Test 1. Random test with property

Input:

I am an ontology engineer working on the development of an ontology for the rail domain, and working with the OWL and SHACL languages.
Given this data property expressed in OWL:

```
@prefix : <http://data.europa.eu/949/iss/> .
:contributingSystemicFactorNarrative rdf:type owl:DatatypeProperty ;
                                     rdfs:domain :ContributingSystemicFactor ;
                                     rdfs:range xsd:string ;
                                     dct:created "2024-07-08"^^xsd:date ;
                                     rdfs:isDefinedBy
<http://data.europa.eu/949/iss> ;
                                     rdfs:label "contributing systemic factor
narrative"@en .
```

Can you generate several SHACL shapes that are applicable to this data property?

Output:

Here you have several SHACL shapes that are aplicable to this datatype property :contributingSystemicFactorNarrative:

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix : <http://data.europa.eu/949/iss/> .

# Shape to ensure that the property exists and has the correct datatype
:ContributingSystemicFactorShape
    a sh:NodeShape ;
    sh:targetClass :ContributingSystemicFactor ;
    sh:property [
        sh:path :contributingSystemicFactorNarrative ;
        sh:datatype xsd:string ;
        sh:minCount 1 ; # Debe haber al menos un valor
        sh:maxCount 1 ; # No debe haber más de un valor
    ] .

# Shape to restrict the minimum and maximum length of its value
:ContributingSystemicFactorNarrativeLengthShape
    a sh:NodeShape ;
    sh:targetClass :ContributingSystemicFactor ;
    sh:property [
        sh:path :contributingSystemicFactorNarrative ;
        sh:datatype xsd:string ;
        sh:minLength 10 ; # Debe tener al menos 10 caracteres
        sh:maxLength 500 ; # No debe exceder 500 caracteres
    ] .

# Shape to ensure that the value follows a specific pattern (example: should start
with a capital letter)
:ContributingSystemicFactorNarrativePatternShape
    a sh:NodeShape ;
    sh:targetClass :ContributingSystemicFactor ;
    sh:property [
        sh:path :contributingSystemicFactorNarrative ;
        sh:datatype xsd:string ;
        sh:pattern "^[A-Z].*" ; # Debe empezar con una letra mayúscula
        sh:message "The contributing systemic factor narrative must start with a
capital letter."@en ;
    ] .
```

Shape to ensure that the property is defined in the ontology and documented

```
:ContributingSystemicFactorNarrativeMetadataShape
    a sh:NodeShape ;
    sh:targetClass :ContributingSystemicFactor ;
    sh:property [
        sh:path :contributingSystemicFactorNarrative ;
        sh:datatype xsd:string ;
        sh:description "A narrative description of the contributing systemic
factor."@en ;
    ] .
```

Each shape has a purpose:

ContributingSystemicFactorShape: Defines the property as mandatory and restricts is cardinality
ContributingSystemicFactorNarrativeLengthShape: Restricts the length of the property.
ContributingSystemicFactorNarrativePatternShape: Ensures that the text follows a specific format.
ContributingSystemicFactorNarrativeMetadataShape: Provides documentation for the property
If you need additional restirctions, tel me and I adjust. 😊

Comments on the test: Some of the constraints are additional to what has been defined as axioms in the ontology, such as the cardinalities, length, pattern (e.g., first letter is uppercase).

Test 2. Test to validate SKOS property trackDirection

Input:
I am an ontology engineer working on the development of an ontology for the rail domain, and working with the OWL and SHACL languages.
Given this data property expressed in OWL:

```
:trackDirection rdf:type owl:ObjectProperty ,
                owl:FunctionalProperty ;
                rdfs:domain :Track ;
                rdfs:range <http://www.w3.org/2004/02/skos/core#Concept> ;
                :inSkosConceptScheme <http://data.europa.eu/949/concepts/track-
running-directions/TrackRunningDirections> ;
                rdfs:comment """Normal running direction"""@en;
                rdfs:label "Normal running direction"@en ;
                <http://www.w3.org/2003/06/sw-vocab-status/ns#term_status> "stable"
.
```

Can you generate several SHACL shapes that are applicable to this data property?

Output:
Certainly! Here are several SHACL shapes that could be applicable to the `trackDirection` property in your rail ontology. These shapes will ensure that the data adheres to specific constraints, such as the correct usage of the property and its associated value types.

SHACL Shapes

1. **Shape for Track Class**
   This shape ensures that any instance of the `Track` class has a valid `trackDirection`.

```
:TrackShape
      a sh:NodeShape ;
      sh:targetClass :Track ;
      sh:property [
          sh:path :trackDirection ;
          sh:minCount 1 ;
          sh:nodeKind sh:IRI ;
```

```
            sh:datatype <http://www.w3.org/2004/02/skos/core#Concept> ;
        ] ;
        rdfs:comment "Shape to validate that Track instances have a valid
trackDirection."@en .
```

Comments on the test: sh:datatype check is incorrect. This is an object property.

2. **Shape for Valid Concepts**
   This shape ensures that the `trackDirection` values are instances of SKOS concepts.

