which nodes in the data graph must conform to the shape;
- Constraint: define how to validate a node;

EXAMPLE

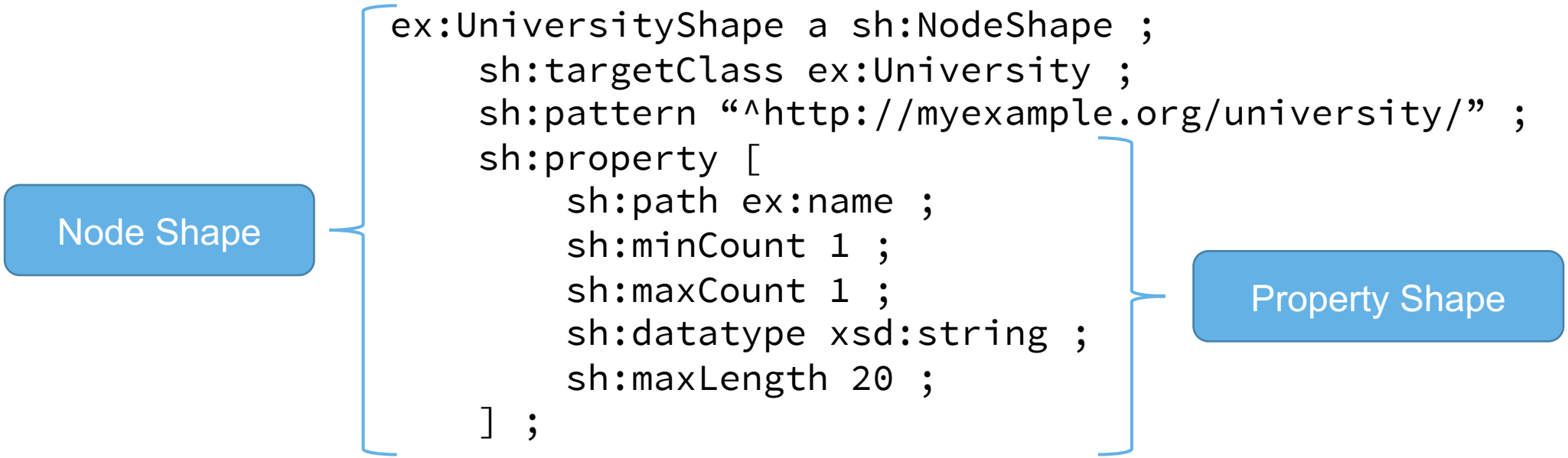

Targets
Constraints
ex:UniversityShape a sh:NodeShape ;
sh:targetClass ex:University ;
sh:property [
sh:path ex:name ;
sh:minCount 1 ;
sh:maxCount 1 ;
sh:datatype xsd:string ;
sh:maxLength 20 ;
] ;

EXAMPLE




```
ex:UniversityShape a sh:NodeShape ;
    sh:targetClass ex:University ;
    sh:property [
        sh:path ex:name ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:datatype xsd:string ;
        sh:maxLength 20 ;
    ] ;
```

EXAMPLE



Node shapes:
- Specify constraints on the target nodes (classes);
- Group property shapes;

Property shapes:
- Specify constraints on target properties and their values;

TARGETS

TARGET DECLARATIONS DEFINE WHAT RESOURCES (RDF GRAPH NODES) WILL BE THE FOCUS NODE FOR A SHAPE, E.G., WHAT THE SHAPE CONTRAINTS APPLY TO

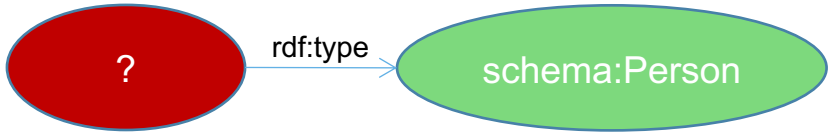
Target declarations:

- sh:targetClass: targets all resources that are instances of a given class;
- sh:targetNode: targets a specific resource, e.g., a given instance;
- sh:targetSubjectOf: targets all the subjects of a given predicate;
- sh:targetObjectOf: targets all the objects of a given predicate;

