Vielen Dank! Hier das Ergebnis des Tests mit der SHACL\_min.ttl und der Valid\_LCycle\_1.ttl:(Ich poste nur das, was "aus der Reihe tanzt" im Log):2025-03-12 14:46:17,107 - DEBUG - Constraint Violation in ClassConstraintComponent (<http://www.w3.org/ns/shacl#ClassConstraintComponent>): Severity: sh:Violation Source Shape: [ sh:class occp:ReviewApproval ] Focus Node: :instant\_completion\_of\_planning Value Node: :instant\_completion\_of\_planning Message: Value does not have class occp:ReviewApproval2025-03-12 14:46:17,109 - DEBUG - Constraint Violation in ClassConstraintComponent (<http://www.w3.org/ns/shacl#ClassConstraintComponent>): Severity: sh:Violation Source Shape: [ sh:class occp:CompletionOfConstruction ; sh:message Literal("Each 'occp:PhaseA\_Planning' must end before 'occp:CompletionOfConstruction' via 'occp:endsBefore'.", lang=en) ; sh:path occp:endsBefore ; sh:severity sh:Violation ] Focus Node: :phaseA Value Node: :phaseC Result Path: occp:endsBefore Message: Each 'occp:PhaseA\_Planning' must end before 'occp:CompletionOfConstruction' via 'occp:endsBefore'.2025-03-12 14:46:17,110 - DEBUG - Constraint Violation in ClassConstraintComponent (<http://www.w3.org/ns/shacl#ClassConstraintComponent>): Severity: sh:Violation Source Shape: [ sh:class occp:PhaseC\_Construction ; sh:message Literal("Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseC\_Construction' via 'occp:beginsBefore'.", lang=en) ; sh:path occp:beginsBefore ; sh:severity sh:Violation ] Focus Node: :phaseA Value Node: :phaseB Result Path: occp:beginsBefore Message: Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseC\_Construction' via 'occp:beginsBefore'.2025-03-12 14:46:17,121 - INFO - Validation Report Conforms: False Results (2): Constraint Violation in ClassConstraintComponent (<http://www.w3.org/ns/shacl#ClassConstraintComponent>): Severity: sh:Violation Source Shape: [ sh:class occp:CompletionOfConstruction ; sh:message Literal("Each 'occp:PhaseA\_Planning' must end before 'occp:CompletionOfConstruction' via 'occp:endsBefore'.", lang=en) ; sh:path occp:endsBefore ; sh:severity sh:Violation ] Focus Node: :phaseA Value Node: :phaseC Result Path: occp:endsBefore Message: Each 'occp:PhaseA\_Planning' must end before 'occp:CompletionOfConstruction' via 'occp:endsBefore'. Constraint Violation in ClassConstraintComponent (<http://www.w3.org/ns/shacl#ClassConstraintComponent>): Severity: sh:Violation Source Shape: [ sh:class occp:PhaseC\_Construction ; sh:message Literal("Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseC\_Construction' via 'occp:beginsBefore'.", lang=en) ; sh:path occp:beginsBefore ; sh:severity sh:Violation ] Focus Node: :phaseA Value Node: :phaseB Result Path: occp:beginsBefore Message: Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseC\_Construction' via 'occp:beginsBefore'.

Danke für das Log! Das ist ein großer Fortschritt – der ursprüngliche Fehler mit dem "not a well-formed SHACL PropertyShape" ist weg, was bedeutet, dass die minimale SHACL-Datei (OCCP\_SHACL\_min.ttl) syntaktisch korrekt ist und von PySHACL akzeptiert wird. Stattdessen haben wir jetzt Validierungsverletzungen, die auf logische Inkonsistenzen in der ABox (OCCP\_Valid\_LCycle\_1.ttl) hinweisen. Das ist ein gutes Zeichen, denn wir können uns jetzt auf die Daten konzentrieren und die SHACL-Regeln Schritt für Schritt wieder erweitern.

Analyse der Validierungsfehler

Ich gehe die Fehler im Log durch und erkläre, was sie bedeuten:

1. Fehler: :instant\_completion\_of\_planning hat nicht die Klasse occp:ReviewApproval

Constraint Violation in ClassConstraintComponent:

Severity: sh:Violation

Source Shape: [ sh:class occp:ReviewApproval ]

Focus Node: :instant\_completion\_of\_planning

Value Node: :instant\_completion\_of\_planning

Message: Value does not have class occp:ReviewApproval

* Quelle: osh:ShapeOfPhaseA mit:

turtle

sh:property [

sh:path occp:hasActualEnd ;

sh:or (

[ sh:class occp:ReviewApproval ]

[ sh:class occp:CompletionOfPlanning ]

) ;

sh:maxCount 1 ;

sh:message "Each 'occp:PhaseA\_Planning' must have exactly one actual end via 'occp:hasActualEnd' referencing 'occp:ReviewApproval' or 'occp:CompletionOfPlanning'."@en ;

sh:severity sh:Violation

] .

* Problem: In der ABox ist :phaseA mit occp:hasActualEnd an :instant\_completion\_of\_planning gebunden:

turtle

:phaseA a occp:PhaseA\_Planning ;

occp:hasActualEnd :instant\_completion\_of\_planning .

:instant\_completion\_of\_planning a occp:CompletionOfPlanning ;

occp:hasActualTime "2025-03-01"^^xsd:date .

Das ist an sich korrekt, denn :instant\_completion\_of\_planning hat die Klasse occp:CompletionOfPlanning, die im sh:or erlaubt ist. Der Fehler scheint hier falsch zu sein – möglicherweise ein Bug in der Berichtserstellung von PySHACL, da occp:CompletionOfPlanning akzeptiert werden sollte. Wir sollten das im Auge behalten, aber es könnte ein Anzeigefehler sein, da der Hauptbericht diesen Fehler nicht aufführt.

2. Fehler: :phaseA endet nicht vor occp:CompletionOfConstruction

Constraint Violation in ClassConstraintComponent:

Severity: sh:Violation

Source Shape: [ sh:class occp:CompletionOfConstruction ; sh:message Literal("Each 'occp:PhaseA\_Planning' must end before 'occp:CompletionOfConstruction' via 'occp:endsBefore'.", lang=en) ; sh:path occp:endsBefore ; sh:severity sh:Violation ]

Focus Node: :phaseA

Value Node: :phaseC

Result Path: occp:endsBefore

Message: Each 'occp:PhaseA\_Planning' must end before 'occp:CompletionOfConstruction' via 'occp:endsBefore'.

* Quelle: osh:ShapeOfPhaseA mit:

turtle

sh:property [

sh:path occp:endsBefore ;

sh:class occp:CompletionOfConstruction ;

sh:message "Each 'occp:PhaseA\_Planning' must end before 'occp:CompletionOfConstruction' via 'occp:endsBefore'."@en ;

sh:severity sh:Violation

] .

* Problem: In der ABox steht:

turtle

:phaseA a occp:PhaseA\_Planning ;

occp:endsBefore :phaseC .

:phaseC a occp:PhaseC\_Construction ;

occp:hasActualEnd :instant\_completion\_of\_construction .

:instant\_completion\_of\_construction a occp:CompletionOfConstruction ;

occp:hasActualTime "2025-06-01"^^xsd:date .

Die Regel verlangt, dass :phaseA via occp:endsBefore direkt mit einer Instanz von occp:CompletionOfConstruction (z. B. :instant\_completion\_of\_construction) verknüpft ist. Stattdessen ist :phaseA mit :phaseC (Typ occp:PhaseC\_Construction) verknüpft. Das ist semantisch nicht dasselbe, da occp:endsBefore eine Beziehung zu einem konkreten Ereignis (CompletionOfConstruction) erwartet, nicht zu einer Phase.

3. Fehler: :phaseA beginnt nicht vor occp:PhaseC\_Construction

Constraint Violation in ClassConstraintComponent:

Severity: sh:Violation

Source Shape: [ sh:class occp:PhaseC\_Construction ; sh:message Literal("Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseC\_Construction' via 'occp:beginsBefore'.", lang=en) ; sh:path occp:beginsBefore ; sh:severity sh:Violation ]

Focus Node: :phaseA

Value Node: :phaseB

Result Path: occp:beginsBefore

Message: Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseC\_Construction' via 'occp:beginsBefore'.

* Quelle: osh:ShapeOfPhaseA mit:

turtle

sh:property [

sh:path occp:beginsBefore ;

sh:class occp:PhaseC\_Construction ;

sh:message "Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseC\_Construction' via 'occp:beginsBefore'."@en ;

sh:severity sh:Violation

] .

* Problem: In der ABox steht:

turtle

:phaseA a occp:PhaseA\_Planning ;

occp:beginsBefore :phaseB .

:phaseB a occp:PhaseB\_Review ;

occp:beginsBefore :phaseC .

:phaseC a occp:PhaseC\_Construction .

Die Regel verlangt, dass :phaseA via occp:beginsBefore direkt mit :phaseC (Typ occp:PhaseC\_Construction) verknüpft ist. Stattdessen ist :phaseA nur mit :phaseB verknüpft. Obwohl :phaseB vor :phaseC liegt (transitive Beziehung), prüft SHACL nur direkte Verknüpfungen, es sei denn, wir verwenden Inferenz oder SPARQL, um transitive Beziehungen zu berücksichtigen.

Zusammenfassung der Probleme

1. Anzeigefehler (?): Der erste Fehler (:instant\_completion\_of\_planning) scheint ein falscher Bericht zu sein, da occp:CompletionOfPlanning erlaubt ist. Das ignorieren wir vorerst.
2. Semantischer Fehler: Die Regeln für occp:endsBefore und occp:beginsBefore erwarten direkte Verknüpfungen zu occp:CompletionOfConstruction bzw. occp:PhaseC\_Construction, aber die ABox verknüpft :phaseA mit :phaseC bzw. :phaseB. SHACL prüft nur die expliziten Triple, nicht die logische Implikation (z. B. durch Zeitstempel oder transitive Beziehungen).

Lösungsvorschläge

1. ABox anpassen

Wir könnten die ABox korrigieren, um die direkten Beziehungen herzustellen, die die SHACL-Regeln erwarten:

turtle

:phaseA a occp:PhaseA\_Planning ;

occp:hasActualBeginning :instant\_beginning\_of\_planning ;

occp:hasActualEnd :instant\_completion\_of\_planning ;

occp:beginsBefore :phaseB ;

occp:beginsBefore :phaseC ; # Hinzugefügt

occp:endsBefore :instant\_completion\_of\_construction ; # Geändert von :phaseC

occp:startsPhase :instant\_beginning\_of\_planning ;

occp:endsPhase :instant\_review\_approval .

* Änderung 1: occp:beginsBefore :phaseC hinzufügen, damit :phaseA direkt vor :phaseC\_Construction liegt.
* Änderung 2: occp:endsBefore :phaseC durch occp:endsBefore :instant\_completion\_of\_construction ersetzen, da die Regel eine Instanz von occp:CompletionOfConstruction erwartet.

2. SHACL-Regeln anpassen

Alternativ könnten wir die SHACL-Regeln flexibler gestalten, um transitive Beziehungen oder Phasen als gültige Ziele zu akzeptieren:

* Für occp:endsBefore:

turtle

sh:property [

sh:path occp:endsBefore ;

sh:or (

[ sh:class occp:CompletionOfConstruction ]

[ sh:class occp:PhaseC\_Construction ]

) ;

sh:message "Each 'occp:PhaseA\_Planning' must end before 'occp:CompletionOfConstruction' or 'occp:PhaseC\_Construction' via 'occp:endsBefore'."@en ;

sh:severity sh:Violation

] ;

* Für occp:beginsBefore: Wir könnten eine SPARQL-Regel hinzufügen, um transitive Beziehungen zu prüfen, aber das machen wir später, wenn nötig.

3. Nächster Test

Ich empfehle, erst die ABox anzupassen, da das einfacher ist und die ursprüngliche Intention der Regeln erhält. Hier die minimale Änderung:

turtle

:phaseA a occp:PhaseA\_Planning ;

occp:hasActualBeginning :instant\_beginning\_of\_planning ;

occp:hasActualEnd :instant\_completion\_of\_planning ;

occp:beginsBefore :phaseB ;

occp:beginsBefore :phaseC ; # Hinzugefügt

occp:endsBefore :instant\_completion\_of\_construction ; # Geändert

occp:startsPhase :instant\_beginning\_of\_planning ;

occp:endsPhase :instant\_review\_approval .

Führe das Script nochmal mit dieser geänderten ABox und OCCP\_SHACL\_min.ttl aus.

Plan

1. Testen: Passe :phaseA in der ABox wie oben an und teste erneut. Teile mir das neue Log mit!
2. Erweitern: Wenn das funktioniert (Conforms: True), können wir die SHACL-Shapes schrittweise wieder erweitern (z. B. Phase B, Zyklen) und prüfen, ob der ursprüngliche Fehler zurückkommt.
3. Debuggen: Falls der Fehler bleibt, schauen wir uns die Zeitstempel-Logik oder PySHACL-spezifische Probleme an.

Was meinst du? Soll ich die ABox für dich anpassen und dir die neue Version schicken, oder machst du das selbst und testest? Ich freue mich auf dein Feedback!

Hier der neue Report mit der geänderten PhaseA in der ABox:2025-03-12 14:55:32,142 - INFO - Validation Report Conforms: False Results (3): Validation Result in ClassConstraintComponent (<http://www.w3.org/ns/shacl#ClassConstraintComponent>): Severity: sh:Warning Source Shape: [ sh:class occp:PhaseC\_Construction ; sh:message Literal("Each 'occp:PhaseA\_Planning' must end before 'occp:PhaseC\_Construction' begins via 'occp:endsBefore'.", lang=en) ; sh:path occp:endsBefore ; sh:severity sh:Warning ] Focus Node: :phaseA Value Node: :instant\_completion\_of\_construction Result Path: occp:endsBefore Message: Each 'occp:PhaseA\_Planning' must end before 'occp:PhaseC\_Construction' begins via 'occp:endsBefore'. Constraint Violation in ClassConstraintComponent (<http://www.w3.org/ns/shacl#ClassConstraintComponent>): Severity: sh:Violation Source Shape: [ sh:class occp:PhaseB\_Review ; sh:message Literal("Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseB\_Review' via 'occp:beginsBefore'.", lang=en) ; sh:path occp:beginsBefore ; sh:severity sh:Violation ] Focus Node: :phaseA Value Node: :phaseC Result Path: occp:beginsBefore Message: Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseB\_Review' via 'occp:beginsBefore'. Constraint Violation in ClassConstraintComponent (<http://www.w3.org/ns/shacl#ClassConstraintComponent>): Severity: sh:Violation Source Shape: [ sh:class occp:PhaseC\_Construction ; sh:message Literal("Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseC\_Construction' via 'occp:beginsBefore'.", lang=en) ; sh:path occp:beginsBefore ; sh:severity sh:Violation ] Focus Node: :phaseA Value Node: :phaseB Result Path: occp:beginsBefore Message: Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseC\_Construction' via 'occp:beginsBefore'.