```
  :TrackDirectionShape
        a sh:NodeShape ;
        sh:targetNode :trackDirection ;  # Assuming this is a specific instance
        sh:property [
            sh:path :trackDirection ;
            sh:nodeKind sh:IRI ;
            sh:in (<http://data.europa.eu/949/concepts/track-running-
directions/TrackRunningDirections>) ;
        ] ;
        rdfs:comment "Shape to ensure that trackDirection values are valid SKOS
concepts."@en .
  ```
```

Comments on the test: :trackDirection is not a specific instance so sh:targetNode is incorrect.
The concept scheme is not a set, incorrect use of sh:in.

3. **Shape for Functional Property Constraint**
   This shape ensures that the `trackDirection` property is functional, meaning that each instance of `Track` can only have one value for this property.

```
  :FunctionalTrackDirectionShape
        a sh:NodeShape ;
        sh:targetClass :Track ;
        sh:property [
            sh:path :trackDirection ;
            sh:maxCount 1 ;
        ] ;
        rdfs:comment "Shape to enforce that trackDirection is a functional
property."@en .
    ```
```

Comments on the test: OK

4. **Shape for Track Direction Labels**
   This shape checks that the labels for the `trackDirection` values are in English.

```
  :TrackDirectionLabelShape
        a sh:NodeShape ;
        sh:targetNode :trackDirection ;  # Assuming this is a specific instance
        sh:property [
            sh:path rdfs:label ;
            sh:datatype xsd:string ;
            sh:pattern ".*" ;  # Matches any string
            sh:language "en" ;
        ] ;
        rdfs:comment "Shape to ensure trackDirection labels are in English."@en .
```

Comments on the test: incorrect use of sh:targetNode as :trackDirection is not an instance.
Does not make sense to check labels of track direction.

Summary

These SHACL shapes can help validate instances of the `Track` class and their associated `trackDirection` property, ensuring compliance with the ontology's design and constraints. Adjust the target nodes or classes as needed based on your specific ontology structure.

Test 3. Previous test for trackDirection but with the ontology defining the range with an axiom restricting the value to be one in the concept scheme for track direction. (ontology with additional constraint)

I am an ontology engineer working on the development of an ontology for the rail domain, and working with the OWL and SHACL languages.
Given this data property expressed in OWL:

Input:

```
:trackDirection rdf:type owl:ObjectProperty ,
                owl:FunctionalProperty ;
                rdfs:domain :Track ;
                rdfs:range [ owl:intersectionOf ( skos:Concept
    [ rdf:type owl:Restriction ;
        owl:onProperty skos:inScheme ;
        owl:hasValue <http://data.europa.eu/949/concepts/track-running-
directions/TrackRunningDirections>
    ]
)
];
        rdfs:comment """Normal running direction"""@en;
                rdfs:label "Normal running direction"@en ;
                <http://www.w3.org/2003/06/sw-vocab-status/ns#term_status> "stable"
.
```

Can you generate several SHACL shapes that are applicable to this data property?

Output:
Given your OWL definition of the trackDirection data property, which includes a restriction for the trackDirection to be an object property with values that are SKOS concepts belonging to a specific concept scheme (era:track-running-directions), I will generate several SHACL shapes to enforce the constraints in your ontology. These shapes will ensure that:

The trackDirection is a valid SKOS concept.
The trackDirection belongs to the correct concept scheme (era:track-running-directions).
The trackDirection property is functional, meaning that each Track can have only one trackDirection.
The trackDirection property is applied to instances of Track.

1. Shape for Track Class and trackDirection Property
This shape ensures that instances of Track must have a valid trackDirection that is a SKOS concept belonging to the era:track-running-directions concept scheme.

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix ex: <http://example.org/ontology#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix era: <http://www.example.org/era#> .

ex:TrackShape
    a sh:NodeShape ;
    sh:targetClass ex:Track ;  # This shape applies to instances of Track

    # Ensures trackDirection is a SKOS Concept
    sh:property [
```

```
        sh:path ex:trackDirection ;
        sh:nodeKind sh:IRI ;  # Expecting IRI, which points to a SKOS concept
        sh:message "trackDirection must be a SKOS Concept." ;
    ] ;

    # Ensures that trackDirection is part of the specified concept scheme
    sh:property [
        sh:path ex:trackDirection ;
        sh:hasValue era:track-running-directions ;  # Track direction must be part
of the era:track-running-directions concept scheme
        sh:message "trackDirection must belong to the era:track-running-directions
concept scheme." ;
    ] ;