TARGETS

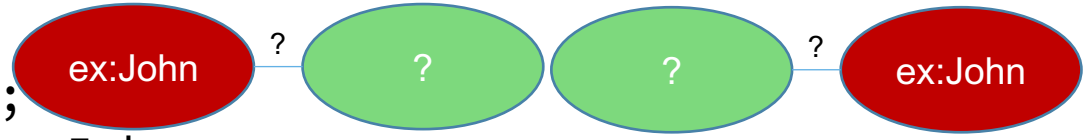
sh:targetClass:

```
:MyClassShape    a sh:NodeShape ;
                 sh:targetClass schema:Person;
```



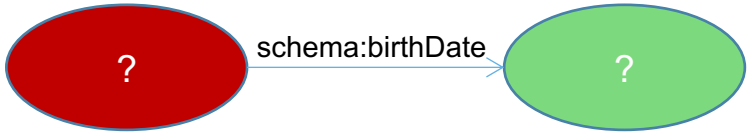
sh:targetNode:

```
:MyNodeShape     a sh:NodeShape ;
                 sh:targetNode ex:John;
```



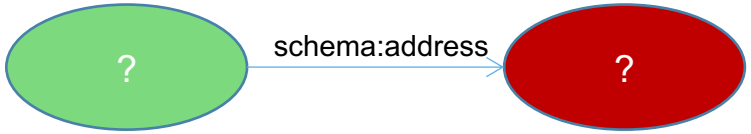
sh:targetSubjectOf:

```
:MySubjShape     a sh:NodeShape ;
                 sh:targetSubjectsOf schema:birthDate;
```



sh:targetObjectOf:

```
:MyObjShape      a sh:NodeShape ;
                 sh:targetObjectsOf schema:address;
```



UNIVERSITY OF TWENTE.

CORE CONSTRAINT COMPONENTS

Type	Constraints
Cardinality	minCount, maxCount
Types of values	class, datatype, nodeKind
Values	node, in, hasValue
Range of values	minInclusive, maxInclusive minExclusive, maxExclusive
String based	minLength, maxLength, pattern, languageIn, uniqueLang
Logical constraints	not, and, or, xone
Closed shapes	closed, ignoredProperties
Property pair constraints	equals, disjoint, lessThan, lessThanOrEquals
Non-validating constraints	name, description, group, order, defaultValue
Qualified shapes	qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount

CARDINALITY CONSTRAINTS

Constraint	Description
minCount	Restricts the minimun amount of occurences of a given property. Default value: 0
maxCount	Restricts the maximum amount of occurences of a given property. Default value: 0

```
:Person a sh:NodeShape, rdfs:Class ;
  sh:property [
    sh:path schema:knows ;
    sh:minCount 1;
    sh:maxCount 2;
  ] .
```

```
:john a schema:Person ;
      schema:knows :mary .

:mary a schema:Person ;
      schema:knows :peter ,
                   :anna ,
                   :fred .

:peter a schema:Person ;
      schema:givenName “Peter” .
```

DATATYPE OF VALUES CONSTRAINT

Constraint	Description
datatype	Restricts the datatype of all value nodes to a given value

```
:Person a sh:NodeShape, rdfs:Class ;
  sh:property [
    sh:path schema:birthDate ;
    sh:datatype xsd:date;
  ] .
```

```
:john a schema:Person ;
     schema:birthDate “1990-05-01”^^xsd:date .

:mary a schema:Person ;
     schema:birthDate “Unknown”^^xsd:date .

:peter a schema:Person ;
     schema:birthDate 1995 .
```

CLASS OF VALUES CONSTRAINT

Constraint	Description
class	Restricts to nodes that are instances of a given class.

```
:Person a sh:NodeShape, rdfs:Class ;
  sh:property [
    sh:path schema:knows ;
    sh:class :Person;
  ] .
```

```
:john a :Person ;
      schema:knows :mary .

:mary a :Person ;
      schema:knows :peter .

:peter a :Customer .
```