Danke für den neuen Report! Wir sind auf dem richtigen Weg – der ursprüngliche Syntaxfehler ist weg, und wir haben jetzt nur noch logische Validierungsfehler, die wir Schritt für Schritt lösen können. Lass uns die neuen Fehler analysieren und beheben.

Analyse des neuen Validation Reports

Der Report zeigt drei Validierungsfehler für :phaseA. Schauen wir sie uns einzeln an:

1. Fehler: :phaseA endet nicht vor occp:PhaseC\_Construction (Warning)

Validation Result in ClassConstraintComponent:

Severity: sh:Warning

Source Shape: [ sh:class occp:PhaseC\_Construction ; sh:message Literal("Each 'occp:PhaseA\_Planning' must end before 'occp:PhaseC\_Construction' begins via 'occp:endsBefore'.", lang=en) ; sh:path occp:endsBefore ; sh:severity sh:Warning ]

Focus Node: :phaseA

Value Node: :instant\_completion\_of\_construction

Result Path: occp:endsBefore

Message: Each 'occp:PhaseA\_Planning' must end before 'occp:PhaseC\_Construction' begins via 'occp:endsBefore'.

* Quelle: osh:ShapeOfPhaseA mit:

turtle

sh:property [

sh:path occp:endsBefore ;

sh:class occp:PhaseC\_Construction ;

sh:message "Each 'occp:PhaseA\_Planning' must end before 'occp:PhaseC\_Construction' begins via 'occp:endsBefore'."@en ;

sh:severity sh:Warning

] .

* ABox:

turtle

:phaseA a occp:PhaseA\_Planning ;

occp:endsBefore :instant\_completion\_of\_construction .

:instant\_completion\_of\_construction a occp:CompletionOfConstruction ;

occp:hasActualTime "2025-06-01"^^xsd:date .

:phaseC a occp:PhaseC\_Construction ;

occp:hasActualBeginning :instant\_construction\_start ;

occp:hasActualEnd :instant\_completion\_of\_construction .

:instant\_construction\_start a occp:ConstructionStart ;

occp:hasActualTime "2025-03-15"^^xsd:date .

* Problem: Die Regel erwartet, dass :phaseA via occp:endsBefore mit einer Instanz von occp:PhaseC\_Construction (z. B. :phaseC) verknüpft ist. Stattdessen ist :phaseA mit :instant\_completion\_of\_construction (Typ occp:CompletionOfConstruction) verknüpft. Das ist ein Typkonflikt, da sh:class occp:PhaseC\_Construction nicht erfüllt wird. Dies ist nur ein Warning, aber es zeigt, dass die Regel nicht das prüft, was wir vermutlich intendieren.

2. Fehler: :phaseA beginnt nicht vor occp:PhaseB\_Review

Constraint Violation in ClassConstraintComponent:

Severity: sh:Violation

Source Shape: [ sh:class occp:PhaseB\_Review ; sh:message Literal("Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseB\_Review' via 'occp:beginsBefore'.", lang=en) ; sh:path occp:beginsBefore ; sh:severity sh:Violation ]

Focus Node: :phaseA

Value Node: :phaseC

Result Path: occp:beginsBefore

Message: Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseB\_Review' via 'occp:beginsBefore'.

* Quelle: osh:ShapeOfPhaseA mit:

turtle

sh:property [

sh:path occp:beginsBefore ;

sh:class occp:PhaseB\_Review ;

sh:message "Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseB\_Review' via 'occp:beginsBefore'."@en ;

sh:severity sh:Violation

] .

* ABox:

turtle

:phaseA a occp:PhaseA\_Planning ;

occp:beginsBefore :phaseB ;

occp:beginsBefore :phaseC .

:phaseB a occp:PhaseB\_Review .

:phaseC a occp:PhaseC\_Construction .

* Problem: Die Regel verlangt, dass :phaseA via occp:beginsBefore mit einer Instanz von occp:PhaseB\_Review (z. B. :phaseB) verknüpft ist. Das ist erfüllt (occp:beginsBefore :phaseB), aber der Fehler meldet :phaseC als Value Node. Das deutet darauf hin, dass SHACL alle occp:beginsBefore-Verknüpfungen prüft und erwartet, dass jede davon den Typ occp:PhaseB\_Review hat. Die zusätzliche Verknüpfung :phaseA occp:beginsBefore :phaseC verletzt diese Regel, da :phaseC nicht vom Typ occp:PhaseB\_Review ist.

3. Fehler: :phaseA beginnt nicht vor occp:PhaseC\_Construction

Constraint Violation in ClassConstraintComponent:

Severity: sh:Violation

Source Shape: [ sh:class occp:PhaseC\_Construction ; sh:message Literal("Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseC\_Construction' via 'occp:beginsBefore'.", lang=en) ; sh:path occp:beginsBefore ; sh:severity sh:Violation ]

Focus Node: :phaseA

Value Node: :phaseB

Result Path: occp:beginsBefore

Message: Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseC\_Construction' via 'occp:beginsBefore'.

* Quelle: osh:ShapeOfPhaseA mit:

turtle

sh:property [

sh:path occp:beginsBefore ;

sh:class occp:PhaseC\_Construction ;

sh:message "Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseC\_Construction' via 'occp:beginsBefore'."@en ;

sh:severity sh:Violation

] .

* ABox:

turtle

:phaseA a occp:PhaseA\_Planning ;

occp:beginsBefore :phaseB ;

occp:beginsBefore :phaseC .

:phaseB a occp:PhaseB\_Review .

:phaseC a occp:PhaseC\_Construction .

* Problem: Ähnlich wie beim zweiten Fehler: Die Regel erwartet, dass jede occp:beginsBefore-Verknüpfung von :phaseA auf eine Instanz von occp:PhaseC\_Construction zeigt. Die Verknüpfung :phaseA occp:beginsBefore :phaseB verletzt dies, da :phaseB nicht vom Typ occp:PhaseC\_Construction ist.

Grundproblem

Die SHACL-Regeln in osh:ShapeOfPhaseA sind zu strikt formuliert:

* Jede Regel mit sh:path occp:beginsBefore oder sh:path occp:endsBefore prüft alle Werte dieser Properties und erwartet, dass sie den angegebenen Typ (sh:class) haben.
* In der ABox hat :phaseA mehrere occp:beginsBefore-Verknüpfungen (:phaseB und :phaseC), aber die Regeln erlauben nur eine spezifische Klasse pro Regel, was zu Konflikten führt.

Lösungsvorschläge

1. ABox anpassen

Wir könnten die ABox so ändern, dass :phaseA nur die erwarteten direkten Verknüpfungen hat, die den Regeln entsprechen:

turtle

:phaseA a occp:PhaseA\_Planning ;

occp:hasActualBeginning :instant\_beginning\_of\_planning ;

occp:hasActualEnd :instant\_completion\_of\_planning ;

occp:beginsBefore :phaseB ; # Beibehalten für PhaseB\_Review

occp:beginsBefore :phaseC ; # Beibehalten für PhaseC\_Construction

occp:endsBefore :instant\_completion\_of\_construction ; # Beibehalten für CompletionOfConstruction

occp:startsPhase :instant\_beginning\_of\_planning ;

occp:endsPhase :instant\_review\_approval .

Das Problem bleibt jedoch, dass SHACL mehrere occp:beginsBefore-Werte nicht korrekt handhabt, da jede Regel nur einen Typ erwartet.

2. SHACL-Regeln anpassen (Empfohlen)

Die SHACL-Regeln sollten flexibler sein, um mehrere gültige Ziele für occp:beginsBefore und occp:endsBefore zu erlauben. Hier eine überarbeitete Version von osh:ShapeOfPhaseA:

turtle

osh:ShapeOfPhaseA a sh:NodeShape ;

sh:targetClass occp:PhaseA\_Planning ;

sh:property [

sh:path occp:endsBefore ;

sh:or (

[ sh:class occp:PhaseC\_Construction ]

[ sh:class occp:CompletionOfConstruction ]

) ;

sh:minCount 1 ; # Mindestens eine endsBefore-Beziehung

sh:message "Each 'occp:PhaseA\_Planning' must end before 'occp:PhaseC\_Construction' or 'occp:CompletionOfConstruction' via 'occp:endsBefore'."@en ;

sh:severity sh:Warning

] ;

sh:property [

sh:path occp:beginsBefore ;

sh:or (

[ sh:class occp:PhaseB\_Review ]

[ sh:class occp:PhaseC\_Construction ]

) ;

sh:minCount 1 ; # Mindestens eine beginsBefore-Beziehung

sh:message "Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseB\_Review' or 'occp:PhaseC\_Construction' via 'occp:beginsBefore'."@en ;

sh:severity sh:Violation

] ;

sh:property [

sh:path occp:hasActualBeginning ;

sh:class occp:BeginningOfPlanning ;

sh:maxCount 1 ;

sh:message "Each 'occp:PhaseA\_Planning' must have exactly one actual beginning via 'occp:hasActualBeginning' referencing 'occp:BeginningOfPlanning'."@en ;

sh:severity sh:Violation

] ;

sh:property [

sh:path occp:hasActualEnd ;

sh:or (

[ sh:class occp:ReviewApproval ]

[ sh:class occp:CompletionOfPlanning ]

) ;

 Andere Änderungen könnten sein:

sh:maxCount 1 ;

sh:message "Each 'occp:PhaseA\_Planning' must have exactly one actual end via 'occp:hasActualEnd' referencing 'occp:ReviewApproval' or 'occp:CompletionOfPlanning'."@en ;

sh:severity sh:Violation

] .

* Änderungen:
  + occp:endsBefore: Akzeptiert nun entweder occp:PhaseC\_Construction oder occp:CompletionOfConstruction.
  + occp:beginsBefore: Akzeptiert nun entweder occp:PhaseB\_Review oder occp:PhaseC\_Construction.
  + sh:minCount 1: Stellt sicher, dass mindestens eine Beziehung existiert, ohne jeden Wert einzeln zu prüfen.

3. Zeitstempel-basierte Validierung (zukünftig)

Die aktuellen Regeln prüfen nur Typen, nicht die tatsächliche zeitliche Reihenfolge (z. B. ob "2025-03-01" wirklich vor "2025-06-01" liegt). Das könnten wir später mit SPARQL-Regeln ergänzen, aber für jetzt reicht die Typprüfung.

Nächster Schritt

Ich empfehle, die SHACL-Regeln wie oben anzupassen, da das die ABox unverändert lässt und die Regeln realistischer macht. Hier die Schritte:

1. SHACL aktualisieren: Ersetze osh:ShapeOfPhaseA in OCCP\_SHACL\_min.ttl mit der neuen Version.
2. Testen: Führe das Script erneut mit der aktuellen ABox aus.
3. Feedback: Teile mir das neue Log mit – ich hoffe, wir sehen dann Conforms: True!

Frage

Soll ich dir die angepasste OCCP\_SHACL\_min.ttl direkt schicken, oder machst du die Änderung selbst? Wenn du willst, kann ich auch das Script ausführen, falls du mir sagst, wo ich die Dateien speichern soll. Lass mich wissen, wie du vorgehen möchtest!

Vielen Dank für Deine Vorschläge! Ich glaube, die ursprüngliche Logik ist an sich in Ordnung. Ich möchte nicht die Logik ändern und die Einführung von minCounts für die Beziehung halte ich auch nicht für zielführend (oder ich verstehe Deinen Ansatz nicht... Warum sollte uns das helfen?). Mein Gedanke ist, dass das Problem in der Nicht-Betrachtung der Zeitstempel liegt, die zu Fehlern oder Unklarheiten führen und ggf. in der Art die ABox aufzubauen. Beispiel: Eine Phase muss vor einer anderen Enden (Phase B muss vor Phase C sein oder zumindest vor der Transition CompletionOfPlanning). Wenn wir ein Individual in der ABox anlegen für die PhaseA, dann steht da erstmal nur drin, wann die Phase beginnt (nämlich zum Zeitpunkt BeginningOfPlanning - für diesen Instant wird ein Zeitstempel hinterlegt, der auch den Anfang von Phase A definiert. Zusätzlich müsste bei der Erstellung des PhaseA-Individuals (wegen der GeneralDatePropertyShape) mind. 1 Wert für hasEnd definiert werden - der mit einem Instant verknüpft werden muss. In diesem Fall wäre das die Transition CompletionOfPlanning, für die man dann wiederum einen Zeitstempel (am ehesten über hasEstimatedTime) hinterlegt wird. Eine Bezugsetzung der PhaseA zu anderen Phasen würde ich nicht vornehmen in der ABox, weil sich diese Bezüge ja aus der TBox und den SHACL-Rules ableiten. Die Validierung muss also nur die Verknüpfungen (also der Instant, der mit dem Anfang/Ende einer Phase verknüpft wird, muss übereinstimmen mit den jeweils zulässigen Instants für die jeweilige Phase) und die Zeitstempel überprüfen, wobei bei den Zeitstempeln immer über die zulässigen Beziehungen ausgelesen werden müsste, welche Zeitstempel miteinander zu vergleichen sind und welche die (gem. Ontologie) korrekte bzw. zulässige zeitliche Abfolge ist, um abschließend zu prüfen, ob die Zeitstempel dieser Abfolge entsprechen oder nicht. Die Abfrage der Zeitstempel haben wir noch gar nicht implementiert... Das sollten wir vielleicht tun. Was meinst Du dazu? Nun stellt sich die Frage, was passiert, wenn weitere Phasen dazukommen. Also: Wenn PhaseC eingetragen wird

Alles klar, ich verstehe deinen Ansatz jetzt besser, und du hast recht – die ursprüngliche Logik der Ontologie und SHACL-Regeln ist grundsätzlich sinnvoll, und das Problem liegt tatsächlich in der Nicht-Betrachtung der Zeitstempel sowie möglicherweise in der Art, wie die ABox aufgebaut ist. Lass uns das mal auseinandernehmen und deinen Vorschlag, die Zeitstempelprüfung einzubauen, konkret angehen.

