SHACL package regarding TQC

This project is to provide a package of SHACL shapes to BFO users. The users can use one or more SHACL shapes in the packege to check a given database, acquiring relevant information regarding temporal qualified value (TQV).

1. The goal: the users can validate that, in a given BFO database,
    1.1 whether a given node (continuant) has TQV, where TQV can be expressed in one or more of the following ways.
        1.1.1 A value in the zero-dimensional temporal region, the data type of the value is xsd: date.
        1.1.2 A value in the one-dimensional temporal region, where it has either the starting date or the ending date, and the data type of the starting date and the ending date is xsd: date, xsd: gYear, xsd: gYearMonth, xsd: gMonth, xsd:gMonthDay, or xsd: gDay.
        1.1.3 Temporally associated with any other node (continuant or occurrent) which has TQV. The notion of temporally associated is described later in the third section.
    1.2 Whether the TQV of the given node C (continuant) - which has been validated as having TQV- is merely a value in the zero-dimensional temporal region. If not, whether its TQV value has starting date or/and ending date.
    1.3 Whether nodes in a given class have TQV respectively, and whether these nodes can be sorted in temporal order.
2. The strategy:

2.1 BFO 2020 has relation types that are directly or indirectly associated with temporal regions. So, SHACL can be used to judge whether a given node has TQV.

2.2 Based on the above strategy, SHACL can be used to validate whether the nodes in a target class have TQV.

2.3 A SHACL shape can be designed to sort nodes in a target class with TQV in temporal order, when these nodes only have TQV with a value in the zero-dimensional temporal region.

2.4 When the nodes in a target class have TQV with a value in the one-dimensional temporal region, and the value includes starting date, then a SHACL shape can be designed to sort these nodes in temporal order in terms of the starting date.

2.5 When the nodes in a target class have TQV with a value in the one-dimensional temporal region, and the value includes ending date, then a SHACL shape can be designed to sort these nodes in temporal order in terms of the ending date.

3. Steps.

3.1 SHACL TQV-I. We design a SHACL shape to validate whether a given node is a subject of exist_at, or occupies_temporal_region, where the object of exist_at or occupies_temporal_region is xsd: date, xsd: gYear, xsd: gYearMonth, xsd: gMonth, xsd:gMonthDay, or xsd: gDay.

3.2 SHACL TQV-II. We design a SHACL shape to validate whether a given node is one of the following:

3.2.1 a subject of has_history, where the object of has_history has a property with value of xsd: date, xsd: gYear, xsd: gYearMonth, xsd: gMonth, xsd:gMonthDay, or xsd: gDay.

3.2.2 a subject of participates_in_at_some_time, where the object has a property with value of xsd: date, xsd: gYear, xsd: gYearMonth, xsd: gMonth, xsd:gMonthDay, or xsd: gDay.

3.2.3 a subject of has_realization, where the object has a property with value of xsd: date, xsd: gYear, xsd: gYearMonth, xsd: gMonth, xsd:gMonthDay, or xsd: gDay.

3.2.4 a subject of environs, where the object has a property with value of xsd: date, xsd: gYear, xsd: gYearMonth, xsd: gMonth, xsd:gMonthDay, or xsd: gDay.

3.2.5 an object of has_participant_at_some_time, where the subject has a property with value of xsd: date, xsd: gYear, xsd: gYearMonth, xsd: gMonth, xsd:gMonthDay, or xsd: gDay.

3.2.6 an object of history_of, where the subject has a property with value of xsd: date, xsd: gYear, xsd: gYearMonth, xsd: gMonth, xsd:gMonthDay, or xsd: gDay.

3.2.7 an object of realizes, where the subject has a property with value of xsd: date, xsd: gYear, xsd: gYearMonth, xsd: gMonth, xsd:gMonthDay, or xsd: gDay.

3.2.8 an object of occurs_in, where the subject has a property with value of xsd: date, xsd: gYear, xsd: gYearMonth, xsd: gMonth, xsd:gMonthDay, or xsd: gDay.

Noting: in this project, we just define ‘a property with value of xsd: date, xsd: gYear, xsd: gYearMonth, xsd: gMonth, xsd:gMonthDay, or xsd: gDay’ as the data

type of the object of the property is xsd: date, xsd: gYear, xsd: gYearMonth, xsd: gMonth, xsd:gMonthDay, or xsd: gDay.