KINDS OF NODES CONSTRAINT

Constraint	Description
nodeKind	Possible values: BlankNode, IRI, Literal, BlankNodeOrIRI, BlankdNodeOrLiteral IRIOrLiteral

```
:Person a sh:NodeShape, rdfs:Class ;
  sh:nodeKind sh:IRI ;
  sh:property [
    sh:path schema:knows ;
    sh:nodeKind sh:BlankNodeOrIRI;
  ];
  sh:property [
    sh:path schema:name ;
    sh:nodeKind sh:BlankNodeOrLiteral;
  ] .
```

```
:john a :Person ;
  schema:knows :mary ;
  schema:name :johnsName .

:mary a :Person ;
  schema:knows [ schema:name “Fred”];
  schema:name _:1 .

:peter a :Person ;
  schema:knows :john ;
  schema:name “Peter” .

_:1 a :Person
```

CONSTRAINT ON VALUES

Constraint	Description
hasValue	Verifies whether the focus node has a given value
in	Enumerates the value options that a property may have

```
:Person a sh:NodeShape, rdfs:Class ;
  sh:property [
    sh:path schema:affiliation ;
    sh:hasValue :OurCompany;
  ];
  sh:property [
    sh:path schema:gender ;
    sh:in (schema:Male schema:Female);
  ] .
```

```
:john a :Person ;
  schema:affiliation :OurCompany ;
  schema:gender schema:Male.

:mary a :Person ;
  schema:affiliation “OurCompany” ;
  schema:gender schema:Female .

:peter a :Person ;
  schema:affiliation :OurCompany ;
  schema:gender schema:Unknown .
```

CONSTRAINT ON VALUES WITH ANOTHER SHAPE

Constraint	Description
node	All values of a given property must comply with a given shape

```
:Person a sh:NodeShape, rdfs:Class ;
  sh:property [
    sh:path schema:affiliation ;
    sh:node :Company;
  ].

:Company a sh:Shape ;
  sh:property [
    sh:path schema:name ;
    sh:datatype xsd:string ;
  ] .
```

```
:john a :Person ;
  schema:affiliation :OurCompany .

:mary a :Person ;
  schema:affiliation :Another .

:OurCompany schema:name “OurCompany”.

:Another schema:name 10 .
```

VALUE RANGES

Constraint	Description
minInclusive	The minimum inclusive value
maxInclusive	The maximum inclusive value
minExclusive	The minimum exclusive value
maxExclusive	The maximum exclusive value

```
:Rating a sh:NodeShape, rdfs:Class;
  sh:property [
    sh:path schema:ratingValue ;
    sh:minInclusive 1;
    sh:maxExclusive 6;
    sh:datatype xsd:integer;
  ] .
```

```
:bad a :Rating ;
       schema:ratingValue 1 .

:veryGood a :Rating ;
       schema:ratingValue 5 .

:zero a :Rating ;
       schema:ratingValue 0 .

:six a :Rating ;
       schema:ratingValue 6 .
```

STRING-BASED CONSTRAINTS (1)

Constraint	Description
minLength	Restricts the minimum string length on value nodes
maxLength	Restricts the maximum string length on value nodes

```
:Person a sh:NodeShape, rdfs:Class;
  sh:property [
    sh:path schema:name ;
    sh:minLength 4;
    sh:maxLength 15;
] .
```

```
:ana a :Person ;
       schema:name “Ana” .

:anna a :Person ;
       schema:name “Anna” .

:peter a :Person
       schema:name “Peter” .
```

STRING-BASED CONSTRAINTS (2)

Constraint	Description
pattern	Checks if the string value matches a regular expression

```
:Product a sh:NodeShape, rdfs:Class;
  sh:property [
    sh:path schema:productID ;
    sh:pattern “^P” ;
    sh:flags “i” ;
] .
```

```
:car a :Product ;
       schema:productID “P1234” .

:boat a :Product ;
       schema:productID “p567” .

:bike a :Product ;
       schema:productID “B123” .
```

STRING-BASED CONSTRAINTS (3)