Analyse deiner Überlegungen

1. Zeitstempel als Kern der Validierung:
   * Du möchtest, dass die Validierung nicht nur die Typen und Verknüpfungen prüft (z. B. occp:beginsBefore, occp:endsBefore), sondern auch die tatsächlichen Zeitstempel (via occp:hasActualTime oder occp:hasEstimatedTime) gegen die in der TBox/SHACL definierte zeitliche Reihenfolge validiert. Das ist absolut logisch, denn die chronologische Korrektheit ist das Herzstück deiner OCCP-Ontologie.
   * Aktuell prüfen die SHACL-Regeln nur, ob die Verknüpfungen zu den richtigen Klassen zeigen (z. B. occp:PhaseA\_Planning muss vor occp:PhaseC\_Construction beginnen), aber nicht, ob die Zeitstempel diese Reihenfolge tatsächlich widerspiegeln.
2. ABox-Aufbau:
   * Dein Punkt ist, dass die ABox primär die Phasen und ihre Start-/End-Instant mit Zeitstempeln definieren sollte (z. B. PhaseA beginnt mit BeginningOfPlanning und endet mit CompletionOfPlanning), ohne explizite Beziehungen wie occp:beginsBefore oder occp:endsBefore zu anderen Phasen direkt anzugeben. Diese Beziehungen sollen sich aus der TBox (z. B. transitive Eigenschaften von occp:before) und den SHACL-Regeln ableiten.
   * Beispiel: Wenn :phaseA mit :instant\_beginning\_of\_planning (2025-01-01) beginnt und mit :instant\_completion\_of\_planning (2025-03-01) endet, und :phaseC mit :instant\_construction\_start (2025-03-15) beginnt, sollte die Validierung prüfen, ob 2025-03-01 < 2025-03-15 ist, anstatt eine direkte occp:endsBefore-Verknüpfung zu erzwingen.
3. Warum sh:minCount nicht hilft:
   * Du hast recht, dass das Einführen von sh:minCount die Logik nicht verbessert. Mein Vorschlag war darauf abzielen, mehrere mögliche Ziele (z. B. PhaseB\_Review oder PhaseC\_Construction) zu erlauben, aber das löst nicht das eigentliche Problem der Zeitstempelprüfung. Es war eher ein Workaround für die Typkonflikte, die wir im letzten Report gesehen haben, und keine echte Lösung für die chronologische Validierung.
4. Fehlende Zeitstempelprüfung:
   * Genau wie du sagst: Wir haben die Zeitstempel noch nicht in die SHACL-Validierung integriert. Die aktuellen Regeln sind rein typbasiert und berücksichtigen nicht die Werte von occp:hasActualTime oder occp:hasEstimatedTime. Das erklärt auch, warum wir Fehler wie ":phaseA beginnt nicht vor PhaseB\_Review" bekommen – SHACL prüft nur die Typen der Verknüpfungen, nicht die zeitliche Logik.

Vorschlag: Zeitstempel-basierte Validierung implementieren

Ich stimme dir zu, dass wir die Zeitstempelprüfung einführen sollten. Das können wir mit SPARQL-basierten SHACL-Regeln machen, die die Zeitstempel vergleichen und die zulässigen Beziehungen aus der Ontologie berücksichtigen. Hier ist ein Plan, wie wir das angehen können:

1. Zeitstempel in die SHACL-Regeln integrieren

Wir ersetzen oder ergänzen die typbasierten Regeln (z. B. sh:class) durch SPARQL-Abfragen, die die Zeitstempel prüfen. Beispiel für osh:ShapeOfPhaseA:

turtle

osh:ShapeOfPhaseA a sh:NodeShape ;

sh:targetClass occp:PhaseA\_Planning ;

sh:property [

sh:path occp:hasActualBeginning ;

sh:class occp:BeginningOfPlanning ;

sh:maxCount 1 ;

sh:minCount 1 ; # Muss definiert sein wegen GeneralTimeShape

sh:message "Each 'occp:PhaseA\_Planning' must have exactly one actual beginning via 'occp:hasActualBeginning' referencing 'occp:BeginningOfPlanning'."@en ;

sh:severity sh:Violation

] ;

sh:property [

sh:path occp:hasActualEnd ;

sh:or (

[ sh:class occp:ReviewApproval ]

[ sh:class occp:CompletionOfPlanning ]

) ;

sh:maxCount 1 ;

sh:minCount 1 ; # Muss definiert sein wegen GeneralTimeShape

sh:message "Each 'occp:PhaseA\_Planning' must have exactly one actual end via 'occp:hasActualEnd' referencing 'occp:ReviewApproval' or 'occp:CompletionOfPlanning'."@en ;

sh:severity sh:Violation

] ;

sh:sparql [

sh:select """

PREFIX occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#>

PREFIX time: <http://www.w3.org/2006/time#>

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

SELECT $this WHERE {

$this occp:hasActualEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

?phaseC a occp:PhaseC\_Construction .

?phaseC occp:hasActualBeginning ?startInstantC .

?startInstantC occp:hasActualTime ?startTimeC .

FILTER (?endTime >= ?startTimeC)

}

""" ;

sh:message "The end of 'occp:PhaseA\_Planning' (time: { ?endTime }) must be before the beginning of 'occp:PhaseC\_Construction' (time: { ?startTimeC })."@en ;

sh:severity sh:Violation

] ;

sh:sparql [

sh:select """

PREFIX occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#>

PREFIX time: <http://www.w3.org/2006/time#>

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

SELECT $this WHERE {

$this occp:hasActualBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

?phaseB a occp:PhaseB\_Review .

?phaseB occp:hasActualBeginning ?startInstantB .

?startInstantB occp:hasActualTime ?startTimeB .

FILTER (?startTime >= ?startTimeB)

}

""" ;

sh:message "The beginning of 'occp:PhaseA\_Planning' (time: { ?startTime }) must be before the beginning of 'occp:PhaseB\_Review' (time: { ?startTimeB })."@en ;

sh:severity sh:Violation

] .

* Erklärung:
  + Die typbasierten Regeln für hasActualBeginning und hasActualEnd bleiben, um sicherzustellen, dass die Instants korrekt verknüpft sind.
  + Zwei SPARQL-Regeln prüfen die Zeitstempel:
    1. Vergleicht das Ende von PhaseA mit dem Anfang von PhaseC\_Construction.
    2. Vergleicht den Anfang von PhaseA mit dem Anfang von PhaseB\_Review.
  + FILTER stellt sicher, dass die Zeitstempel chronologisch korrekt sind (z. B. ?endTime < ?startTimeC).

2. ABox ohne direkte Beziehungen

Dein Vorschlag, die ABox ohne explizite occp:beginsBefore/occp:endsBefore-Beziehungen zu gestalten, passt dazu. Eine reduzierte ABox könnte so aussehen:

turtle

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix : <http://www.example.de/example#> .

:phaseA a occp:PhaseA\_Planning ;

occp:hasActualBeginning :instant\_beginning\_of\_planning ;

occp:hasActualEnd :instant\_completion\_of\_planning .

:phaseB a occp:PhaseB\_Review ;

occp:hasActualBeginning :instant\_submission\_to\_review ;

occp:hasActualEnd :instant\_review\_approval .

:phaseC a occp:PhaseC\_Construction ;

occp:hasActualBeginning :instant\_construction\_start ;

occp:hasActualEnd :instant\_completion\_of\_construction .

:instant\_beginning\_of\_planning a occp:BeginningOfPlanning ;

occp:hasActualTime "2025-01-01"^^xsd:date .

:instant\_completion\_of\_planning a occp:CompletionOfPlanning ;

occp:hasActualTime "2025-03-01"^^xsd:date .

:instant\_submission\_to\_review a occp:SubmissionToReview ;

occp:hasActualTime "2025-02-15"^^xsd:date .

:instant\_review\_approval a occp:ReviewApproval ;

occp:hasActualTime "2025-02-28"^^xsd:date .

:instant\_construction\_start a occp:ConstructionStart ;

occp:hasActualTime "2025-03-15"^^xsd:date .

:instant\_completion\_of\_construction a occp:CompletionOfConstruction ;

occp:hasActualTime "2025-06-01"^^xsd:date .

* Änderung: Entferne alle occp:beginsBefore, occp:endsBefore, occp:after, etc. Die zeitliche Reihenfolge wird nur über die Zeitstempel und die SHACL-Regeln geprüft.

3. Erweiterung für weitere Phasen

Wenn :phaseC hinzukommt, müssen wir sicherstellen, dass die SPARQL-Regeln flexibel genug sind, um alle relevanten Phasen zu prüfen. Die obige Regel für PhaseC\_Construction funktioniert schon, da sie alle Instanzen von occp:PhaseC\_Construction abfragt. Für komplexere Szenarien (z. B. mehrere PhaseC-Instanzen oder transitive Beziehungen) könnten wir die Abfrage anpassen:

turtle

sh:sparql [

sh:select """

PREFIX occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#>

PREFIX time: <http://www.w3.org/2006/time#>

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

SELECT $this WHERE {

$this occp:hasActualEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

?nextPhase occp:hasActualBeginning ?startInstantNext .

?startInstantNext occp:hasActualTime ?startTimeNext .

FILTER (?endTime >= ?startTimeNext)

FILTER (?nextPhase != $this)

FILTER (?nextPhase a occp:PhaseC\_Construction || ?nextPhase a occp:PhaseB\_Review)

}

""" ;

sh:message "The end of 'occp:PhaseA\_Planning' (time: { ?endTime }) must be before the beginning of a subsequent phase (e.g., PhaseB\_Review or PhaseC\_Construction, time: { ?startTimeNext })."@en ;

sh:severity sh:Violation

] .

* Vorteil: Prüft dynamisch alle Phasen, die nach PhaseA kommen könnten.

Nächste Schritte

1. SHACL mit Zeitstempeln testen:
   * Ich schlage vor, wir nehmen die reduzierte ABox (oben) und die neue osh:ShapeOfPhaseA mit SPARQL-Regeln und testen das mit deinem Script.
   * Das Script bleibt unverändert, da PySHACL SPARQL-Regeln nativ unterstützt.
2. Minimaltest:
   * Starte mit PhaseA, PhaseB, und PhaseC, wie in der reduzierten ABox gezeigt. Wenn das funktioniert (Conforms: True), können wir weitere Phasen hinzufügen.
3. Feedback einholen:
   * Du führst den Test durch und teilst mir das Log mit. Wenn es Probleme gibt, passen wir die SPARQL-Abfragen an.

Meine Meinung

Dein Ansatz, die Zeitstempelprüfung einzubauen, ist der richtige Weg, denn er bringt uns näher an die eigentliche Intention der OCCP: chronologische Konsistenz sicherzustellen. Die Typprüfungen allein sind zu statisch und führen zu den aktuellen Fehlern, weil sie die dynamische Natur der Phasenfolge nicht abbilden. Mit SPARQL können wir das elegant lösen.

Frage an dich

* Soll ich dir die neue OCCP\_SHACL\_min.ttl mit den SPARQL-Regeln schicken, oder möchtest du sie selbst anpassen?
* Willst du die reduzierte ABox direkt übernehmen, oder hast du eine andere Vorstellung, wie sie aussehen soll?

Lass uns das zusammen umsetzen – ich denke, wir sind kurz davor, die Validierung richtig hinzubekommen! Was meinst du?

Ok, vielen Dank! ich denke, die Einführung von SPARQL-Abfragen für die Überprüfung der Reihenfolgen anhand der Zeitstempel ist essentiell. Frage: Wäre es möglich, dass wir dafür allgemeine Regeln aufstellen statt für jede Phase individuelle Abfragen zu erstellen? Bsp.: Wenn Instanz A endeBefore Instanz B, dann: Ermittle, wann Instanz A endet (Auslesen des Zeitstempels der als Ende für Instanz A verwendeten Instanz) und wann Instanz B beginnt (analog zu Instanz A) und prüfe, ob A vor B liegt. Diese Logik können wir für alle Object properties definieren und immer wieder aufgreifen - oder?

Du hast absolut recht – die Idee, allgemeine SPARQL-Regeln für die Zeitstempelprüfung zu definieren, die auf Object Properties wie occp:endsBefore, occp:beginsBefore, etc. anwendbar sind, ist nicht nur effizient, sondern auch elegant und wiederverwendbar. Das würde die SHACL-Datei deutlich schlanker machen und die Wartbarkeit verbessern, da wir nicht für jede Phase individuelle Regeln schreiben müssen. Lass uns das konzeptuell durchdenken und umsetzen!

Konzept: Allgemeine Zeitstempel-basierte Regeln

Dein Beispiel ist ein perfekter Ausgangspunkt:

* Logik: Wenn eine Instanz A via occp:endsBefore mit Instanz B verknüpft ist, dann:
  1. Ermittle den Zeitstempel des Endes von Instanz A (über occp:hasActualEnd → occp:hasActualTime).
  2. Ermittle den Zeitstempel des Beginns von Instanz B (über occp:hasActualBeginning → occp:hasActualTime).
  3. Prüfe, ob der Endzeitpunkt von A vor dem Startzeitpunkt von B liegt (?endTimeA < ?startTimeB).

Diese Logik können wir auf alle relevanten Object Properties anwenden, die chronologische Beziehungen definieren, z. B.:

* occp:endsBefore
* occp:beginsBefore
* occp:before
* occp:after
* occp:endsAfter
* occp:beginsAfter

Vorteile

* Wiederverwendbarkeit: Eine einzige Regel kann für alle Phasen, Zyklen oder Instants gelten, solange sie diese Properties verwenden.
* Flexibilität: Funktioniert unabhängig davon, ob die ABox explizite Beziehungen (z. B. :phaseA occp:endsBefore :phaseC) enthält oder ob die Reihenfolge implizit aus den Zeitstempeln abgeleitet wird.
* Skalierbarkeit: Wenn neue Phasen oder Beziehungen hinzukommen, müssen wir nur die ABox erweitern, nicht die SHACL-Regeln.

Herausforderung

* Generalisierung: Die Regel muss so formuliert sein, dass sie sowohl direkte Beziehungen (z. B. occp:endsBefore) als auch indirekte Beziehungen (z. B. Zeitstempelvergleich ohne explizite Property) handhaben kann.
* Fehlerlokalisierung: Die Fehlermeldungen sollten spezifisch genug sein, um den Nutzer zu informieren, welche Instanzen und Zeitstempel betroffen sind.

Umsetzung: Allgemeine SHACL-Regeln mit SPARQL

Ich schlage vor, wir definieren eine generische Regel für jede der chronologischen Object Properties und binden sie an time:TemporalEntity (oder eine spezifischere Oberklasse wie occp:Phase, falls gewünscht). Hier ein Beispiel für occp:endsBefore:

turtle

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

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix osh: <http://www.occpshape.de/shape#> .

### Generische Zeitstempel-Regeln

osh:EndsBeforeTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ; # Gilt für alle Phasen, Zyklen, etc.

sh:sparql [

sh:select """

PREFIX occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#>

PREFIX time: <http://www.w3.org/2006/time#>

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

SELECT $this ?other ?endTime ?startTime WHERE {

$this occp:endsBefore ?other .