    # Ensures trackDirection is a valid SKOS concept (SKOS:Concept)
    sh:property [
        sh:path ex:trackDirection ;
        sh:node skos:Concept ;  # trackDirection must be a SKOS Concept
        sh:message "trackDirection must be a valid SKOS Concept." ;
    ] .
```

Comments on the test: The shape that ensures that track direction is a SKOS concept, only ensures that it is an IRI. The constraint that validates it is part of a concept scheme is incorrect, because the sh:hasValue should be a value of the concept, not the value that is the concept scheme.The third constraint is correct and is checking that the value is a skos:Concept. On the whole the shapes do not manage to validate that the value belongs to a sepcific concept scheme.

2. Shape for Valid trackDirection Values (SKOS Concept Scheme Validation)
This shape ensures that trackDirection values are valid SKOS concepts and that they belong to the era:track-running-directions concept scheme, as defined in your OWL ontology.

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix ex: <http://example.org/ontology#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix era: <http://www.example.org/era#> .

ex:TrackDirectionShape
    a sh:NodeShape ;
    sh:targetNode ex:trackDirection ;  # This shape applies to the trackDirection
property

    # Ensure trackDirection is part of the correct concept scheme (era:track-
running-directions)
    sh:property [
        sh:path ex:trackDirection ;
        sh:hasValue era:track-running-directions ;  # Must be part of the
era:track-running-directions concept scheme
        sh:message "trackDirection must belong to the era:track-running-directions
concept scheme." ;
    ] ;

    # Ensure trackDirection is a valid SKOS concept (SKOS:Concept)
    sh:property [
        sh:path ex:trackDirection ;
        sh:node skos:Concept ;  # trackDirection must be an IRI pointing to a SKOS
concept
        sh:message "trackDirection must be a valid SKOS Concept." ;
    ] .
```

Comments on the test: the first constraint is not correct, the value is a concept and not a concept scheme. The second constraint is correct and checks that the value is a skos:Concept. On the whole the two shapes do not manage to validate that the value belongs to a sepcific concept scheme.

3. Shape for Functional Property (Single trackDirection per Track)
This shape ensures that the trackDirection is functional, meaning each Track can have only one trackDirection.

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix ex: <http://example.org/ontology#> .

ex:FunctionalTrackDirectionShape
    a sh:NodeShape ;
    sh:targetClass ex:Track ;  # Applies to instances of Track

    # trackDirection is a functional property (each Track can have only one
trackDirection)
    sh:property [
        sh:path ex:trackDirection ;
        sh:maxCount 1 ;  # Only one trackDirection allowed per Track
        sh:message "trackDirection must be functional; each Track instance can have
only one trackDirection." ;
    ] .
```

Comments on the test: correct

4. Shape for trackDirection Domain Validation (Track Class)
This shape ensures that the trackDirection property is used only for instances of the Track class.

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix ex: <http://example.org/ontology#> .

ex:TrackDirectionDomainShape
    a sh:NodeShape ;
    sh:targetNode ex:trackDirection ;  # This shape applies to the trackDirection
property

    # Ensures trackDirection is associated with instances of Track
    sh:property [
        sh:path ex:trackDirection ;
        sh:node ex:Track ;  # Ensures the domain of trackDirection is Track
        sh:message "trackDirection must be used with instances of the Track class."
;
    ] .
```

Comments on the test: Correct

5. Shape for Concept Scheme Validation (trackDirection as SKOS Concept)
This shape ensures that the trackDirection is a SKOS concept and belongs to the correct concept scheme (era:track-running-directions), which is a combination of both SKOS and OWL restrictions as described in your OWL definition.

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix ex: <http://example.org/ontology#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix era: <http://www.example.org/era#> .

ex:TrackDirectionConceptSchemeShape
    a sh:NodeShape ;
    sh:targetNode ex:trackDirection ;

    # trackDirection must be a SKOS concept
    sh:property [
        sh:path ex:trackDirection ;
        sh:node skos:Concept ;  # Ensures trackDirection is a SKOS Concept
        sh:message "trackDirection must be a SKOS Concept." ;
    ] ;
```