Constraint	Description
uniqueLang	Checks that no pair of nodes use the same language tag
languageIn	Checks that the property values are expressed in the given list of languages

```
:Country a sh:NodeShape, rdfs:Class;
  sh:property [
    sh:path schema:name ;
    sh:languageIn (“en” “es”) ;
    sh:uniqueLang true ;
] .
```

```
:spain a ex:Country ;
       schema:name “Spain”@en ,
                   “España”@es .

:france a ex:Country ;
       schema:name “France”@en,
                   “Francia”@es .

:usa a ex:Country ;
       schema:name “USA”@en ,
                   “United States”@en .
```

LOGICAL CONSTRAINTS

Constraint	Description
and	Conjunction of a list of shapes

```
:Person a sh:NodeShape;
  sh:and (
    [ sh:property [
        sh:path schema:name ;
        sh:minCount 1;
      ]
    ]
    [ sh:property [
        sh:path schema:affiliation ;
        sh:MinCount 1;
      ]
    ]
  ) .
```

≡

```
:Person a sh:NodeShape;
  [ sh:property [
    sh:path schema:name ;
    sh:minCount 1;
    ]
  ]
  [ sh:property [
    sh:path schema:affiliation ;
    sh:MinCount 1;
    ]
  ] .
```

LOGICAL CONSTRAINTS

Constraint	Description
not	Negation of a shape

```
:Person a sh:NodeShape, rdfs:Class;
  sh:not
    [ sh:path foaf:name ;
      sh:minCount 1;
    ] .
```

```
:john a :Person ;
        schema:name “John” .

:mary a :Person ;
        foaf:name “Mary” .

:peter a :Person ;
        rdfs:label “Peter”.
```

LOGICAL CONSTRAINTS

Constraint	Description
or	Each node conforms to at least one of the provided shapes or both

```
:Person a sh:NodeShape, rdfs:Class;
  sh:or (
    [ sh:path foaf:name ;
      sh:minCount 1 ;
    ]
    [ sh:path schema:name ;
      sh:minCount 1 ;
    ]
  ) .
```

```
:john a :Person ;
  schema:name “John” ;
  foaf:name “John” .

:mary a :Person ;
  foaf:name “Mary” .

:peter a :Person ;
  rdfs:label “Peter”.
```

LOGICAL CONSTRAINTS

Constraint	Description
xone	Specifies a condition that each value node conforms to exactly one of the provided shapes (not both)

```
:Person a sh:NodeShape, rdfs:Class;
  sh:xone (
    [ sh:property [
        sh:path ex:fullName ;
        sh:minCount 1;
      ]
    ]
    [
      sh:property [
        sh:path ex:firstName ;
        sh:minCount 1 ;
      ] ;
      sh:property [
        sh:path ex:lastName ;
        sh:minCount 1 ;
      ]
  ]
).
```

```
:john a :Person ;
  ex:firstName “John” ;
  ex:lastName “Doe” .

:mary a :Person ;
  ex:fullName “Mary Jane” .

:peter a :Person ;
  ex:firstName “Peter” ;
  ex:lastName “Parker” ;
  ex:fullName “Peter Parker” .
```

CLOSED CONSTRAINTS

Constraint	Description
closed	If set to true, properties that are not explicitly described in the shape are not allowed
ignoredProperties	Optional list of properties that are also permitted in addition to those explicitly enumerated in the shape

```
ex:Person a sh:NodeShape, rdfs:Class;
  sh:closed true ;
  sh:ignoredProperties (rdf:type) ;
  sh:property [
    sh:path ex:firstName ;
  ] ;
  sh:property [
    sh:path ex:lastName ;
  ] .
```

```
ex:john a ex:Person;
        ex:firstName “John” .

ex:Mary a ex:Person;
        ex:firstName “Mary” ;
        ex:middleInitial “J” .
```