$this occp:hasActualEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

?other occp:hasActualBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

FILTER (?endTime >= ?startTime)

}

""" ;

sh:message "The end of { $this } (time: { ?endTime }) must be before the beginning of { ?other } (time: { ?startTime }) via 'occp:endsBefore'."@en ;

sh:severity sh:Violation

] .

osh:BeginsBeforeTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

PREFIX occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#>

PREFIX time: <http://www.w3.org/2006/time#>

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

SELECT $this ?other ?startTime ?otherStartTime WHERE {

$this occp:beginsBefore ?other .

$this occp:hasActualBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

?other occp:hasActualBeginning ?otherStartInstant .

?otherStartInstant occp:hasActualTime ?otherStartTime .

FILTER (?startTime >= ?otherStartTime)

}

""" ;

sh:message "The beginning of { $this } (time: { ?startTime }) must be before the beginning of { ?other } (time: { ?otherStartTime }) via 'occp:beginsBefore'."@en ;

sh:severity sh:Violation

] .

### Allgemeine Struktur-Regeln (bleiben bestehen)

osh:GeneralTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:property [

sh:path ( time:hasBeginning time:hasEnd ) ;

sh:class time Maf ;

sh:minCount 1 ;

sh:message "At least one 'time:hasBeginning' or 'time:hasEnd' must reference a 'time:Instant'."@en ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:hasActualBeginning ;

sh:class time:Instant ;

sh:maxCount 1 ;

sh:minCount 1 ; # Muss definiert sein

sh:message "Each entity must have exactly one actual beginning via occp:hasActualBeginning."@en ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:hasActualEnd ;

sh:class time:Instant ;

sh:maxCount 1 ;

sh:minCount 1 ; # Muss definiert sein

sh:message "Each entity must have exactly one actual end via occp:hasActualEnd."@en ;

sh:severity sh:Violation ;

] .

osh:InstantTimeShape a sh:NodeShape ;

sh:targetClass time:Instant ;

sh:property [

sh:path occp:hasActualTime ;

sh:datatype xsd:date ;

sh:maxCount 1 ;

sh:minCount 1 ; # Muss definiert sein

sh:message "Each instant must have exactly one actual timestamp via occp:hasActualTime of type xsd:date."@en ;

sh:severity sh:Violation

] .

Erklärung

* osh:EndsBeforeTimeShape:
  + Prüft alle Instanzen, die occp:endsBefore verwenden.
  + Vergleicht den Endzeitpunkt von $this mit dem Startzeitpunkt von ?other.
  + Meldet einen Verstoß, wenn ?endTime >= ?startTime.
* osh:BeginsBeforeTimeShape:
  + Prüft alle Instanzen mit occp:beginsBefore.
  + Vergleicht den Startzeitpunkt von $this mit dem Startzeitpunkt von ?other.
* Allgemeine Regeln: Stellen sicher, dass jede Phase/Instanz die notwendigen Zeitstempel hat (minCount 1).

ABox-Beispiel

turtle

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix : <http://www.example.de/example#> .

:phaseA a occp:PhaseA\_Planning ;

occp:hasActualBeginning :instant\_beginning\_of\_planning ;

occp:hasActualEnd :instant\_completion\_of\_planning ;

occp:endsBefore :phaseC .

:phaseB a occp:PhaseB\_Review ;

occp:hasActualBeginning :instant\_submission\_to\_review ;

occp:hasActualEnd :instant\_review\_approval ;

occp:beginsBefore :phaseC .

:phaseC a occp:PhaseC\_Construction ;

occp:hasActualBeginning :instant\_construction\_start ;

occp:hasActualEnd :instant\_completion\_of\_construction .

:instant\_beginning\_of\_planning a occp:BeginningOfPlanning ;

occp:hasActualTime "2025-01-01"^^xsd:date .

:instant\_completion\_of\_planning a occp:CompletionOfPlanning ;

occp:hasActualTime "2025-03-01"^^xsd:date .

:instant\_submission\_to\_review a occp:SubmissionToReview ;

occp:hasActualTime "2025-02-15"^^xsd:date .

:instant\_review\_approval a occp:ReviewApproval ;

occp:hasActualTime "2025-02-28"^^xsd:date .

:instant\_construction\_start a occp:ConstructionStart ;

occp:hasActualTime "2025-03-15"^^xsd:date .

:instant\_completion\_of\_construction a occp:CompletionOfConstruction ;

occp:hasActualTime "2025-06-01"^^xsd:date .

* Test: Diese ABox sollte Conforms: True ergeben, da "2025-03-01" < "2025-03-15" (für endsBefore) und "2025-02-15" < "2025-03-15" (für beginsBefore).

Erweiterung für andere Properties

Wir können ähnliche Regeln für occp:before, occp:after, etc. erstellen:

* occp:before:

turtle

osh:BeforeTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

SELECT $this ?other ?endTime ?startTime WHERE {

$this occp:before ?other .

$this occp:hasActualEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

?other occp:hasActualBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

FILTER (?endTime >= ?startTime)

}

""" ;

sh:message "The end of { $this } (time: { ?endTime }) must be before the beginning of { ?other } (time: { ?startTime }) via 'occp:before'."@en ;

sh:severity sh:Violation

] .

Ohne explizite Beziehungen

Falls die ABox keine occp:endsBefore/occp:beginsBefore enthält, könnten wir eine Regel schreiben, die die implizite Reihenfolge prüft:

turtle

osh:PhaseOrderTimeShape a sh:NodeShape ;

sh:targetClass occp:Phase ;

sh:sparql [

sh:select """

PREFIX occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#>

PREFIX time: <http://www.w3.org/2006/time#>

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

SELECT $this ?other ?endTime ?startTime WHERE {

$this occp:hasActualEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

?other occp:hasActualBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

FILTER (?endTime >= ?startTime)

FILTER (?other != $this)

FILTER (

($this a occp:PhaseA\_Planning && ?other a occp:PhaseC\_Construction) ||

($this a occp:PhaseB\_Review && ?other a occp:PhaseC\_Construction)

)

}

""" ;

sh:message "The end of { $this } (time: { ?endTime }) must be before the beginning of { ?other } (time: { ?startTime }) based on phase order."@en ;

sh:severity sh:Violation

] .

* Hinweis: Hier müssen wir die erlaubten Reihenfolgen (z. B. A vor C, B vor C) explizit im FILTER auflisten, was weniger generisch ist.

Nächste Schritte

1. Minimalversion testen:
   * Nutze die oben gezeigten osh:EndsBeforeTimeShape und osh:BeginsBeforeTimeShape in einer neuen OCCP\_SHACL\_min.ttl.
   * Teste mit der ABox, die occp:endsBefore und occp:beginsBefore enthält.
2. Erweitern:
   * Füge Regeln für occp:before, occp:after, etc. hinzu.
   * Optional: Teste eine ABox ohne explizite Beziehungen mit der impliziten Regel.
3. Feedback:
   * Du führst den Test durch und teilst mir das Log mit.

Antwort auf deine Frage

Ja, wir können absolut allgemeine Regeln für alle Object Properties definieren! Die SPARQL-basierte Lösung oben ist genau dafür gemacht. Sie ist wiederverwendbar und kann auf jede Property angewendet werden, die eine chronologische Beziehung definiert. Die Herausforderung ohne explizite Beziehungen lösen wir entweder durch eine implizite Regel (mit festgelegter Reihenfolge) oder durch Reasoning in der TBox (z. B. transitive Eigenschaften), aber für den Anfang reicht die explizite Variante.

Vorschlag

Ich schicke dir eine OCCP\_SHACL\_min.ttl mit den generischen Regeln für endsBefore und beginsBefore. Du kannst sie mit deiner ABox testen. Soll ich das machen, oder willst du die Regeln selbst schreiben? Wie siehst du das?

Vielen Dank! In Deinem Codevorschlag für die SHACL-Regeln mit SPARQL steht u.a. folgende Zeile: sh:class time Maf ;Dafür habe ich sh:class time:Instant ; eingesetzt. Der Test mit Deiner angepassten, minimalen ABox lief durch (vielen Dank für Deine Vorschläge, sehr wertvoll!!) und das Ergebnis ist:2025-03-12 17:22:19,631 - INFO - Konformit�t (inference=none): True:-)Yeah! Danke!Aus meiner Sicht dürfte in der ABox bei PhaseA und B nicht occp:endsBefore :phaseC drinstehen, denn das ist überflüssig (weil die Reihenfolge von den Verknüpfungen mit den Instanzen definiert wird und die TBox die zulässigen Verknüpfungen und die zulässige Reihenfolge (im Bezug auf die Zeitstempel) vorgibt). Ich habe die beiden Angaben mal rausgelöscht und das Ergebnis der Validierung war (erwartungsgemäß) dasselbe. Aber es ist wichtig, dass die ABoxen nur die Informationen enthalten, die sie enthalten müssen, um eine vollständige Chronologie herstellen zu können.Wie wollen wir weitermachen?

Super, das freut mich riesig – Conforms: True ist ein großer Erfolg! Und danke fürs Korrigieren von sh:class time Maf zu sh:class time:Instant – das war definitiv ein Tippfehler meinerseits (wahrscheinlich ein Copy-Paste-Fehler). Schön, dass der Test mit der minimalen ABox durchgelaufen ist und auch ohne die expliziten occp:endsBefore-Angaben funktioniert hat. Dein Punkt zur ABox ist absolut richtig, und ich stimme dir zu, dass wir jetzt eine solide Basis haben, um weiterzumachen.

Analyse des aktuellen Standes

* Zeitstempelprüfung funktioniert: Die SPARQL-Regeln prüfen die chronologische Reihenfolge korrekt anhand der Zeitstempel, wenn occp:endsBefore oder occp:beginsBefore vorhanden sind.
* Redundanz in der ABox: Du hast vollkommen recht, dass occp:endsBefore :phaseC in :phaseA und :phaseB überflüssig ist, solange die Zeitstempel und die Verknüpfungen zu den Instants (z. B. :instant\_completion\_of\_planning) die Reihenfolge definieren. Die TBox und SHACL-Regeln sollten die erlaubten Verknüpfungen und die chronologische Logik abdecken, während die ABox nur die minimal notwendigen Daten liefert (Phasen + Start/End-Instant mit Zeitstempeln).
* Minimalistische ABox: Dein Test ohne diese Beziehungen zeigt, dass wir die ABox schlank halten können, was die Datenpflege erleichtert und Redundanzen vermeidet.

Dein Punkt zur ABox-Philosophie

Ich unterstütze deine Sichtweise voll und ganz:

* Die ABox sollte nur die Rohdaten enthalten: Phasen, ihre Start- und End-Instant sowie deren Zeitstempel.
* Die Reihenfolge und Constraints (z. B. "Phase A muss vor Phase C enden") sollten aus der TBox (Ontologie-Definitionen) und den SHACL-Regeln abgeleitet werden.
* Das bedeutet, dass wir die SHACL-Regeln so anpassen müssen, dass sie die chronologische Reihenfolge auch ohne explizite Beziehungen wie occp:endsBefore prüfen können – basierend auf den Typen der Phasen und den Zeitstempeln.

Wie weitermachen?

Wir haben jetzt zwei Optionen, wie wir die Validierung erweitern können, um deine Vision vollständig umzusetzen:

1. SHACL-Regeln ohne explizite Beziehungen (implizite Reihenfolge)

Da du die occp:endsBefore/occp:beginsBefore-Angaben aus der ABox entfernen möchtest, müssen wir die SHACL-Regeln so umstellen, dass sie die Reihenfolge basierend auf den Phasen-Typen und Zeitstempeln prüfen, ohne auf direkte Beziehungen angewiesen zu sein. Beispiel:

turtle

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

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix osh: <http://www.occpshape.de/shape#> .

### Allgemeine Struktur-Regeln

osh:GeneralTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:property [

sh:path ( time:hasBeginning time:hasEnd ) ;

sh:class time:Instant ;

sh:minCount 1 ;

sh:message "At least one 'time:hasBeginning' or 'time:hasEnd' must reference a 'time:Instant'."@en ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:hasActualBeginning ;

sh:class time:Instant ;

sh:maxCount 1 ;

sh:minCount 1 ;

sh:message "Each entity must have exactly one actual beginning via occp:hasActualBeginning."@en ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:hasActualEnd ;

sh:class time:Instant ;

sh:maxCount 1 ;

sh:minCount 1 ;

sh:message "Each entity must have exactly one actual end via occp:hasActualEnd."@en ;

sh:severity sh:Violation ;

] .

osh:InstantTimeShape a sh:NodeShape ;

sh:targetClass time:Instant ;

sh:property [

sh:path occp:hasActualTime ;

sh:datatype xsd:date ;

sh:maxCount 1 ;

sh:minCount 1 ;

sh:message "Each instant must have exactly one actual timestamp via occp:hasActualTime of type xsd:date."@en ;

sh:severity sh:Violation

] .

### Implizite Reihenfolge-Regel

osh:PhaseOrderTimeShape a sh:NodeShape ;

sh:targetClass occp:Phase ;

sh:sparql [

sh:select """

PREFIX occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#>

PREFIX time: <http://www.w3.org/2006/time#>

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

SELECT $this ?other ?endTime ?startTime WHERE {

$this occp:hasActualEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

?other occp:hasActualBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

FILTER (?endTime >= ?startTime)

FILTER (?other != $this)

FILTER (

($this a occp:PhaseA\_Planning && ?other a occp:PhaseB\_Review) ||

($this a occp:PhaseA\_Planning && ?other a occp:PhaseC\_Construction) ||

($this a occp:PhaseB\_Review && ?other a occp:PhaseC\_Construction)

)

}

""" ;

sh:message "The end of { $this } (time: { ?endTime }) must be before the beginning of { ?other } (time: { ?startTime }) based on phase order."@en ;

sh:severity sh:Violation

] .

* ABox-Test:

turtle

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix : <http://www.example.de/example#> .

:phaseA a occp:PhaseA\_Planning ;

occp:hasActualBeginning :instant\_beginning\_of\_planning ;

occp:hasActualEnd :instant\_completion\_of\_planning .

:phaseB a occp:PhaseB\_Review ;

occp:hasActualBeginning :instant\_submission\_to\_review ;

occp:hasActualEnd :instant\_review\_approval .

:phaseC a occp:PhaseC\_Construction ;

occp:hasActualBeginning :instant\_construction\_start ;

occp:hasActualEnd :instant\_completion\_of\_construction .

:instant\_beginning\_of\_planning a occp:BeginningOfPlanning ;

occp:hasActualTime "2025-01-01"^^xsd:date .

:instant\_completion\_of\_planning a occp:CompletionOfPlanning ;

occp:hasActualTime "2025-03-01"^^xsd:date .