```
  # Ensures trackDirection belongs to the correct concept scheme (era:track-running-directions)
  sh:property [
        sh:path ex:trackDirection ;
        sh:hasValue era:track-running-directions ;  # trackDirection must belong to
the era:track-running-directions concept scheme
        sh:message "trackDirection must belong to the era:track-running-directions
concept scheme." ;
    ] .
```

Comments on the test: Does not check correctly that the value belongs to a concept scheme. sh:hasValue points to the concept scheme, not to the value of the concept.

Summary of SHACL Shapes
TrackShape: Ensures that instances of Track have a valid trackDirection, which is a SKOS concept within the era:track-running-directions concept scheme.
TrackDirectionShape: Validates that trackDirection values are valid SKOS concepts from the era:track-running-directions concept scheme.
FunctionalTrackDirectionShape: Enforces the functional property constraint for trackDirection, allowing only one trackDirection per Track instance.
TrackDirectionDomainShape: Ensures that trackDirection is used only with instances of the Track class.
TrackDirectionConceptSchemeShape: Ensures that trackDirection is a SKOS concept that belongs to the era:track-running-directions concept scheme.
These SHACL shapes enforce the rules you have defined in your ontology for the trackDirection property, including its association with the correct SKOS concept scheme and its functional nature. Let me know if you need further modifications or additional shapes!

Test 3.1 We try to improve the output by indicating to use the skos:inScheme property.

Input:
Use the skos:inScheme to indicate that the concept belongs to a certain conceptScheme.

Output:
Ah, I see what you're saying! The restriction on skos:inScheme (which ensures that the trackDirection property values belong to a specific SKOS concept scheme) must be explicitly handled within the SHACL shapes.

To reflect this restriction, we will modify the SHACL shapes so that they ensure the trackDirection points to a SKOS concept that belongs to the era:track-running-directions concept scheme, as expressed in the owl:intersectionOf restriction in your OWL definition.

Updated SHACL Shapes to Include skos:inScheme Restriction
The SHACL shapes need to explicitly check that the trackDirection is a SKOS concept and that it has the appropriate skos:inScheme property pointing to the correct concept scheme.

1. Shape for Track Class and trackDirection Property
This shape ensures that instances of Track must have a trackDirection that is a SKOS concept, which is part of the era:track-running-directions concept scheme, as indicated by the skos:inScheme property.

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix ex: <http://example.org/ontology#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
```

```
@prefix era: <http://www.example.org/era#> .

ex:TrackShape
    a sh:NodeShape ;
    sh:targetClass ex:Track ;  # This shape applies to instances of Track

    # Ensure trackDirection is a SKOS Concept
    sh:property [
        sh:path ex:trackDirection ;
        sh:nodeKind sh:IRI ;  # Expecting IRI, which points to a SKOS concept
        sh:message "trackDirection must be a SKOS Concept." ;
    ] ;

    # Ensure that trackDirection belongs to the 'era:track-running-directions'
concept scheme
    sh:property [
        sh:path ex:trackDirection ;
        sh:node [
            sh:property [
                sh:path skos:inScheme ;
                sh:hasValue era:track-running-directions ;  # Must be part of the
era:track-running-directions concept scheme
                sh:message "trackDirection must belong to the era:track-running-
directions concept scheme." ;
            ]
        ] ;
        sh:message "trackDirection must be a valid SKOS Concept from the era:track-
running-directions scheme." ;
    ] .
```

Comments on the test: Correct but does not check that the value is a skos:Concept.

2. Shape for trackDirection Values as Valid SKOS Concepts (with skos:inScheme)
This shape ensures that trackDirection points to a valid SKOS concept, and explicitly checks that this SKOS concept is part of the era:track-running-directions concept scheme by validating the skos:inScheme property.

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix ex: <http://example.org/ontology#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix era: <http://www.example.org/era#> .

ex:TrackDirectionShape
    a sh:NodeShape ;
    sh:targetNode ex:trackDirection ;  # This shape applies to the trackDirection
property

    # Ensure trackDirection is a valid SKOS concept
    sh:property [
        sh:path ex:trackDirection ;
        sh:node skos:Concept ;  # trackDirection must be an IRI pointing to a SKOS
concept
        sh:message "trackDirection must be a valid SKOS Concept." ;
    ] ;

    # Ensure trackDirection belongs to the correct concept scheme using
skos:inScheme
    sh:property [
        sh:path ex:trackDirection ;
        sh:node [
            sh:property [
                sh:path skos:inScheme ;
                sh:hasValue era:track-running-directions ;  # trackDirection must
belong to the era:track-running-directions concept scheme
```

```
                sh:message "trackDirection must belong to the era:track-running-
directions concept scheme." ;
            ]
        ] ;
        sh:message "trackDirection must be a SKOS Concept from the era:track-
running-directions concept scheme." ;
    ] .
```

Comments on the test: Wrong use of the sh:targetNode property, it should refer to a specific instance, maybe it should have used sh:targetSubjectsOf?

3. Shape for Functional Property (Ensuring Single trackDirection per Track)
This shape enforces the functional property constraint, meaning that each Track can have only one trackDirection.

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix ex: <http://example.org/ontology#> .

ex:FunctionalTrackDirectionShape
    a sh:NodeShape ;
    sh:targetClass ex:Track ;  # Applies to instances of Track

    # trackDirection is a functional property (each Track can have only one
trackDirection)
    sh:property [
        sh:path ex:trackDirection ;
        sh:maxCount 1 ;  # Only one trackDirection allowed per Track
        sh:message "trackDirection must be functional; each Track instance can have
only one trackDirection." ;
    ] .
```

Comments on the test: correct.

4. Shape for trackDirection Domain Validation (Track Class)
This shape ensures that the trackDirection property is applied only to instances of the Track class.