PROPERTY PAIR CONSTRAINTS

Constraint	Description
equals	The sets of values of both properties at a given focus node must be equal
disjoint	The sets of values of both properties at a given focus node must be different
lessThan	The values must be smaller than the values of another property
lessThanOrEquals	The values must be smaller or equal than the values of another property

```
:Person a sh:NodeShape, rdfs:Class;
  sh:property [
    sh:path schema:givenName ;
    sh:equals foaf:firstName ;
  ] ;
  sh:property [
    sh:path schema:givenName ;
    sh:disjoint schema:lastName ;
  ] .
```

```
ex:John  schema:givenName  “John”;
         schema:lastName   “Doe”;
         foaf:firstName    “John”;

ex:Mary  schema:givenName  “Mary” ;
         schema:lastName   “Mary” ;
         foaf:firstName    “Marianne” .
```

NON VALIDATING CONSTRAINTS (1)

Constraint	Description
name	Provide human-readable labels for a property
description	Provide a description of a property
order	Relative order of the property
group	Group several constraints together

```
:Person a sh:NodeShape;
  sh:property [
    sh:path schema:url;
    sh:name “URL”;
    sh:description “User URL”;
    sh:order 1;
  ] ;
  sh:property [
    sh:path schema:name;
    sh:name “Name”;
    sh:description “Person name”;
    sh:order 2;
  ] .
```

NON VALIDATING CONSTRAINTS (2)

Constraint	Description
name	Provide human-readable labels for a property
description	Provide a description of a property
order	Relative order of the property
group	Group several constraints together

```
:Person a sh:NodeShape;
  sh:property [
    sh:path schema:url;
    sh:name “URL”;
    sh:group :userDetails;
  ] ;
  sh:property [
    sh:path schema:name;
    sh:name “Name”;
    sh:group :userDetails;
  ]
  sh:property [
    sh:path schema:address;
    sh:name “Address”;
    sh:group :location;
  ]
  sh:property [
    sh:path schema:country;
    sh:name “Country”;
    sh:group :location
  ] .
```

```
:userDetails a
sh:PropertyGroup;
  sh:order 0;
  rdfs:label “User details”.

:location a sh:PropertyGroup;
  sh:order 1;
  rdfs:label “Location” .
```

Possible generated form:

```
User details
  URL:  _______________________________
  Name: _______________________________

Location
  Address: ____________________________
  Country: ____________________________
```

NON VALIDATING CONSTRAINTS (3)

Constraint	Description
defaultValue	Define the default value of a given shape

```
:Person a sh:NodeShape;
  sh:property [
    sh:path schema:url;
    sh:name “URL”;
    sh:group :userDetails;
  ] ;
  sh:property [
    sh:path schema:name;
    sh:name “Name”;
    sh:group :userDetails;
  ]
  sh:property [
    sh:path schema:address;
    sh:name “Address”;
    sh:group :location;
  ]
  sh:property [
    sh:path schema:country;
    sh:name “Country”;
    sh:group :location;
    sh:defaultValue “UK”;
    ] .
:userDetails a
sh:PropertyGroup;
  sh:order 0;
  rdfs:label “User details”.

:location a sh:PropertyGroup;
  sh:group 1;
  rdfs:label “Location” .
```

Possible generated form:

```
User details
  URL:  _______________________________
  Name: _______________________________

Location
  Address: ____________________________
  Country: _UK_________________________
```

QUALIFIED SHAPES

Constraint	Description
sh:qualifiedValueShape	The shape that the specified number of nodes needs to conform to
sh:qualifiedValueShapesDisjoint	If set to `true`, for the counting, the nodes must not conform to any of the sibling shapes
sh:qualifiedMinCount	Minimum number of nodes that conform to the shape
sh:qualifiedMaxCount	Maximum number of nodes that conform to the shape

```
:QualifiedExampleShape a sh:NodeShape;
  sh:targetNode :Offspring ;
  sh:property [
    sh:path :parent;
    sh:minCount 2;
    sh:maxCount 2;
    sh:qualifiedValueShape [
      sh:path :gender ;
      sh:hasValue :Female ;
    ] ;
    sh:qualifiedMinCount 1 ;
  ] .
```

```
:Offspring
  :parent :John ;
  :parent :Mary .

:John :gender :Male .

:Mary :gender :Female .
```