:instant\_submission\_to\_review a occp:SubmissionToReview ;

occp:hasActualTime "2025-02-15"^^xsd:date .

:instant\_review\_approval a occp:ReviewApproval ;

occp:hasActualTime "2025-02-28"^^xsd:date .

:instant\_construction\_start a occp:ConstructionStart ;

occp:hasActualTime "2025-03-15"^^xsd:date .

:instant\_completion\_of\_construction a occp:CompletionOfConstruction ;

occp:hasActualTime "2025-06-01"^^xsd:date .

* Erwartung: Conforms: True, da die Zeitstempel die Reihenfolge A → B → C einhalten.

2. Vollständige SHACL-Datei erweitern

Sobald die implizite Regel funktioniert, können wir die vollständige OCCP\_SHACL.ttl wieder einbeziehen und:

* Die typbasierten Regeln (z. B. sh:class) beibehalten, um die erlaubten Instants pro Phase zu prüfen (z. B. PhaseA endet mit ReviewApproval oder CompletionOfPlanning).
* Die SPARQL-Regeln für die Zeitstempelprüfung hinzufügen, entweder generisch (wie oben) oder spezifisch für bestimmte Phasen, falls du doch mehr Granularität willst.

Mein Vorschlag

* Schritt 1: Teste die obige osh:PhaseOrderTimeShape mit der reduzierten ABox ohne explizite Beziehungen. Das validiert deine Idee, die ABox minimalistisch zu halten.
* Schritt 2: Wenn das klappt, erweitern wir die SPARQL-Regel, um alle Phasen aus deiner ursprünglichen Ontologie abzudecken (z. B. Phase D, E, F, etc.) – das FILTER wird dann länger, aber bleibt generisch.
* Schritt 3: Integriere die generische Regel in die volle OCCP\_SHACL.ttl und teste mit deiner kompletten ABox (OCCP\_Valid\_LCycle\_1.ttl), natürlich ohne die redundanten Beziehungen.

Wie vorgehen?

* Ich kann dir die OCCP\_SHACL\_min.ttl mit der impliziten osh:PhaseOrderTimeShape-Regel schicken, und du testest sie mit der reduzierten ABox.
* Alternativ kannst du die Regel selbst in deine aktuelle OCCP\_SHACL\_min.ttl einfügen und testen.

Was meinst du? Soll ich dir die Datei schicken, oder gehst du direkt ran? Und wie siehst du die Erweiterung auf die volle Ontologie – sollen wir das gleich angehen oder erst die implizite Regel perfektionieren? Ich freue mich auf dein Feedback – wir sind echt gut vorangekommen!

Sehr gut, danke! Ja, lass uns die generischen SPARQL-Abfragen erstellen, einbauen und testen. Ich denke, wenn wir die hinbekommen, dann reicht das erstmal für ein Minimalbeispiel aus (und wenn ich später die Zeit finde, kann man die generischen Regeln aufgreifen und an die einzelnen Phasen und ggf. Instants individuell anpassen). Wichtig ist, dass wir keine Object Property (OP) vergessen (bei der Erstellung unserer SPARQL-Abfragen (SA)). Ich denke, wir werden wahrscheinlich auch eine SA für time:after und time:before (und ggf. weitere time:OPs) benötigen (nicht nur für die ganzen occp:OPs). Aber eins nach dem anderen. Lass uns mit OCCP after und before beginnen, oder was sagst Du?

Alles klar, ich bin dabei! Dein Plan, mit generischen SPARQL-Abfragen (SAs) zu starten und sie später bei Bedarf an spezifische Phasen oder Instants anzupassen, ist perfekt. Und du hast recht – wir müssen sicherstellen, dass wir keine relevante Object Property (OP) vergessen, sowohl aus dem occp:-Namespace als auch aus time:, falls sie in der Ontologie verwendet werden. Ich stimme dir zu, dass wir mit occp:after und occp:before beginnen sollten, da das die Kernbeziehungen für die chronologische Reihenfolge sind. Danach können wir schrittweise erweitern (z. B. time:after, time:before, und andere occp:-OPs wie endsBefore, beginsBefore).

Strategie

* Ziel: Generische SAs für occp:after und occp:before, die Zeitstempel prüfen, unabhängig von spezifischen Phasen.
* Minimalbeispiel: Wir testen sie mit einer kleinen ABox, um sicherzugehen, dass sie funktionieren.
* Erweiterung: Später fügen wir SAs für weitere OPs hinzu (z. B. time:after, time:before, occp:endsBefore) und prüfen, ob wir sie anpassen müssen.

Generische SPARQL-Regeln für occp:after und occp:before

Hier sind die SAs für occp:after und occp:before, die auf time:TemporalEntity angewendet werden, um maximale Generizität zu gewährleisten:

turtle

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

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix osh: <http://www.occpshape.de/shape#> .

### Allgemeine Struktur-Regeln

osh:GeneralTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:property [

sh:path ( time:hasBeginning time:hasEnd ) ;

sh:class time:Instant ;

sh:minCount 1 ;

sh:message "At least one 'time:hasBeginning' or 'time:hasEnd' must reference a 'time:Instant'."@en ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:hasActualBeginning ;

sh:class time:Instant ;

sh:maxCount 1 ;

sh:minCount 1 ;

sh:message "Each entity must have exactly one actual beginning via occp:hasActualBeginning."@en ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:hasActualEnd ;

sh:class time:Instant ;

sh:maxCount 1 ;

sh:minCount 1 ;

sh:message "Each entity must have exactly one actual end via occp:hasActualEnd."@en ;

sh:severity sh:Violation ;

] .

osh:InstantTimeShape a sh:NodeShape ;

sh:targetClass time:Instant ;

sh:property [

sh:path occp:hasActualTime ;

sh:datatype xsd:date ;

sh:maxCount 1 ;

sh:minCount 1 ;

sh:message "Each instant must have exactly one actual timestamp via occp:hasActualTime of type xsd:date."@en ;

sh:severity sh:Violation

] .

### Generische Zeitstempel-Regeln für occp:after

osh:AfterTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

PREFIX occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#>

PREFIX time: <http://www.w3.org/2006/time#>

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

SELECT $this ?other ?startTime ?endTime WHERE {

$this occp:after ?other .

$this occp:hasActualBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

?other occp:hasActualEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

FILTER (?startTime <= ?endTime)

}

""" ;

sh:message "The beginning of { $this } (time: { ?startTime }) must be after the end of { ?other } (time: { ?endTime }) via 'occp:after'."@en ;

sh:severity sh:Violation

] .

### Generische Zeitstempel-Regeln für occp:before

osh:BeforeTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

PREFIX occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#>

PREFIX time: <http://www.w3.org/2006/time#>

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

SELECT $this ?other ?endTime ?startTime WHERE {

$this occp:before ?other .

$this occp:hasActualEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

?other occp:hasActualBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

FILTER (?endTime >= ?startTime)

}

""" ;

sh:message "The end of { $this } (time: { ?endTime }) must be before the beginning of { ?other } (time: { ?startTime }) via 'occp:before'."@en ;

sh:severity sh:Violation

] .

Erklärung

* osh:AfterTimeShape:
  + Prüft: Wenn $this occp:after ?other, dann muss der Start von $this nach dem Ende von ?other liegen (?startTime > ?endTime).
  + Fehler: Wenn ?startTime <= ?endTime, wird ein Verstoß gemeldet.
* osh:BeforeTimeShape:
  + Prüft: Wenn $this occp:before ?other, dann muss das Ende von $this vor dem Start von ?other liegen (?endTime < ?startTime).
  + Fehler: Wenn ?endTime >= ?startTime, wird ein Verstoß gemeldet.
* Struktur-Regeln: Stellen sicher, dass jede Phase/Instanz die notwendigen Zeitstempel hat.

Test-ABox

Um das zu testen, nehmen wir eine kleine ABox mit expliziten occp:after und occp:before-Beziehungen:

turtle

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix : <http://www.example.de/example#> .

:phaseA a occp:PhaseA\_Planning ;

occp:hasActualBeginning :instant\_beginning\_of\_planning ;

occp:hasActualEnd :instant\_completion\_of\_planning ;

occp:before :phaseB .

:phaseB a occp:PhaseB\_Review ;

occp:hasActualBeginning :instant\_submission\_to\_review ;

occp:hasActualEnd :instant\_review\_approval ;

occp:after :phaseA ;

occp:before :phaseC .

:phaseC a occp:PhaseC\_Construction ;

occp:hasActualBeginning :instant\_construction\_start ;

occp:hasActualEnd :instant\_completion\_of\_construction ;

occp:after :phaseB .

:instant\_beginning\_of\_planning a occp:BeginningOfPlanning ;

occp:hasActualTime "2025-01-01"^^xsd:date .

:instant\_completion\_of\_planning a occp:CompletionOfPlanning ;

occp:hasActualTime "2025-03-01"^^xsd:date .

:instant\_submission\_to\_review a occp:SubmissionToReview ;

occp:hasActualTime "2025-02-15"^^xsd:date .

:instant\_review\_approval a occp:ReviewApproval ;

occp:hasActualTime "2025-02-28"^^xsd:date .

:instant\_construction\_start a occp:ConstructionStart ;

occp:hasActualTime "2025-03-15"^^xsd:date .

:instant\_completion\_of\_construction a occp:CompletionOfConstruction ;

occp:hasActualTime "2025-06-01"^^xsd:date .

* Erwartung: Conforms: True, da:
  + PhaseA (Ende: 2025-03-01) vor PhaseB (Start: 2025-02-15) → Fehler, weil 2025-03-01 > 2025-02-15!
  + PhaseB (Start: 2025-02-15) nach PhaseA (Ende: 2025-03-01) → Fehler, weil 2025-02-15 < 2025-03-01!
  + PhaseC (Start: 2025-03-15) nach PhaseB (Ende: 2025-02-28) → OK, weil 2025-03-15 > 2025-02-28.

Moment – die ABox hat hier ein Problem! Die Zeitstempel sind inkonsistent mit den Beziehungen:

* PhaseA occp:before PhaseB verlangt, dass 2025-03-01 < 2025-02-15 → falsch.
* PhaseB occp:after PhaseA verlangt, dass 2025-02-15 > 2025-03-01 → falsch.

Korrigierte ABox

turtle

:phaseA a occp:PhaseA\_Planning ;

occp:hasActualBeginning :instant\_beginning\_of\_planning ;

occp:hasActualEnd :instant\_completion\_of\_planning ;

occp:before :phaseB .

:phaseB a occp:PhaseB\_Review ;

occp:hasActualBeginning :instant\_submission\_to\_review ;

occp:hasActualEnd :instant\_review\_approval ;

occp:after :phaseA ;

occp:before :phaseC .

:phaseC a occp:PhaseC\_Construction ;

occp:hasActualBeginning :instant\_construction\_start ;

occp:hasActualEnd :instant\_completion\_of\_construction ;

occp:after :phaseB .

:instant\_beginning\_of\_planning a occp:BeginningOfPlanning ;

occp:hasActualTime "2025-01-01"^^xsd:date .

:instant\_completion\_of\_planning a occp:CompletionOfPlanning ;

occp:hasActualTime "2025-02-01"^^xsd:date . # Geändert

:instant\_submission\_to\_review a occp:SubmissionToReview ;

occp:hasActualTime "2025-02-15"^^xsd:date .

:instant\_review\_approval a occp:ReviewApproval ;

occp:hasActualTime "2025-02-28"^^xsd:date .

:instant\_construction\_start a occp:ConstructionStart ;

occp:hasActualTime "2025-03-15"^^xsd:date .

:instant\_completion\_of\_construction a occp:CompletionOfConstruction ;

occp:hasActualTime "2025-06-01"^^xsd:date .

* Jetzt:
  + PhaseA (Ende: 2025-02-01) vor PhaseB (Start: 2025-02-15) → OK.
  + PhaseB (Start: 2025-02-15) nach PhaseA (Ende: 2025-02-01) → OK.
  + PhaseC (Start: 2025-03-15) nach PhaseB (Ende: 2025-02-28) → OK.
  + Erwartung: Conforms: True.

Nächste Schritte

1. Testen:
   * Nutze die obige SHACL-Datei (OCCP\_SHACL\_min.ttl) mit den SAs für occp:after und occp:before.
   * Teste mit der korrigierten ABox.
2. Erweitern:
   * Füge SAs für weitere occp:-OPs hinzu, z. B.:
     + occp:endsBefore
     + occp:beginsBefore
     + occp:endsAfter
     + occp:beginsAfter
   * Danach können wir time:after und time:before ergänzen, falls sie in deiner TBox verwendet werden.
3. Feedback:
   * Du führst den Test durch und teilst mir das Log mit. Wenn es Probleme gibt, passen wir die SAs an.

Meine Meinung

Ich denke, mit occp:after und occp:before zu starten ist ideal, da sie die grundlegendsten chronologischen Beziehungen abdecken. Sobald wir das stabil haben, können wir die anderen OPs schrittweise einbauen. Dein Hinweis auf time:after und time:before ist wichtig – wir sollten checken, ob sie in OCCP\_V0.3.ttl vorkommen, und sie dann bei Bedarf ergänzen.

Soll ich dir die OCCP\_SHACL\_min.ttl mit den beiden SAs schicken, oder passt du sie selbst an? Und wie siehst du die Reihenfolge für die weiteren OPs? Lass uns das rocken!

Vielen Dank! Müssen wir die PREFIXes in jeder SA von Neuem angeben? Vor allem, wenn diese bereits in der TTL-Datei am Anfang definiert wurden? Und: Du nutzt in Deiner Abfrage Actual-Zeiten - sollten wir da vielleicht (um es maximal generisch zu gestalten) die übergeordneten OPs (also z.B. hasBeginning anstelle von hasActualBeginning) nutzen oder spricht da aus Deiner Sicht etwas dagegen? Deine korrigierte ABox ist an mehreren Stellen falsch. Du hast wieder zusätzliche Beziehungen (occp:before :phaseB) gemacht, die wir nicht brauchen. Außerdem hast Du den Zeitpunkt für CompletionOfPlanning VOR SubmissionToReview gelegt, das wäre auch nicht möglich. Bitte prüf das noch einmal und korrigiere es.

Danke für dein Feedback! Du hast vollkommen recht – ich werde deine Punkte Schritt für Schritt adressieren und die Vorschläge entsprechend anpassen. Lass uns das klären und korrigieren!