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix ex: <http://example.org/ontology#> .

ex:TrackDirectionDomainShape
    a sh:NodeShape ;
    sh:targetNode ex:trackDirection ;  # This shape applies to the trackDirection
property

    # Ensures trackDirection is associated with instances of Track
    sh:property [
        sh:path ex:trackDirection ;
        sh:node ex:Track ;  # Ensures the domain of trackDirection is Track
        sh:message "trackDirection must be used with instances of the Track class."
;
    ] .
```

Comments on the test: again wrong use of sh:targetNode as it does not point to an instance. This does not provide additional validation if one chooses 1.

5. Shape for Concept Scheme Validation (trackDirection as SKOS Concept in the Correct Scheme)
This shape ensures that trackDirection is a SKOS concept and that it is part of the era:track-running-directions concept scheme using the skos:inScheme property.

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
```

```
@prefix ex: <http://example.org/ontology#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix era: <http://www.example.org/era#> .

ex:TrackDirectionConceptSchemeShape
    a sh:NodeShape ;
    sh:targetNode ex:trackDirection ;

    # trackDirection must be a SKOS concept
    sh:property [
        sh:path ex:trackDirection ;
        sh:node skos:Concept ;  # Ensures trackDirection is a SKOS Concept
        sh:message "trackDirection must be a SKOS Concept." ;
    ] ;

    # trackDirection must belong to the era:track-running-directions concept scheme
    sh:property [
        sh:path ex:trackDirection ;
        sh:node [
            sh:property [
                sh:path skos:inScheme ;
                sh:hasValue era:track-running-directions ;  # trackDirection must
belong to the era:track-running-directions concept scheme
                sh:message "trackDirection must belong to the era:track-running-
directions concept scheme." ;
            ]
        ] ;
        sh:message "trackDirection must be a SKOS Concept in the era:track-running-
directions concept scheme." ;
    ] .
```

Comments on the test: again wrong use of sh:targetNode. In general, would probably combine 1. with the functional property check 3., and the check that it must be a skos:Concept in 5.

Test 3.2 This was the output in a similar test, did not manage to get the same output than before but is equivalent:

Input: use the skos:inScheme property to validate that the :trackDirection property values belong to a specific SKOS concept scheme

Output:
To validate that the values of the `:trackDirection` property belong to a specific SKOS concept scheme using the `skos:inScheme` property, you can create a SHACL shape that targets the `:trackDirection` property and checks that its values are valid SKOS concepts associated with the specified concept scheme. Here's how to structure the SHACL shape:

```
### SHACL Shape for `:trackDirection` Using `skos:inScheme`

:TrackDirectionShape
    a sh:NodeShape ;
    sh:targetClass :Track ;  # Target the Track class
    sh:property [
        sh:path :trackDirection ;  # Validate the trackDirection property
        sh:minCount 1 ;  # At least one trackDirection must be present
        sh:nodeKind sh:IRI ;  # Ensure it's an IRI (i.e., a valid SKOS concept)
        sh:message "Each Track must have a valid trackDirection."@en ;
    ] ;
    sh:property [
        sh:path :trackDirection/skos:inScheme ;  # Validate the inScheme property
of trackDirection
        sh:hasValue <http://data.europa.eu/949/concepts/track-running-
directions/TrackRunningDirections> ;  # Specify the target concept scheme
```

```
        sh:message "trackDirection must belong to the specified concept scheme."@en
;
    ] ;
    rdfs:comment "Shape to ensure that trackDirection values belong to a specific
SKOS concept scheme."@en .
```

Additional SHACL Shapes

You can also keep the other shapes for additional validations as follows:

1. **Shape for Functional Property Constraint**
   This shape ensures that the `:trackDirection` property is functional.

```
  :FunctionalTrackDirectionShape
        a sh:NodeShape ;
        sh:targetClass :Track ;
        sh:property [
            sh:path :trackDirection ;
            sh:maxCount 1 ;  # Ensure that trackDirection is a functional property
            sh:message "trackDirection must be functional, allowing only one value
per Track."@en ;
        ] ;
        rdfs:comment "Shape to enforce that trackDirection is a functional
property."@en .
```

2. **Shape for Track Direction Labels**
   This shape checks that any labels for `:trackDirection` are in English.

```
  :TrackDirectionLabelShape
        a sh:NodeShape ;
        sh:targetNode :trackDirection ;  # Assuming this is a specific instance
        sh:property [
            sh:path rdfs:label ;
            sh:datatype xsd:string ;
            sh:language "en" ;  # Ensure labels are in English
            sh:message "trackDirection labels must be in English."@en ;
        ] ;
        rdfs:comment "Shape to ensure trackDirection labels are in English."@en .
    ```
```

### Summary
The `:TrackDirectionShape` now validates that the `:trackDirection` property values not only exist but also belong to the specified SKOS concept scheme through the use of `skos:inScheme`. This ensures compliance with your ontology requirements in the rail domain. Adjust any identifiers and URIs as needed for your specific ontology context!

Comments on the test: Correct but required several tries. Only with right input "use the skos:inScheme property to validate that the :trackDirection property values belong to a specific SKOS concept scheme". It again validates labels which does not make sense.