3.3 SHACL TQV-III. We design a SHACL shape that first validates whether the nodes in a class with TQV and their TQVs are the value in zero-dimensional temporal region; second, sorts the nodes in orders in terms of their temporal region values.

3.4 SHACL TQV-IV. We design a SHACL shape that first validates whether the nodes in a class with TQV, their TQVs are the value in one-dimensional temporal region, and their TQV values have starting date; second, sorts the nodes in orders in terms of their starting dates.

Note: in this project, we just define that TQVs with one-dimensional temporal region are just contained in has_history and history_of, and that starting date is birth date.

3.5 SHACL TQV-V. We design a SHACL shape that first validates whether the nodes in a class with TQV, their TQVs are the value in one-dimensional temporal region, and their TQV values have ending date; second, sorts the nodes in orders in terms of their ending dates.

Note: in this project, we just define that TQVs with one-dimensional temporal region are just contained in has_history and history_of, and that starting date is death date.

The SHACL shade part:

SHACL TQV-I.

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

# Define the shape for the nodes that are subjects of exist_at or occupies_temporal_region
ex:TemporalShape
```

```
    a sh:NodeShape ;
    sh:targetClass ex:TemporalNode ;
    sh:property [       sh:path ex:exist_at ;       sh:or (             [ sh:datatype xsd:date ]
          [ sh:datatype xsd:gYear ]
          [ sh:datatype xsd:gYearMonth ]
          [ sh:datatype xsd:gMonth ]
          [ sh:datatype xsd:gMonthDay ]
          [ sh:datatype xsd:gDay ]
       ) ;
       sh:maxCount 1 ; # Exist_at can have only one value
    ] ;
    sh:property [       sh:path ex:occupies_temporal_region ;       sh:or (             [ sh:datatype
xsd:date ]
          [ sh:datatype xsd:gYear ]
          [ sh:datatype xsd:gYearMonth ]
          [ sh:datatype xsd:gMonth ]
          [ sh:datatype xsd:gMonthDay ]
          [ sh:datatype xsd:gDay ]
       ) ;
       sh:maxCount 1 ; # Occupies_temporal_region can have only one value
    ] .

  # Define the target class of the shape
  ex:TemporalNode
    a rdfs:Class .
```

SHACL TQV-II.

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX sh: <http://www.w3.org/ns/shacl#>

  # A shape that checks if a node satisfies any of the conditions 1-8
  # specified in the question.
  ex:DateValidationShape a sh:NodeShape ;
    sh:targetClass ?targetClass ;
    sh:or (
     [sh:property [
        sh:path (has_history) ;
        sh:node [
          sh:property [
            sh:path ?propertyPath ;
```

```
            sh:datatype (xsd:date xsd:gYear xsd:gYearMonth xsd:gMonth xsd:gMonthDay
xsd:gDay)
           ]
         ]
       ]
     ]
     [sh:property [
        sh:path (participates_in_at_some_time) ;
        sh:node [
          sh:property [
            sh:path ?propertyPath ;
            sh:datatype (xsd:date xsd:gYear xsd:gYearMonth xsd:gMonth xsd:gMonthDay
xsd:gDay)
           ]
         ]
       ]
     ]
     [sh:property [
        sh:path (has_realization) ;
        sh:node [
          sh:property [
            sh:path ?propertyPath ;
            sh:datatype (xsd:date xsd:gYear xsd:gYearMonth xsd:gMonth xsd:gMonthDay
xsd:gDay)
           ]
         ]
       ]
     ]
     [sh:property [
        sh:path (environs) ;
        sh:node [
          sh:property [
            sh:path ?propertyPath ;
            sh:datatype (xsd:date xsd:gYear xsd:gYearMonth xsd:gMonth xsd:gMonthDay
xsd:gDay)
           ]
         ]
       ]
     ]
     [sh:property [
        sh:path ?propertyPath ;
        sh:datatype (xsd:date xsd:gYear xsd:gYearMonth xsd:gMonth xsd:gMonthDay
xsd:gDay) ;
        sh:inversePath (has_participant_at_some_time)
       ]
     ]
```

```
      [sh:property [
         sh:path ?propertyPath ;
         sh:datatype (xsd:date xsd:gYear xsd:gYearMonth xsd:gMonth xsd:gMonthDay
xsd:gDay) ;
         sh:inversePath (history_of)
       ]
      ]
      [sh:property [
         sh:path ?propertyPath ;
         sh:datatype (xsd:date xsd:gYear xsd:gYearMonth xsd:gMonth xsd:gMonthDay
xsd:gDay) ;
         sh:inversePath (realizes)
       ]
      ]
      [sh:property [
         sh:path ?propertyPath ;
         sh:datatype (xsd:date xsd:gYear xsd:gYearMonth xsd:gMonth xsd:gMonthDay
xsd:gDay) ;
         sh:inversePath (occurs_in)
       ]
      ]
    ) .



  # Define the target class for the shape (i.e., the class of nodes that
  # we want to validate).
  ex:MyTargetClass rdfs:subClassOf owl:Thing .

  # Use the shape to validate instances of the target class.
  ex:MyTargetClass sh:property [
    sh:path ?node ;
    sh:node ex:DateValidationShape
  ] .


  # Define the target class for the shape (i.e., the class of nodes that
  # we want to validate).
  ex:MyTargetClass rdfs:subClassOf owl:Thing .

  # Use the shape to validate instances of the target class.
  ex:MyTargetClass sh:property [
    sh:path ?node ;
    sh:node ex:DateValidationShape
  ] .
```