1. PREFIXes in SPARQL-Abfragen

* Frage: Müssen wir die PREFIXes in jeder SA neu angeben, wenn sie schon am Anfang der TTL-Datei definiert sind?
* Antwort: Nein, das ist nicht nötig! Die PREFIX-Definitionen am Anfang der TTL-Datei gelten für die gesamte Datei, einschließlich aller eingebetteten SPARQL-Abfragen. SHACL (und RDF im Allgemeinen) übernimmt die Namespace-Definitionen aus dem Dokument-Kontext. Ich hatte die PREFIXes in den SAs aus Gewohnheit und für Lesbarkeit wiederholt, aber wir können sie weglassen, da sie redundant sind. Ich werde das in der überarbeiteten Version korrigieren.

2. Verwendung von hasActualBeginning vs. time:hasBeginning

* Dein Punkt: Sollten wir generischere OPs wie time:hasBeginning statt occp:hasActualBeginning verwenden, um maximale Flexibilität zu gewährleisten?
* Meine Sicht: Das ist eine ausgezeichnete Idee! time:hasBeginning und time:hasEnd sind Teil der W3C TIME-Ontologie und übergeordnet zu den spezifischen occp:-Properties wie hasActualBeginning und hasActualEnd. Wenn wir time:hasBeginning/time:hasEnd verwenden:
  + Vorteile:
    - Generischer: Funktioniert auch, wenn jemand zusätzlich zu hasActualBeginning andere Subproperties definiert (z. B. hasEstimatedBeginning).
    - Konsistenter mit der TIME-Ontologie, die deine OCCP erweitert.
    - Weniger Abhängigkeit von spezifischen occp:-Properties, was die Regeln robuster macht.
  + Gegenargumente:
    - Wenn deine Ontologie strikt nur occp:hasActualBeginning/occp:hasActualEnd erlaubt (z. B. durch Domain/Range-Einschränkungen oder SHACL-Regeln), könnten wir uns darauf beschränken, um spezifischer zu sein.
    - time:hasBeginning könnte mehrere Instants erlauben (kein maxCount 1), während hasActualBeginning durch die bestehenden SHACL-Regeln auf genau einen Instant beschränkt ist.
  + Fazit: Da wir generische Regeln wollen und deine osh:GeneralTimeShape schon sicherstellt, dass time:hasBeginning/time:hasEnd mindestens einen Instant haben (und hasActualBeginning/hasActualEnd zusätzlich maxCount 1 erzwingen), können wir time:hasBeginning/time:hasEnd verwenden. Das macht die SAs flexibler und zukunftssicher. Ich passe das an!

3. Korrektur der ABox

* Deine Kritik:
  1. Ich habe unnötige Beziehungen wie occp:before :phaseB hinzugefügt, obwohl wir die ABox minimalistisch halten wollen.
  2. Der Zeitpunkt für instant\_completion\_of\_planning (2025-02-01) liegt vor instant\_submission\_to\_review (2025-02-15), was chronologisch nicht zulässig ist, da CompletionOfPlanning nach SubmissionToReview kommen muss.
* Analyse:
  1. Überflüssige Beziehungen: Mea culpa – ich wollte die Beziehungen nur für den Test explizit machen, aber das widerspricht unserem Ziel, die ABox schlank zu halten. Ich entferne sie komplett.
  2. Zeitstempel-Fehler: Du hast recht, die Reihenfolge ist falsch. In deiner Ontologie muss SubmissionToReview (Start von Phase B) vor CompletionOfPlanning (Ende von Phase A) liegen, da die Planungsphase (A) erst mit der Fertigstellung nach der Überprüfung (B) endet. Ich hatte die Zeitstempel vertauscht, um die after-Regel zu testen, ohne die Logik deiner Ontologie zu prüfen. Das korrigiere ich jetzt.

Korrigierte ABox (minimalistisch)

turtle

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix : <http://www.example.de/example#> .

:phaseA a occp:PhaseA\_Planning ;

occp:hasActualBeginning :instant\_beginning\_of\_planning ;

occp:hasActualEnd :instant\_completion\_of\_planning .

:phaseB a occp:PhaseB\_Review ;

occp:hasActualBeginning :instant\_submission\_to\_review ;

occp:hasActualEnd :instant\_review\_approval .

:phaseC a occp:PhaseC\_Construction ;

occp:hasActualBeginning :instant\_construction\_start ;

occp:hasActualEnd :instant\_completion\_of\_construction .

:instant\_beginning\_of\_planning a occp:BeginningOfPlanning ;

occp:hasActualTime "2025-01-01"^^xsd:date .

:instant\_submission\_to\_review a occp:SubmissionToReview ;

occp:hasActualTime "2025-02-15"^^xsd:date .

:instant\_review\_approval a occp:ReviewApproval ;

occp:hasActualTime "2025-02-28"^^xsd:date .

:instant\_completion\_of\_planning a occp:CompletionOfPlanning ;

occp:hasActualTime "2025-03-01"^^xsd:date . # Nach ReviewApproval

:instant\_construction\_start a occp:ConstructionStart ;

occp:hasActualTime "2025-03-15"^^xsd:date .

:instant\_completion\_of\_construction a occp:CompletionOfConstruction ;

occp:hasActualTime "2025-06-01"^^xsd:date .

* Änderungen:
  + Entfernt: Alle occp:before, occp:after-Beziehungen, um die ABox rein auf Phasen und Zeitstempel zu beschränken.
  + Korrigierte Zeitstempel: instant\_completion\_of\_planning jetzt auf 2025-03-01 (nach SubmissionToReview und ReviewApproval), was der chronologischen Logik entspricht.

Überarbeitete SHACL-Datei

Da wir die ABox jetzt ohne explizite Beziehungen haben, müssen die SAs die Reihenfolge implizit aus den Phasen-Typen und Zeitstempeln ableiten. Ich passe die Regeln an, um ohne occp:after/occp:before zu arbeiten, und füge später die expliziten Regeln für diese OPs hinzu.

turtle

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

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix osh: <http://www.occpshape.de/shape#> .

### Allgemeine Struktur-Regeln

osh:GeneralTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:property [

sh:path time:hasBeginning ;

sh:class time:Instant ;

sh:minCount 1 ;

sh:message "At least one 'time:hasBeginning' must reference a 'time:Instant'."@en ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path time:hasEnd ;

sh:class time:Instant ;

sh:minCount 1 ;

sh:message "At least one 'time:hasEnd' must reference a 'time:Instant'."@en ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:hasActualBeginning ;

sh:class time:Instant ;

sh:maxCount 1 ;

sh:minCount 1 ;

sh:message "Each entity must have exactly one actual beginning via occp:hasActualBeginning."@en ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:hasActualEnd ;

sh:class time:Instant ;

sh:maxCount 1 ;

sh:minCount 1 ;

sh:message "Each entity must have exactly one actual end via occp:hasActualEnd."@en ;

sh:severity sh:Violation ;

] .

osh:InstantTimeShape a sh:NodeShape ;

sh:targetClass time:Instant ;

sh:property [

sh:path occp:hasActualTime ;

sh:datatype xsd:date ;

sh:maxCount 1 ;

sh:minCount 1 ;

sh:message "Each instant must have exactly one actual timestamp via occp:hasActualTime of type xsd:date."@en ;

sh:severity sh:Violation

] .

### Implizite Reihenfolge-Regel (ohne explizite OPs)

osh:PhaseOrderTimeShape a sh:NodeShape ;

sh:targetClass occp:Phase ;

sh:sparql [

sh:select """

SELECT $this ?other ?endTime ?startTime WHERE {

$this time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

?other time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

FILTER (?endTime >= ?startTime)

FILTER (?other != $this)

FILTER (

($this a occp:PhaseA\_Planning && ?other a occp:PhaseB\_Review) ||

($this a occp:PhaseA\_Planning && ?other a occp:PhaseC\_Construction) ||

($this a occp:PhaseB\_Review && ?other a occp:PhaseC\_Construction)

)

}

""" ;

sh:message "The end of { $this } (time: { ?endTime }) must be before the beginning of { ?other } (time: { ?startTime }) based on phase order."@en ;

sh:severity sh:Violation

] .

### Generische Regeln für explizite occp:after

osh:AfterTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

SELECT $this ?other ?startTime ?endTime WHERE {

$this occp:after ?other .

$this time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

?other time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

FILTER (?startTime <= ?endTime)

}

""" ;

sh:message "The beginning of { $this } (time: { ?startTime }) must be after the end of { ?other } (time: { ?endTime }) via 'occp:after'."@en ;

sh:severity sh:Violation

] .

### Generische Regeln für explizite occp:before

osh:BeforeTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

SELECT $this ?other ?endTime ?startTime WHERE {

$this occp:before ?other .

$this time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

?other time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

FILTER (?endTime >= ?startTime)

}

""" ;

sh:message "The end of { $this } (time: { ?endTime }) must be before the beginning of { ?other } (time: { ?startTime }) via 'occp:before'."@en ;

sh:severity sh:Violation

] .

Änderungen

* PREFIXes: Entfernt aus den SAs, da sie am Anfang der Datei definiert sind.
* time:hasBeginning/time:hasEnd: Verwendet statt nur occp:hasActualBeginning/hasActualEnd in den SAs, für Generizität. Die spezifischen hasActual\*-Regeln bleiben in osh:GeneralTimeShape, um maxCount 1 sicherzustellen.
* ABox: Die korrigierte Version oben ohne explizite Beziehungen wird mit osh:PhaseOrderTimeShape getestet. Die AfterTimeShape und BeforeTimeShape sind für Fälle gedacht, in denen explizite OPs verwendet werden (optional).

Test

* Mit der korrigierten ABox: Sollte Conforms: True ergeben, da:
  + PhaseA (Ende: 2025-03-01) vor PhaseC (Start: 2025-03-15).
  + PhaseB (Ende: 2025-02-28) vor PhaseC (Start: 2025-03-15).
  + PhaseA (Ende: 2025-03-01) nach PhaseB (Start: 2025-02-15) → OK, da keine umgekehrte Reihenfolge vorgeschrieben.

Nächste Schritte

1. Testen: Du führst die obige SHACL-Datei mit der korrigierten ABox durch und teilst mir das Log mit.
2. Erweitern: Wenn das funktioniert, fügen wir SAs für weitere OPs hinzu:
   * occp:endsBefore, occp:beginsBefore, etc.
   * Dann time:after, time:before, falls sie in deiner TBox relevant sind (ich prüfe das in OCCP\_V0.3.ttl).
3. Vollständige Ontologie: Sobald die generischen Regeln stabil sind, integrieren wir sie in die volle OCCP\_SHACL.ttl.

Ich schicke dir die Datei gerne als OCCP\_SHACL\_min.ttl, wenn du willst, oder du kannst die Änderungen selbst übernehmen. Wie möchtest du vorgehen? Und danke für deine scharfen Augen – das hält uns auf Kurs!

Vielen Dank für Deine Einschätzungen!!!! Ein Problem mit den super-generischen Angaben (time:hasBeginning anstelle von occp:hasEstimatedBeginning) könnte sich daraus ergeben, dass es mehrere Angaben für EstimatedBeginning gibt und die Abfrage dann mehrere Werte prüfen müsste - das müssten wir bei der Erstellung der Abfrage irgendwie berücksichtigen (z.B.: Wenn A vor B sein soll (gem. TBox), dann prüfe in der ABox, welcher der SPÄTESTE EstimatedBeginning von A ist und vergleiche ihn mit dem FRÜHESTEN EstimatedBeginning von B - und so weiter). Ist das umsetzbar?

Danke für den Hinweis – du hast absolut recht, und das ist ein wichtiger Punkt, den wir berücksichtigen müssen! Die Verwendung von generischen Properties wie time:hasBeginning bringt tatsächlich die Herausforderung mit sich, dass mehrere Instants (z. B. mehrere occp:hasEstimatedBeginning-Werte) existieren könnten, was die Zeitstempelprüfung komplexer macht. Dein Vorschlag, den spätesten Zeitpunkt von A mit dem frühesten Zeitpunkt von B zu vergleichen, ist eine elegante und logische Lösung, um sicherzustellen, dass die chronologische Reihenfolge auch bei mehreren Zeitangaben konsistent bleibt. Das ist definitiv umsetzbar – lass uns das durchdenken und implementieren!

Problemstellung

* time:hasBeginning und Mehrfachwerte: Wenn eine Phase mehrere time:hasBeginning-Werte hat (z. B. durch occp:hasActualBeginning und mehrere occp:hasEstimatedBeginning), müssen wir entscheiden, welche Zeitstempel verglichen werden. Ohne Einschränkung könnte die SPARQL-Abfrage mehrere Ergebnisse liefern, was zu uneindeutigen oder falschen Validierungen führen könnte.
* Logik:
  + Für occp:before (A vor B): Das Ende von A muss vor dem frühesten Anfang von B liegen → wir brauchen den spätesten time:hasEnd von A und den frühesten time:hasBeginning von B.
  + Für occp:after (A nach B): Der Anfang von A muss nach dem spätesten Ende von B liegen → wir brauchen den frühesten time:hasBeginning von A und den spätesten time:hasEnd von B.
* Umsetzbarkeit: SPARQL bietet mit Aggregatfunktionen wie MAX() und MIN() die Möglichkeit, den spätesten bzw. frühesten Zeitstempel zu ermitteln. Das können wir nutzen!

Lösung: Generische SAs mit Aggregaten

Ich passe die SAs an, um diese Logik zu implementieren. Hier ist die überarbeitete Version mit MAX() und MIN() für occp:before und occp:after:

turtle

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

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix osh: <http://www.occpshape.de/shape#> .

### Allgemeine Struktur-Regeln

osh:GeneralTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:property [

sh:path time:hasBeginning ;

sh:class time:Instant ;

sh:minCount 1 ;

sh:message "At least one 'time:hasBeginning' must reference a 'time:Instant'."@en ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path time:hasEnd ;

sh:class time:Instant ;

sh:minCount 1 ;

sh:message "At least one 'time:hasEnd' must reference a 'time:Instant'."@en ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:hasActualBeginning ;

sh:class time:Instant ;

sh:maxCount 1 ;

sh:minCount 1 ;

sh:message "Each entity must have exactly one actual beginning via occp:hasActualBeginning."@en ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:hasActualEnd ;

sh:class time:Instant ;

sh:maxCount 1 ;

sh:minCount 1 ;

sh:message "Each entity must have exactly one actual end via occp:hasActualEnd."@en ;

sh:severity sh:Violation ;

] .

osh:InstantTimeShape a sh:NodeShape ;

sh:targetClass time:Instant ;

sh:property [

sh:path occp:hasActualTime ;

sh:datatype xsd:date ;

sh:maxCount 1 ;

sh:minCount 1 ;

sh:message "Each instant must have exactly one actual timestamp via occp:hasActualTime of type xsd:date."@en ;

sh:severity sh:Violation

] ;

sh:property [

sh:path occp:hasEstimatedTime ;

sh:datatype xsd:date ;

sh:message "Each estimated timestamp via occp:hasEstimatedTime must be of type xsd:date."@en ;

sh:severity sh:Violation

] .