DECLARING THE SEVERITY OF A SHAPE

- Shapes can specify one value for the property `sh:severity` in the shapes graph;
- Three types of severities are defined:
  - `sh:Info` – A non-critical constraint violation indicating an informative message;
  - `sh:Warning` – A non-critical constraint violation indicating a warning;
  - `sh:Violation` – A critical constraint violation;

```
ex:Person a sh:NodeShape, rdfs:Class ;
  sh:property [
    sh:path ex:nickName ;
    sh:minCount 1;
    sh:datatype xsd:string;
    sh:severity sh:Warning;
  ] .
```

DECLARING MESSAGES FOR A SHAPE

- A shape can have one value for the property `sh:message` with the same language tag;
- Messages are used to provide a customised information in the case validation violation.

```
ex:Person a sh:NodeShape, rdfs:Class ;
  sh:property [
    sh:path ex:firstName ;
    sh:maxLength 15;
    sh:datatype xsd:string;
    sh:message “Too many characters”@en;
    sh:message “Zu vielen Zeichen”@de;
    sh:message “Limite de caracters excedido”@pt;
  ] .
```

VALIDATION RESULTS VOCABULARY

Concept	Description
sh:ValidationReport	Result of the validation process that reports the conformance and the set of all validation results
sh:conformsTo	`true` if a focus node conforms with a given shape and `false` otherwise
sh:result	Contains the result graph for each validation result
sh:ValidationResult	Represents each individual validation result
sh:focusNode	The node which was tested and caused the validation result
sh:resultPath	Points to a well-formed SHACL property path
sh:value	Points to the value of the property that caused the result
sh:sourceShape	Points to the shape that the given focus node was validated against
sh:sourceConstraintComponent	Refers to which constraint component has been violated
sh:detail	May point to further details about the cause of the violation
sh:resultMessage	Textual details about the error. The message can be affected by the `sh:message` property in the shapes graph
sh:resultSeverity	A value which is equal to the `sh:severity` value of the shape that caused the violation error. The default value is `sh:Violation`

VALIDATION EXAMPLE

Shapes graph

```
:Person a sh:NodeShape, rdfs:Class ;
  sh:property [
    sh:path schema:birthDate ;
    sh:datatype xsd:date;
  ] ;
  sh:property [
    sh:path schema:givenName ;
    sh:disjoint schema:lastName ;
  ] .
```

Data graph

```
ex:john a ex:Person ;
        schema:birthDate "2000-05-15"^^xsd:date ;
        schema:givenName "John" ;
        schema:lastName "Doe" .

ex:mary a ex:Person ;
        schema:birthDate "4 July 2005" ;
        schema:givenName "Mary" ;
        schema:lastName "Mary" .
```

Results graph

```
_:report a sh:ValidationReport ;
  sh:conforms false
  sh:result [
    rdf:type sh:ValidationResult ;
    sh:resultSeverity sh:Violation ;
    sh:sourceConstraintComponent sh:DatatypeConstraintComponent ;
    sh:sourceShape [
      sh:path schema:birthDate ;
      sh:datatype xsd:date ;
    ] ;
    sh:focusNode <http://example.com#mary> ;
    sh:resultPath schema:birthDate ;
    sh:value "4 July 2005" ;
    sh:resultMessage "Value does not have datatype
<http://www.w3.org/2001/XMLSchema#date>" ;
  ], [
    rdf:type sh:ValidationResult ;
    sh:resultSeverity sh:Violation ;
    sh:sourceConstraintComponent sh:DisjointConstraintComponent ;
    sh:sourceShape [
      sh:path schema:givenName ;
      sh:disjoint schema:lastName ;
    ] ;
    sh:focusNode <http://example.com#mary> ;
    sh:resultPath schema:givenName ;
    sh:value "Mary" ;
    sh:resultMessage "Value node must not also be one of the
values of <http://schema.org/lastName>" ;
  ] .

sh:Violation a sh:Severity .
```