SHACL TQV-III.

First, check whether a node in a given class has only one TQV.

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX sh: <http://www.w3.org/ns/shacl#>
PREFIX ex: <http://example.org/>

# Custom constraint component for checking if a node satisfies exactly one condition
ex:ExactlyOneConditionConstraint a sh:ConstraintComponent ;
  sh:parameter [
    sh:path ex:count ;
    sh:datatype xsd:integer ;
    sh:minExclusive 1
  ] ;
  sh:validate [
    a sh:NodeValidationFunction ;
    sh:function ex:exactlyOneCondition
  ] .

# Custom function for checking if a node satisfies exactly one condition
ex:exactlyOneCondition a sh:Function ;
  sh:message "Node satisfies more than one condition." ;
  # ...define the function logic here...

# A shape that checks if a node satisfies any of the conditions 1-9
# specified in the question and exactly one condition.
ex:DateValidationShape a sh:NodeShape ;
  sh:targetClass ?targetClass ;
  sh:or (
    [sh:property [
       sh:path (participates_in_at_some_time) ;
       sh:node [
         sh:property [
           sh:path ?propertyPath ;
           sh:datatype xsd:date
         ]
       ]
     ]
    ]
    [sh:property [
       sh:path (has_realization) ;
```

```
    sh:node [
      sh:property [
        sh:path ?propertyPath ;
        sh:datatype xsd:date
      ]
    ]
  ]
 ]
 [sh:property [
    sh:path (environs) ;
    sh:node [
      sh:property [
        sh:path ?propertyPath ;
        sh:datatype xsd:date
      ]
    ]
  ]
 ]
 [sh:property [
    sh:path ?propertyPath ;
    sh:datatype xsd:date;
    sh:inversePath (has_participant_at_some_time)
  ]
 ]
 [sh:property [
    sh:path ?propertyPath ;
    sh:datatype xsd:date;
    sh:inversePath (history_of)
  ]
 ]
 [sh:property [
    sh:path ?propertyPath ;
    sh:datatype xsd:date;
    sh:inversePath (realizes)
  ]
 ]
 [sh:property [
    sh:path ?propertyPath ;
    sh:datatype xsd:date;
    sh:inversePath (occurs_in)
  ]
 ]
[sh:property [
    sh:path (exist_at) ;
    sh:datatype xsd:date
  ]
```

```
    ]
    [sh:property [
       sh:path (occupies_temporal_region) ;
       sh:datatype xsd:date
     ]
    ]
  ) ;
  sh:property [
    sh:path ex:count ;
    sh:node ex:ExactlyOneConditionConstraint
  ] .

# Define the target class for the shape (i.e., the class of nodes that
# we want to validate).
ex:MyTargetClass rdfs:subClassOf owl:Thing .