### Generische Regel für occp:before mit Aggregaten

osh:BeforeTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

SELECT $this ?other ?latestEndTime ?earliestStartTime WHERE {

$this occp:before ?other .

{

SELECT $this (MAX(?endTime) AS ?latestEndTime)

WHERE {

$this time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY $this

}

{

SELECT ?other (MIN(?startTime) AS ?earliestStartTime)

WHERE {

?other time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY ?other

}

FILTER (?latestEndTime >= ?earliestStartTime)

}

""" ;

sh:message "The latest end of { $this } (time: { ?latestEndTime }) must be before the earliest beginning of { ?other } (time: { ?earliestStartTime }) via 'occp:before'."@en ;

sh:severity sh:Violation

] .

### Generische Regel für occp:after mit Aggregaten

osh:AfterTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

SELECT $this ?other ?earliestStartTime ?latestEndTime WHERE {

$this occp:after ?other .

{

SELECT $this (MIN(?startTime) AS ?earliestStartTime)

WHERE {

$this time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY $this

}

{

SELECT ?other (MAX(?endTime) AS ?latestEndTime)

WHERE {

?other time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY ?other

}

FILTER (?earliestStartTime <= ?latestEndTime)

}

""" ;

sh:message "The earliest beginning of { $this } (time: { ?earliestStartTime }) must be after the latest end of { ?other } (time: { ?latestEndTime }) via 'occp:after'."@en ;

sh:severity sh:Violation

] .

### Implizite Reihenfolge-Regel (falls keine expliziten OPs vorhanden)

osh:PhaseOrderTimeShape a sh:NodeShape ;

sh:targetClass occp:Phase ;

sh:sparql [

sh:select """

SELECT $this ?other ?latestEndTime ?earliestStartTime WHERE {

{

SELECT $this (MAX(?endTime) AS ?latestEndTime)

WHERE {

$this time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY $this

}

{

SELECT ?other (MIN(?startTime) AS ?earliestStartTime)

WHERE {

?other time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY ?other

}

FILTER (?latestEndTime >= ?earliestStartTime)

FILTER (?other != $this)

FILTER (

($this a occp:PhaseA\_Planning && ?other a occp:PhaseB\_Review) ||

($this a occp:PhaseA\_Planning && ?other a occp:PhaseC\_Construction) ||

($this a occp:PhaseB\_Review && ?other a occp:PhaseC\_Construction)

)

}

""" ;

sh:message "The latest end of { $this } (time: { ?latestEndTime }) must be before the earliest beginning of { ?other } (time: { ?earliestStartTime }) based on phase order."@en ;

sh:severity sh:Violation

] .

Erklärung

* Aggregatfunktionen:
  + MAX(?endTime): Ermittelt den spätesten Endzeitpunkt einer Phase (falls mehrere time:hasEnd existieren).
  + MIN(?startTime): Ermittelt den frühesten Startzeitpunkt einer Phase (falls mehrere time:hasBeginning existieren).
* osh:BeforeTimeShape:
  + Prüft, ob der späteste time:hasEnd von $this vor dem frühesten time:hasBeginning von ?other liegt.
* osh:AfterTimeShape:
  + Prüft, ob der früheste time:hasBeginning von $this nach dem spätesten time:hasEnd von ?other liegt.
* osh:PhaseOrderTimeShape:
  + Gleiche Logik, aber für implizite Reihenfolgen ohne explizite OPs, basierend auf Phasen-Typen.
* occp:hasEstimatedTime:
  + Ich habe es in osh:InstantTimeShape ergänzt, um sicherzustellen, dass geschätzte Zeitstempel ebenfalls xsd:date sind. Wir könnten später eine Regel hinzufügen, die zwischen hasActualTime und hasEstimatedTime priorisiert, falls nötig.

Test-ABox mit mehreren Zeitstempeln

Um das zu testen, fügen wir mehrere hasEstimatedBeginning-Werte hinzu:

turtle

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix : <http://www.example.de/example#> .

:phaseA a occp:PhaseA\_Planning ;

occp:hasActualBeginning :instant\_beginning\_of\_planning ;

occp:hasActualEnd :instant\_completion\_of\_planning .

:phaseB a occp:PhaseB\_Review ;

occp:hasActualBeginning :instant\_submission\_to\_review ;

occp:hasActualEnd :instant\_review\_approval ;

time:hasBeginning :instant\_estimated\_start\_b1 ; # Zusätzlicher geschätzter Start

time:hasBeginning :instant\_estimated\_start\_b2 .

:phaseC a occp:PhaseC\_Construction ;

occp:hasActualBeginning :instant\_construction\_start ;

occp:hasActualEnd :instant\_completion\_of\_construction .

:instant\_beginning\_of\_planning a occp:BeginningOfPlanning ;

occp:hasActualTime "2025-01-01"^^xsd:date .

:instant\_completion\_of\_planning a occp:CompletionOfPlanning ;

occp:hasActualTime "2025-03-01"^^xsd:date .

:instant\_submission\_to\_review a occp:SubmissionToReview ;

occp:hasActualTime "2025-02-15"^^xsd:date .

:instant\_estimated\_start\_b1 a time:Instant ;

occp:hasEstimatedTime "2025-02-10"^^xsd:date . # Früherer geschätzter Start

:instant\_estimated\_start\_b2 a time:Instant ;

occp:hasEstimatedTime "2025-02-20"^^xsd:date . # Späterer geschätzter Start

:instant\_review\_approval a occp:ReviewApproval ;

occp:hasActualTime "2025-02-28"^^xsd:date .

:instant\_construction\_start a occp:ConstructionStart ;

occp:hasActualTime "2025-03-15"^^xsd:date .

:instant\_completion\_of\_construction a occp:CompletionOfConstruction ;

occp:hasActualTime "2025-06-01"^^xsd:date .

* Erwartung mit osh:PhaseOrderTimeShape:
  + Spätestes Ende von PhaseA (2025-03-01) vs. frühestes Anfang von PhaseB (2025-02-10) → Fehler, da 2025-03-01 > 2025-02-10.
  + Spätestes Ende von PhaseB (2025-02-28) vs. frühestes Anfang von PhaseC (2025-03-15) → OK, da 2025-02-28 < 2025-03-15.

Umsetzbarkeit

Ja, das ist umsetzbar! Die Aggregatfunktionen lösen das Problem der Mehrfachwerte:

* Spätestes Ende: MAX(?endTime) stellt sicher, dass selbst bei mehreren Endpunkten der letzte relevant ist.
* Frühestes Anfang: MIN(?startTime) stellt sicher, dass der erste Startpunkt verglichen wird.
* Flexibilität: Funktioniert mit hasActualTime und hasEstimatedTime, da beide unter time:hasBeginning/time:hasEnd fallen.

Nächste Schritte

1. Testen: Du kannst die obige SHACL-Datei mit der Test-ABox ausprobieren. Das Log sollte den Fehler für PhaseA vor PhaseB zeigen (wegen 2025-03-01 > 2025-02-10).
2. Korrektur testen: Passe einen Zeitstempel an (z. B. instant\_completion\_of\_planning auf "2025-02-05"^^xsd:date) und prüfe, ob Conforms: True wird.
3. Erweitern: Wenn das stabil ist, fügen wir SAs für occp:endsBefore, occp:beginsBefore, etc. hinzu.

Soll ich dir die Datei als OCCP\_SHACL\_min.ttl schicken, oder übernimmst du die Änderungen? Wie siehst du das mit der Priorisierung von hasActualTime vs. hasEstimatedTime – sollen wir das später separat regeln? Lass mich wissen!

Vielen Dank für Deine Einschätzungen!!!! Ein Problem mit den super-generischen Angaben (time:hasBeginning anstelle von occp:hasEstimatedBeginning) könnte sich daraus ergeben, dass es mehrere Angaben für EstimatedBeginning gibt und die Abfrage dann mehrere Werte prüfen müsste - das müssten wir bei der Erstellung der Abfrage irgendwie berücksichtigen (z.B.: Wenn A vor B sein soll (gem. TBox), dann prüfe in der ABox, welcher der SPÄTESTE EstimatedBeginning von A ist und vergleiche ihn mit dem FRÜHESTEN EstimatedBeginning von B - und so weiter). Ist das umsetzbar?Frage zu Deinem SA-Vorschlag: Sollte nicht auch time:hasTime anstelle von occp:hasActualTime verwendet werden?Die General Shacl Rules (osh:GeneralDatePropertyShape, osh:GeneralTimeShape und osh:InstantTimeShape) würde ich gern in der ursprünglichen Form beibehalten, oder spricht da etwas dagegen?

Danke für deine Rückmeldung! Ich gehe gerne auf deine Fragen und Anmerkungen ein – wir sind auf einem guten Weg, und deine Punkte helfen, das Ganze noch präziser zu machen. Lass uns das Schritt für Schritt durchgehen!

1. Mehrere EstimatedBeginnings und Aggregatfunktionen

Du hattest schon erwähnt, dass mehrere occp:hasEstimatedBeginning-Werte ein Problem darstellen könnten, und ich habe das mit MAX() und MIN() adressiert. Dein Beispiel (spätester EstimatedBeginning von A vs. frühester EstimatedBeginning von B) ist genau das, was wir mit den Aggregatfunktionen umgesetzt haben:

* Spätester EstimatedBeginning von A: Wird durch MAX(?startTime) abgedeckt, wenn wir time:hasBeginning verwenden (das sowohl hasActualBeginning als auch hasEstimatedBeginning umfasst).
* Frühester EstimatedBeginning von B: Wird durch MIN(?startTime) abgedeckt.
* Umsetzbarkeit: Ja, das ist umsetzbar, und meine vorherige Antwort hat das schon implementiert! Die SPARQL-Abfragen mit MAX() und MIN() lösen dieses Problem, indem sie die extremsten Werte auswählen. Ich werde das weiter unten nochmal anpassen, um sicherzustellen, dass es klar ist.

2. time:hasTime anstelle von occp:hasActualTime?

* Deine Frage: Sollten wir in den SAs time:hasTime statt occp:hasActualTime verwenden, um generischer zu bleiben?
* Analyse:
  + time:hasTime: Ist eine generische Property aus der W3C TIME-Ontologie, die einen Zeitstempel (z. B. xsd:dateTime) oder eine komplexere Zeitangabe (z. B. time:TemporalEntity) mit einem Instant verknüpfen kann. Sie ist nicht spezifisch auf xsd:date beschränkt und wird in der TIME-Ontologie oft für allgemeine Zeitbezüge verwendet.
  + occp:hasActualTime: Ist in deiner Ontologie spezifisch definiert und auf xsd:date beschränkt (siehe osh:GeneralDatePropertyShape), was es präziser macht, aber weniger flexibel.
  + Vorteile von time:hasTime:
    - Generischer: Funktioniert mit jedem Zeitstempel, unabhängig davon, ob er hasActualTime, hasEstimatedTime oder eine andere Subproperty ist.
    - Konsistenter mit der TIME-Ontologie, falls du sie erweitern möchtest (z. B. mit time:GeneralDateDescription).
  + Nachteile:
    - Weniger Kontrolle: time:hasTime erzwingt nicht xsd:date, was zu Inkonsistenzen führen könnte, wenn andere Formate auftauchen (z. B. xsd:dateTime).
    - Deine bestehenden Regeln (z. B. osh:InstantTimeShape) verwenden occp:hasActualTime und occp:hasEstimatedTime explizit mit sh:datatype xsd:date.
  + Meine Empfehlung:
    - Für die SAs sollten wir bei occp:hasActualTime und occp:hasEstimatedTime bleiben, da deine Ontologie diese spezifischen Properties verwendet und sie durch osh:InstantTimeShape auf xsd:date beschränkt sind. Das hält die Validierung konsistent mit deinen bisherigen Definitionen.
    - Wenn wir time:hasTime einführen würden, müssten wir entweder die Struktur-Regeln anpassen (z. B. osh:InstantTimeShape auf time:hasTime umstellen) oder eine zusätzliche Regel schreiben, um sicherzustellen, dass nur xsd:date erlaubt ist. Das wäre ein größerer Umbau, der nicht unbedingt nötig ist, solange deine Instants nur hasActualTime/hasEstimatedTime verwenden.
  + Kompromiss: Wir könnten beide einbauen und priorisieren (z. B. occp:hasActualTime bevorzugt, sonst occp:hasEstimatedTime), aber ich denke, das können wir später machen, wenn es nötig wird.

3. Ursprüngliche General SHACL Rules beibehalten

* Dein Wunsch: Die ursprünglichen osh:GeneralDatePropertyShape, osh:GeneralTimeShape und osh:InstantTimeShape beibehalten.
* Meine Einschätzung: Das ist völlig in Ordnung und spricht aus mehreren Gründen nicht dagegen:
  + Konsistenz: Die ursprünglichen Regeln sind spezifisch auf deine Ontologie zugeschnitten (z. B. occp:hasActualTime und occp:hasEstimatedTime mit xsd:date, maxCount 1 für hasActualBeginning/hasActualEnd).
  + Klarheit: Sie definieren genau, wie deine Instants und TemporalEntities strukturiert sein sollen, was die Basis für die SAs bildet.
  + Keine Redundanz: Meine Änderungen (z. B. time:hasBeginning statt occp:hasActualBeginning) waren nur für die SAs gedacht, um Generizität zu erreichen, aber die Struktur-Regeln können spezifisch bleiben.
  + Fazit: Wir behalten die ursprünglichen Regeln bei und passen nur die SAs an, um mit time:hasBeginning/time:hasEnd zu arbeiten, während die Struktur-Regeln weiterhin occp:-Properties erzwingen.

Überarbeitete SHACL-Datei

Ich stelle die ursprünglichen General-Regeln wieder her und passe die SAs an, um deine Punkte zu berücksichtigen:

turtle

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

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix osh: <http://www.occpshape.de/shape#> .

### Ursprüngliche General SHACL Rules

osh:GeneralDatePropertyShape a sh:PropertyShape ;

sh:path ( occp:hasActualTime occp:hasEstimatedTime ) ;

sh:datatype xsd:date ;

sh:message "Dates must be in YYYY-MM-DD format."@en ;

sh:severity sh:Violation .

osh:GeneralTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:property [

sh:path ( time:hasBeginning time:hasEnd ) ;

sh:class time:Instant ;

sh:minCount 1 ;

sh:message "At least one 'time:hasBeginning' or 'time:hasEnd' must reference a 'time:Instant'."@en ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:hasActualBeginning ;

sh:class time:Instant ;

sh:maxCount 1 ;

sh:message "Each entity must have at most one actual beginning via occp:hasActualBeginning."@en ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:hasActualEnd ;

sh:class time:Instant ;

sh:maxCount 1 ;

sh:message "Each entity must have at most one actual end via occp:hasActualEnd."@en ;

sh:severity sh:Violation ;

] .

osh:InstantTimeShape a sh:NodeShape ;

sh:targetClass time:Instant ;

sh:property [

sh:path occp:hasActualTime ;

sh:datatype xsd:date ;

sh:maxCount 1 ;

sh:message "Each instant must have at most one actual timestamp via occp:hasActualTime of type xsd:date."@en ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:hasEstimatedTime ;

sh:datatype xsd:date ;

sh:message "Each instant must have estimated timestamps via occp:hasEstimatedTime of type xsd:date."@en ;

sh:severity sh:Violation ;

] .