## Test 4. sparql-based constraints

**Input:**
Give me a shape that is sparql-based *(for the previous property trackDirection)*
```

Output:

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix ex: <http://example.org/ontology#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix era: <http://www.example.org/era#> .

ex:TrackDirectionSPARQLShape
    a sh:NodeShape ;
    sh:targetClass ex:Track ;  # Targeting instances of Track

    # SPARQL constraint: Validate trackDirection as a SKOS concept and its scheme
membership
    sh:property [
        sh:path ex:trackDirection ;
        sh:sparql [
            a sh:SPARQLConstraint ;
            sh:select """
                SELECT ?track ?trackDirection
                WHERE {
                    ?track ex:trackDirection ?trackDirection .
                    # Ensure that trackDirection is a SKOS concept
                    ?trackDirection a skos:Concept .
                    # Ensure that trackDirection is in the correct concept scheme
                    ?trackDirection skos:inScheme era:track-running-directions .
                }
            """ ;
            sh:message "trackDirection must be a valid SKOS Concept from the
era:track-running-directions concept scheme." ;
        ] ;
    ] ;

    # SPARQL constraint: Ensure trackDirection is functional (only one
trackDirection per Track)
    sh:property [
        sh:path ex:trackDirection ;
        sh:sparql [
            a sh:SPARQLConstraint ;
            sh:select """
                SELECT ?track
                WHERE {
                    ?track ex:trackDirection ?trackDirection .
                }
                GROUP BY ?track
                HAVING (COUNT(?trackDirection) > 1)
            """ ;
        sh:message "trackDirection must be functional; each Track instance can have only
one trackDirection." ;
    ] ;
  ] .
```

Comments on the test: correct and only lacking the check that ?track a era:Track, not really with focus node so that the domain is correct.

Test 4.1 Use of placeholder in message to improve output message when an error occurs.

Input:
the error message should show the specific track and value

Output:

To ensure that the error message in the SPARQL-based SHACL shape displays the specific track and its corresponding `trackDirection` value when validation fails, you can use the `sh:message` property with a format that includes the specific variables from the SPARQL query. Here's how you can set it up:

SPARQL-Based SHACL Shape with Specific Error Message

```
:TrackDirectionSPARQLShape
    a sh:NodeShape ;
    sh:targetClass :Track ;  # Target the Track class
    sh:property [
        sh:path :trackDirection ;
        sh:message "The track {track} has an invalid trackDirection value:
{trackDirection}."@en ;  # Placeholders for track and value
        sh:SPARQL """
            SELECT ?track ?trackDirection
            WHERE {
                ?track a :Track .
                ?track :trackDirection ?trackDirection .
                FILTER NOT EXISTS {
                    ?trackDirection a skos:Concept ;
                                    skos:inScheme
<http://data.europa.eu/949/concepts/track-running-
directions/TrackRunningDirections> .
                }
            }
    """ ;
    ] ;
    rdfs:comment "SPARQL-based shape to ensure that trackDirection values belong to
a specific SKOS concept scheme."@en .
```

Explanation

- **`sh:message`**: The message now contains `{track}` and `{trackDirection}` as placeholders that will be replaced by the actual track and its corresponding `trackDirection` value when the validation fails.
- **SPARQL Query**:
  - The `FILTER NOT EXISTS` clause is used to check if the `trackDirection` does not belong to the specified SKOS concept scheme. If this condition is met, the track and its direction will be reported in the error message.

Example of Expected Behavior
When a validation error occurs, the message might look like this:
```
"The track <trackURI> has an invalid trackDirection value: <trackDirectionURI>."
```

This setup provides specific feedback, allowing users to quickly identify the problematic `trackDirection` for each `Track` instance that does not conform to the expected constraints. Adjust any specific identifiers or namespaces as needed for your ontology!

Comments on the test:The placeholders are not correct, should be ?track and ?trackDirection.

Test 5. Business rules

Input:

I am an ontology engineer working on the development of an ontology for the rail domain, and working with the OWL and SHACL languages.
Given this data property expressed in OWL:

```
:trackSomeDocument rdf:type owl:ObjectProperty ,
                owl:FunctionalProperty ;
                rdfs:domain :Track ;
                rdfs:range :Document;
                rdfs:comment """Some document of track"""@en;
                rdfs:label "Some document"@en ;
                <http://www.w3.org/2003/06/sw-vocab-status/ns#term_status> "stable"
.

:trackHasSomeDocument rdf:type owl:DatatypeProperty ,
                owl:FunctionalProperty ;
                rdfs:domain :Track ;
                rdfs:range xsd:boolean;
                rdfs:comment """ has some document of track"""@en;
                rdfs:label " has some document"@en ;
                <http://www.w3.org/2003/06/sw-vocab-status/ns#term_status> "stable"
.
```