VALIDATION EXAMPLE

Shapes graph

```
:Person a sh:NodeShape, rdfs:Class ;
  sh:property [
    sh:path schema:birthDate ;
    sh:datatype xsd:date;
  ] ;
  sh:property [
    sh:path schema:givenName ;
    sh:disjoint schema:lastName ;
  ] .
```

Data graph

```
ex:john a ex:Person ;
        schema:birthDate "2000-05-15"^^xsd:date ;
        schema:givenName "John" ;
        schema:lastName "Doe" .

ex:mary a ex:Person ;
        schema:birthDate "4 July 2005" ;
        schema:givenName "Mary" ;
        schema:lastName "Mary" .
```

Results graph

```
_:report a sh:ValidationReport ;
  sh:conforms false
  sh:result [
    rdf:type sh:ValidationResult ;
    sh:resultSeverity sh:Violation ;
    sh:sourceConstraintComponent sh:DatatypeConstraintComponent ;
    sh:sourceShape [
      sh:path schema:birthDate ;
      sh:datatype xsd:date ;
    ] ;
    sh:focusNode <http://example.com#mary> ;
    sh:resultPath schema:birthDate ;
    sh:value "4 July 2005" ;
    sh:resultMessage "Value does not have datatype
<http://www.w3.org/2001/XMLSchema#date>" ;
  ], [
    rdf:type sh:ValidationResult ;
    sh:resultSeverity sh:Violation ;
    sh:sourceConstraintComponent sh:DisjointConstraintComponent ;
    sh:sourceShape [
      sh:path schema:givenName ;
      sh:disjoint schema:lastName ;
    ] ;
    sh:focusNode <http://example.com#mary> ;
    sh:resultPath schema:givenName ;
    sh:value "Mary" ;
    sh:resultMessage "Value node must not also be one of the
values of <http://schema.org/lastName>" ;
  ] .

sh:Violation a sh:Severity .
```

VALIDATION EXAMPLE

Shapes graph

```
:Person a sh:NodeShape, rdfs:Class ;
  sh:property [
    sh:path schema:birthDate ;
    sh:datatype xsd:date;
  ] ;
  sh:property [
    sh:path schema:givenName ;
    sh:disjoint schema:lastName ;
  ] .
```

Data graph

```
ex:john a ex:Person ;
        schema:birthDate "2000-05-15"^^xsd:date ;
        schema:givenName "John" ;
        schema:lastName "Doe" .

ex:mary a ex:Person ;
        schema:birthDate "4 July 2005" ;
        schema:givenName "Mary" ;
        schema:lastName "Mary" .
```

Results graph

```
_:report a sh:ValidationReport ;
  sh:conforms false
  sh:result [
    rdf:type sh:ValidationResult ;
    sh:resultSeverity sh:Violation ;
    sh:sourceConstraintComponent sh:DatatypeConstraintComponent ;
    sh:sourceShape [
      sh:path schema:birthDate ;
      sh:datatype xsd:date ;
    ] ;
    sh:focusNode <http://example.com#mary> ;
    sh:resultPath schema:birthDate ;
    sh:value "4 July 2005" ;
    sh:resultMessage "Value does not have datatype
<http://www.w3.org/2001/XMLSchema#date>" ;
  ], [
    rdf:type sh:ValidationResult ;
    sh:resultSeverity sh:Violation ;
    sh:sourceConstraintComponent sh:DisjointConstraintComponent ;
    sh:sourceShape [
      sh:path schema:givenName ;
      sh:disjoint schema:lastName ;
    ] ;
    sh:focusNode <http://example.com#mary> ;
    sh:resultPath schema:givenName ;
    sh:value "Mary" ;
    sh:resultMessage "Value node must not also be one of the
values of <http://schema.org/lastName>" ;
  ] .

sh:Violation a sh:Severity .
```

UNIVERSITY OF TWENTE.

Leiden University Medical Center

Luiz Bonino

Associate Services and Cybersecurity – U. Twente
Associate Professor BioSemantics – LUMC

E-mail: l.o.boninodasilvasantos@utwente.nl