# Use the shape to validate instances of the target class.
ex:MyTargetClass sh:property [
  sh:path ?node ;
  sh:node ex:DateValidationShape
] .
```

Second, add a SHACL rule to the ex:DateValidationShape that will add a new property ex:tqValue to the validated nodes:

```
ex:DateValidationShape a sh:NodeShape ;
  ...


  sh:rule [
    a sh:SPARQLRule ;
    sh:construct """
      PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
      PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
      PREFIX ex: <http://example.org/>

      CONSTRUCT {
        ?node ex:tqValue ?tqValue .
      }
      WHERE {
        ?node a ex:MyTargetClass .
        {
          ?node ex:has_history ?tqValue .
        } UNION {
```

```
        ?node ex:participates_in_at_some_time ?tqValue .
      } UNION {
        ?node ex:has_realization ?tqValue .
      } UNION {
        ?node ex:environs ?tqValue .
      } UNION {
        ?tqValue ex:has_participant_at_some_time ?node .
      } UNION {
        ?tqValue ex:history_of ?node .
      } UNION {
        ?tqValue ex:realizes ?node .
      } UNION {
        ?tqValue ex:occurs_in ?node .
      } UNION {
        ?node ex:exist_at ?tqValue .
      } UNION {
        ?node ex:occupies_temporal_region ?tqValue .
      }
      FILTER (
        datatype(?tqValue) = xsd:date
      )
    }
  """
] .
```

Third, After executing the SHACL validation with this shape, the validated nodes will have a new property ex:tqValue. Then using a SPARQL query to sort the nodes based on this new property:

```
PREFIX ex: <http://example.org/>
SELECT ?node ?tqValue
WHERE {
  ?node ex:tqValue ?tqValue .
}
ORDER BY ?tqValue
```

This SPARQL query will return the nodes and their ex:tqValue properties sorted in ascending order based on their TQ values.

SHACL TQV-IV

First, check the nodes in a given class (Person, supposedly) with the property has_history or history_of have birth date value.

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix ex: <http://example.com/> .

ex:BirthDateShape
   a sh:NodeShape ;
   sh:targetClass ex:Person ;
   sh:property [
      sh:path ex:has_history ;
      sh:node [
         sh:property [
            sh:path ex:birthDate ;
            sh:datatype xsd:date ;
            sh:minCount 1 ;
         ]
      ]
   ] ;
   sh:property [
      sh:path ex:history_of ;
      sh:node [
         sh:property [
            sh:path ex:birthDate ;
            sh:datatype xsd:date ;
            sh:minCount 1 ;
         ]
      ]
   ] .
```

Second, use another SHACL shape to list the nodes in order in terms of their birth date.

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix ex: <http://example.com/> .

ex:OrderedBirthDateShape
   a sh:NodeShape ;
   sh:targetClass ex:Person ;
   sh:property [
      sh:path ex:birthDate ;
      sh:datatype xsd:date ;
      sh:minCount 1 ;
```

```
        sh:order 1 ;
    ] .
```

SHACL TQV-V (similar as SHACL TQV-IV)

First, check the nodes in a given class (Person, supposedly) with the property has_history or history_of have death date value.

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix ex: <http://example.com/> .

ex:DeathDateShape
   a sh:NodeShape ;
   sh:targetClass ex:Person ;
   sh:property [
      sh:path ex:has_history ;
      sh:node [
         sh:property [
            sh:path ex:deathDate ;
            sh:datatype xsd:date ;
            sh:minCount 1 ;
         ]
      ]
   ] ;
   sh:property [
      sh:path ex:history_of ;
      sh:node [
         sh:property [
            sh:path ex:deathDate ;
            sh:datatype xsd:date ;
            sh:minCount 1 ;
         ]
      ]
   ] .
```

Second, use another SHACL shape to list the nodes, which are validated, in order in terms of their birth date.

```
   @prefix sh: <http://www.w3.org/ns/shacl#> .
   @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
   @prefix ex: <http://example.com/> .

   ex:OrderedDeathDateShape
      a sh:NodeShape ;
```

```
sh:targetClass ex:Person ;
sh:property [
    sh:path ex:deathDate ;
    sh:datatype xsd:date ;
    sh:minCount 1 ;
    sh:order 1 ;
] .
```