Can you generate a SHACL shape that validates that trackSomeDocument should have a value if trackHasSomeDocument is true

Output:
Certainly! To create a SHACL shape that ensures the `:trackSomeDocument` property has a value when the `:trackHasSomeDocument` property is true (i.e., has the value `true`), you can use a SPARQL-based shape. Below is how you can define such a SHACL shape:

SHACL Shape for Conditional Validation

```
:TrackDocumentValidationShape
    a sh:NodeShape ;
    sh:targetClass :Track ;  # Target the Track class
    sh:property [
        sh:path :trackSomeDocument ;  # Validate the trackSomeDocument property
        sh:message "If trackHasSomeDocument is true, trackSomeDocument must have a
value."@en ;
        sh:SPARQL """
            SELECT ?track
            WHERE {
                ?track a :Track .
                ?track :trackHasSomeDocument true .
                FILTER NOT EXISTS {
                    ?track :trackSomeDocument ?document .
                }
            }
        """ ;
    ] ;
    rdfs:comment "Shape to ensure that trackSomeDocument has a value if
trackHasSomeDocument is true."@en .
```

Explanation

- **`sh:NodeShape`**: This defines a shape for validating instances of the `:Track` class.
- **`sh:targetClass :Track`**: Specifies that this shape targets all instances of the `:Track` class.
- **`sh:property`**: Describes the properties to validate:

- **`sh:path :trackSomeDocument`**: Indicates that we are validating the `:trackSomeDocument` property.
- **`sh:message`**: Provides an error message that indicates the condition for validation.
- **`sh:SPARQL`**: Contains the SPARQL query that checks if `:trackHasSomeDocument` is true. If true, it checks that `:trackSomeDocument` does not have a value.

Expected Behavior
If an instance of `:Track` has `:trackHasSomeDocument` set to `true` but does not have a corresponding `:trackSomeDocument`, the validation will trigger the specified error message.

This SHACL shape effectively enforces the conditional requirement between the two properties in your rail domain ontology. Adjust any specific identifiers or namespaces as necessary for your implementation!

Comments on the test: It detected that it must be a sparql-based shape and it is correct.

Test 6. A general usiness rule

Input:
To create a SHACL shape that ensures the `:otherParameter` property has a value when the `:mainParameter` property is either "A" or "B", you can define a SPARQL-based shape as follows:

SHACL Shape for Conditional Validation

```
:OtherParameterValidationShape
    a sh:NodeShape ;
    sh:targetClass :Track ;  # Target the Track class
    sh:property [
        sh:path :otherParameter ;  # Validate the otherParameter property
        sh:message "If mainParameter is 'A' or 'B', otherParameter must have a
value."@en ;
        sh:SPARQL """
            SELECT ?track
            WHERE {
                ?track a :Track .
                ?track :mainParameter ?mainValue .
                FILTER (?mainValue = "A" || ?mainValue = "B")
                FILTER NOT EXISTS {
                    ?track :otherParameter ?otherValue .
                }
            }
        """ ;
    ] ;
    rdfs:comment "Shape to ensure that otherParameter has a value if mainParameter
is 'A' or 'B'."@en .
```

Explanation

- **`sh:NodeShape`**: This defines a shape for validating instances of the `:Track` class.
- **`sh:targetClass :Track`**: Specifies that this shape targets all instances of the `:Track` class.
- **`sh:property`**: Describes the properties to validate:
  - **`sh:path :otherParameter`**: Indicates that we are validating the `:otherParameter` property.
  - **`sh:message`**: Provides an error message that indicates the condition for validation.

- **`sh:SPARQL`**: Contains the SPARQL query that checks if `:mainParameter` has a value of "A" or "B". If so, it verifies that `:otherParameter` does not have a value.

Expected Behavior
If an instance of `:Track` has `:mainParameter` set to "A" or "B" but does not have a corresponding `:otherParameter`, the validation will trigger the specified error message.

This SHACL shape effectively enforces the conditional requirement between the two properties in your rail domain ontology. Adjust any specific identifiers or namespaces as necessary for your implementation!

Comments on the test: correct

Test 7. A more complex business rule

Input:
I am an ontology engineer working on the development of an ontology for the rail domain, and working with the OWL and SHACL languages.
Given this data property expressed in OWL:

```
:sidingId rdf:type owl:DatatypeProperty ,
                rdfs:domain :Siding ;
                rdfs:range xsd:string;
                rdfs:comment “””siding id””” parameter@en;
                rdfs:label “siding id”@en ;
                <http://www.w3.org/2003/06/sw-vocab-status/ns#term_status> "stable"
.

:hasSiding rdf:type owl:ObjectProperty ,
                rdfs:domain :OperationalPoint ;
                rdfs:range :Siding ;
                rdfs:comment “””has siding””” parameter@en;
                rdfs:label “has siding”@en ;
                <http://www.w3.org/2003/06/sw-vocab-status/ns#term_status> "stable"
.

:OperationalPoint rdf:type owl:Class ,
                rdfs:label “Operacional point“@en ;
                <http://www.w3.org/2003/06/sw-vocab-status/ns#term_status> "stable"
.

:Siding rdf:type owl:Class ,
                rdfs:label “siding@en ;
                <http://www.w3.org/2003/06/sw-vocab-status/ns#term_status> "stable"
.
```

Can you generate a SHACL shape that validates that a siding Id must be unique within an operational point .

Output:
To create a SHACL shape that validates that the `sidingId` must be unique within each instance of `:OperationalPoint`, you can use the `sh:UniqueProperty` constraint. Here's how you can define such a SHACL shape:

SHACL Shape for Unique `sidingId` within `OperationalPoint`

```
:UniqueSidingIdShape
    a sh:NodeShape ;
    sh:targetClass :OperationalPoint ;  # Target the OperationalPoint class
    sh:property [
        sh:path :hasSiding ;  # Validate the hasSiding property
        sh:message "sidingId must be unique within each OperationalPoint."@en ;
        sh:node [
            a sh:NodeShape ;
            sh:property [
                sh:path :sidingId ;  # Validate the sidingId property
                sh:uniqueProperty ;  # Enforce uniqueness
            ]
        ]
    ] ;
    rdfs:comment "Shape to ensure that sidingId must be unique within each OperationalPoint."@en .
```

Explanation

- **`sh:NodeShape`**: This defines a shape for validating instances of the `:OperationalPoint` class.
- **`sh:targetClass :OperationalPoint`**: Specifies that this shape targets all instances of the `:OperationalPoint` class.
- **`sh:property`**: Describes the properties to validate:
  - **`sh:path :hasSiding`**: Indicates that we are validating the `:hasSiding` property.
  - **`sh:message`**: Provides an error message that will be shown when the validation fails.
  - **`sh:node`**: Defines a nested node shape for the `:Siding` class, which validates the `sidingId`.
    - **`sh:property`**: Specifies the `sidingId` property to be validated for uniqueness using `sh:uniqueProperty`.

Expected Behavior
When validating an `:OperationalPoint`, the shape ensures that all associated `Siding` instances have unique `sidingId` values. If any duplicate `sidingId` values are found within the same `OperationalPoint`, the validation will trigger the specified error message.

This SHACL shape effectively enforces the uniqueness constraint for `sidingId` within the context of `OperationalPoint` in your rail domain ontology. Adjust any specific identifiers or namespaces as necessary for your implementation!

Comments on the test: incorrect, sh:uniqueProperty does not exist in the language

Test 7.1: when asking the same but with a sparql-based shape

Input:
Generate a sparql-based shape

Output:
To create a SPARQL-based SHACL shape that validates that the `sidingId` must be unique within each `OperationalPoint`, you can use a SPARQL query to check for duplicates. Here's how you can define such a shape:

SPARQL-Based SHACL Shape for Unique `sidingId`

```
:UniqueSidingIdShape
    a sh:NodeShape ;
```

```
    sh:targetClass :OperationalPoint ;  # Target the OperationalPoint class
    sh:property [
        sh:path :hasSiding ;  # Validate the hasSiding property
        sh:message "sidingId must be unique within each OperationalPoint."@en ;
        sh:SPARQL """
            SELECT ?op ?sidingId (COUNT(?siding) AS ?count)
            WHERE {
                ?op a :OperationalPoint .
                ?op :hasSiding ?siding .
                ?siding :sidingId ?sidingId .
            }
            GROUP BY ?op ?sidingId
            HAVING (COUNT(?siding) > 1)
        """ ;
    ] ;
    rdfs:comment "SPARQL-based shape to ensure that sidingId must be unique within
each OperationalPoint."@en .
```

Explanation

- **`sh:NodeShape`**: Defines a shape for validating instances of the `:OperationalPoint` class.
- **`sh:targetClass :OperationalPoint`**: Specifies that this shape targets all instances of the `:OperationalPoint` class.
- **`sh:property`**: Describes the properties to validate:
  - **`sh:path :hasSiding`**: Indicates that we are validating the `:hasSiding` property.
  - **`sh:message`**: Provides an error message to be displayed when validation fails.
  - **`sh:SPARQL`**: Contains the SPARQL query that checks for unique `sidingId` values within each `OperationalPoint`:
    - The query selects each operational point and counts the number of sidings associated with the same `sidingId`.
    - It groups results by `?op` (operational point) and `?sidingId`.
    - The `HAVING` clause ensures that if the count of sidings for a particular `sidingId` is greater than 1, it triggers the validation error.

Expected Behavior
When the validation runs, if any `OperationalPoint` has duplicate `sidingId` values among its associated sidings, the validation will trigger the specified error message.

This SPARQL-based shape effectively enforces the uniqueness constraint for `sidingId` within the context of `OperationalPoint` in your rail domain ontology. Adjust any specific identifiers or namespaces as necessary for your implementation!

Comments on the test: correct