### Generische Regel für occp:before mit Aggregaten

osh:BeforeTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

SELECT $this ?other ?latestEndTime ?earliestStartTime WHERE {

$this occp:before ?other .

{

SELECT $this (MAX(?endTime) AS ?latestEndTime)

WHERE {

$this time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY $this

}

{

SELECT ?other (MIN(?startTime) AS ?earliestStartTime)

WHERE {

?other time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY ?other

}

FILTER (?latestEndTime >= ?earliestStartTime)

}

""" ;

sh:message "The latest end of { $this } (time: { ?latestEndTime }) must be before the earliest beginning of { ?other } (time: { ?earliestStartTime }) via 'occp:before'."@en ;

sh:severity sh:Violation

] .

### Generische Regel für occp:after mit Aggregaten

osh:AfterTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

SELECT $this ?other ?earliestStartTime ?latestEndTime WHERE {

$this occp:after ?other .

{

SELECT $this (MIN(?startTime) AS ?earliestStartTime)

WHERE {

$this time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY $this

}

{

SELECT ?other (MAX(?endTime) AS ?latestEndTime)

WHERE {

?other time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY ?other

}

FILTER (?earliestStartTime <= ?latestEndTime)

}

""" ;

sh:message "The earliest beginning of { $this } (time: { ?earliestStartTime }) must be after the latest end of { ?other } (time: { ?latestEndTime }) via 'occp:after'."@en ;

sh:severity sh:Violation

] .

### Implizite Reihenfolge-Regel

osh:PhaseOrderTimeShape a sh:NodeShape ;

sh:targetClass occp:Phase ;

sh:sparql [

sh:select """

SELECT $this ?other ?latestEndTime ?earliestStartTime WHERE {

{

SELECT $this (MAX(?endTime) AS ?latestEndTime)

WHERE {

$this time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY $this

}

{

SELECT ?other (MIN(?startTime) AS ?earliestStartTime)

WHERE {

?other time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY ?other

}

FILTER (?latestEndTime >= ?earliestStartTime)

FILTER (?other != $this)

FILTER (

($this a occp:PhaseA\_Planning && ?other a occp:PhaseB\_Review) ||

($this a occp:PhaseA\_Planning && ?other a occp:PhaseC\_Construction) ||

($this a occp:PhaseB\_Review && ?other a occp:PhaseC\_Construction)

)

}

""" ;

sh:message "The latest end of { $this } (time: { ?latestEndTime }) must be before the earliest beginning of { ?other } (time: { ?earliestStartTime }) based on phase order."@en ;

sh:severity sh:Violation

] .

Änderungen

* General-Regeln: Wieder in der ursprünglichen Form, mit maxCount statt minCount für hasActualBeginning/hasActualEnd, wie in deiner OCCP\_SHACL.ttl.
* SAs: Verwenden time:hasBeginning/time:hasEnd mit Aggregaten, aber bleiben bei occp:hasActualTime für die Zeitstempel, um mit deinen Struktur-Regeln konsistent zu sein. hasEstimatedTime wird in den Testdaten ergänzt, aber nicht explizit in den SAs abgefragt (da es unter time:hasBeginning fällt).

Test-ABox mit EstimatedBeginnings

turtle

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix : <http://www.example.de/example#> .

:phaseA a occp:PhaseA\_Planning ;

occp:hasActualBeginning :instant\_beginning\_of\_planning ;

occp:hasActualEnd :instant\_completion\_of\_planning .

:phaseB a occp:PhaseB\_Review ;

occp:hasActualBeginning :instant\_submission\_to\_review ;

occp:hasActualEnd :instant\_review\_approval ;

time:hasBeginning :instant\_estimated\_start\_b1 ; # Zusätzlicher geschätzter Start

time:hasBeginning :instant\_estimated\_start\_b2 .

:phaseC a occp:PhaseC\_Construction ;

occp:hasActualBeginning :instant\_construction\_start ;

occp:hasActualEnd :instant\_completion\_of\_construction .

:instant\_beginning\_of\_planning a occp:BeginningOfPlanning ;

occp:hasActualTime "2025-01-01"^^xsd:date .

:instant\_completion\_of\_planning a occp:CompletionOfPlanning ;

occp:hasActualTime "2025-03-01"^^xsd:date .

:instant\_submission\_to\_review a occp:SubmissionToReview ;

occp:hasActualTime "2025-02-15"^^xsd:date .

:instant\_estimated\_start\_b1 a time:Instant ;

occp:hasEstimatedTime "2025-02-10"^^xsd:date . # Früherer geschätzter Start

:instant\_estimated\_start\_b2 a time:Instant ;

occp:hasEstimatedTime "2025-02-20"^^xsd:date . # Späterer geschätzter Start

:instant\_review\_approval a occp:ReviewApproval ;

occp:hasActualTime "2025-02-28"^^xsd:date .

:instant\_construction\_start a occp:ConstructionStart ;

occp:hasActualTime "2025-03-15"^^xsd:date .

:instant\_completion\_of\_construction a occp:CompletionOfConstruction ;

occp:hasActualTime "2025-06-01"^^xsd:date .

* Erwartung:
  + PhaseA (spätestes Ende: 2025-03-01) vs. PhaseB (frühestes Anfang: 2025-02-10) → Fehler, da 2025-03-01 > 2025-02-10.
  + PhaseB (spätestes Ende: 2025-02-28) vs. PhaseC (frühestes Anfang: 2025-03-15) → OK.

Nächste Schritte

1. Testen: Führe die obige SHACL-Datei mit der Test-ABox aus und teile mir das Log mit.
2. Anpassen: Wenn der Fehler wie erwartet auftritt, korrigiere den Zeitstempel (z. B. instant\_completion\_of\_planning auf "2025-02-05"^^xsd:date) und prüfe, ob Conforms: True wird.
3. Erweitern: Danach können wir SAs für occp:endsBefore, occp:beginsBefore, etc. hinzufügen.

Ich kann dir die OCCP\_SHACL\_min.ttl schicken, wenn du willst, oder du passt sie selbst an. Wie möchtest du vorgehen? Und was denkst du über occp:hasActualTime vs. time:hasTime – sollen wir das später nochmal überdenken? Freue mich auf dein Feedback!

Break. Wir arbeiten jetzt mit Apache Jena, weil PySHACL Probleme mit der Interpretation von SPARQL hat. Danke für Deine Hilfe! Ich habe um Zeilen zu sparen wieder an dieser Stelle angesetzt. Die neue validation\_shacl.py sieht jetzt so aus:import owlready2 from rdflib import Graph, Namespace, RDF from rdflib.namespace import SH, OWL import logging import os import sys import subprocess import json import shutil from io import StringIOKonfiguration des LoggingsBASE\_DIR = os.path.dirname(os.path.abspath(file)) logging.basicConfig( filename=os.path.join(BASE\_DIR, "validation.log"), level=logging.DEBUG, format="%(asctime)s - %(levelname)s - %(message)s", filemode="w" )Logging-Setup (falls nicht schon vorhanden)logging.basicConfig(level=logging.INFO) logger = logging.getLogger(name)Pfade und NamespaceTBOX\_PATH = os.path.join(BASE\_DIR, "OCCP\_V0.3.ttl") ABOX\_DIR = os.path.join(BASE\_DIR, "OCCP\_ABox") SHAPES\_PATH = os.path.join(BASE\_DIR, "OCCP\_SHACL\_min.ttl") JAVA\_EXE = r"G:\Java\JDK\_23\bin\java.exe".replace("\", "/") JENA\_HOME = os.path.join(BASE\_DIR, "apache-jena-5.3.0") # Hauptverzeichnis der Jena-Installation OULD = Namespace("[http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#](http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD)") OCCP = Namespace("[http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#](http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod_V0.1)")def combine\_and\_reason(tbox\_path=TBOX\_PATH, abox\_path=None, java\_exe=JAVA\_EXE): try: tbox\_path\_normalized = tbox\_path.replace("\", "/") abox\_path\_normalized = abox\_path.replace("\", "/") if abox\_path else None onto = owlready2.get\_ontology(f"file://{tbox\_path\_normalized}").load(format="turtle") if abox\_path: abox\_onto = owlready2.get\_ontology(f"file://{abox\_path\_normalized}").load(format="turtle") with onto: for indiv in abox\_onto.individuals(): new\_indiv = onto.get\_entities(indiv.name, indiv.class) if not new\_indiv: new\_indiv = indiv.class(indiv.name, namespace=onto) for prop in indiv.get\_properties(): for value in prop[indiv]: prop[new\_indiv] = value owlready2.sync\_reasoner\_pellet(infer\_property\_values=True, infer\_data\_property\_values=True, debug=2) data\_graph = Graph() data\_graph.parse(tbox\_path, format="turtle") if abox\_path: data\_graph.parse(abox\_path, format="turtle") output\_file = os.path.join(BASE\_DIR, "inferred\_ontology.ttl") data\_graph.serialize(destination=output\_file, format="turtle") logger.info(f"Inferierte Ontologie gespeichert: {output\_file}") # Generische Disjunktheitsprüfung logger.info("Prüfe Ontologie auf Disjunktheit...") disjoint\_pairs = set() for s, p, o in data\_graph.triples((None, OWL.disjointWith, None)): disjoint\_pairs.add((s, o)) disjoint\_pairs.add((o, s)) # Bidirektional logger.debug(f"Disjunkte Klassenpaare: {disjoint\_pairs}") for subj in data\_graph.subjects(RDF.type, None): types = set(o for s, p, o in data\_graph.triples((subj, RDF.type, None))) for class1, class2 in disjoint\_pairs: if class1 in types and class2 in types: logger.error(f"Disjunktheitsverletzung gefunden: {subj} hat Typen {class1} und {class2}") raise Exception(f"Ontology is inconsistent: {subj} has disjoint types {class1} and {class2}") logger.info("Keine Disjunktheitsverletzungen gefunden.") return output\_file except Exception as e: logger.error(f"Fehler beim Reasoning oder Disjunktheitsprüfung: {e}") raisedef debug\_sparql(data\_file): try: data\_graph = Graph().parse(data\_file, format="turtle") query = """ PREFIX ould: [http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#](http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD) SELECT ?chain (COUNT(?u) AS ?updateCount) WHERE { ?chain a ould:UpdateChain . ?chain ould:hasUpdate ?u . } GROUP BY ?chain """ logger.info("Starte SPARQL-Abfrage für alle UpdateChains...") results = data\_graph.query(query) logger.info("SPARQL-Abfrage Ergebnisse:") for row in results: logger.info(f"Chain: {row.chain}, UpdateCount: {row.updateCount}") return len(results) > 0 except Exception as e: logger.error(f"Fehler bei der SPARQL-Abfrage: {e}") raisedef perform\_shacl\_jena\_validation(data\_file, shapes\_path=SHAPES\_PATH): try: # Suche shacl im PATH jena\_shacl\_cmd = shutil.which("shacl") if not jena\_shacl\_cmd: # Fallback: Versuche den direkten Pfad jena\_shacl\_cmd = os.path.join(JENA\_HOME, "bin", "shacl.bat") if os.name == 'nt' else os.path.join(JENA\_HOME, "bin", "shacl") if not os.path.exists(jena\_shacl\_cmd): logger.error(f"Jena SHACL-Tool nicht gefunden: {jena\_shacl\_cmd}") return False

# Konvertiere Pfade für Jena (Backslashes zu Schrägstrichen)

data\_file\_jena = data\_file.replace("\\", "/")

shapes\_path\_jena = shapes\_path.replace("\\", "/")

# Schreibe die Ausgabe in eine temporäre Datei

report\_file = os.path.join(BASE\_DIR, "validation\_report.ttl")

cmd = [

jena\_shacl\_cmd,

"validate",

"--data", data\_file\_jena,

"--shapes", shapes\_path\_jena

]

with open(report\_file, "w", encoding="utf-8") as f:

result = subprocess.run(cmd, stdout=f, stderr=subprocess.PIPE, text=True, env={\*\*os.environ, "JENA\_HOME": JENA\_HOME})

# Logge stderr

if result.stderr:

logger.error(f"Jena SHACL validation stderr: {result.stderr}")

if result.returncode == 0:

# Lese die Ausgabe aus der Datei

with open(report\_file, "r", encoding="utf-8") as f:

report\_data = f.read()

logger.info(f"Jena SHACL validation stdout: {report\_data}")

# Parse die Turtle-Ausgabe mit rdflib

report\_graph = Graph()

report\_graph.parse(data=report\_data, format="turtle")

conforms = False

for s, p, o in report\_graph.triples((None, SH.conforms, None)):

conforms = o.toPython() # o sollte ein Literal mit "true" oder "false" sein

logger.info(f"Konformität (inference=none): {conforms}")

if not conforms:

# Extrahiere Ergebnisse für detaillierte Fehler

for s, p, o in report\_graph.triples((None, SH.result, None)):

for result\_obj in report\_graph.objects(s, SH.result):

message = report\_graph.value(result\_obj, SH.message) or "No message"

focus\_node = report\_graph.value(result\_obj, SH.focusNode) or "Unknown"

path = report\_graph.value(result\_obj, SH.path) or "Unknown"

severity = report\_graph.value(result\_obj, SH.severity) or "Unknown"

logger.error(f"Validation error: {message} (Focus Node: {focus\_node}, Path: {path}, Severity: {severity})")

return conforms

else:

logger.error("Jena SHACL validation failed with non-zero exit code.")

return False

except Exception as e:

logger.error(f"Fehler bei der Jena SHACL-Validierung: {e}")

return False

if name == "main": ABOX\_PATH = os.path.join(ABOX\_DIR, "OCCP\_Valid\_LCycle\_1.ttl") inferred\_file = combine\_and\_reason(tbox\_path=TBOX\_PATH, abox\_path=ABOX\_PATH, java\_exe=JAVA\_EXE) debug\_sparql(inferred\_file) conforms = perform\_shacl\_jena\_validation(inferred\_file)

if conforms:

logger.info("Validation successful: Conforms to SHACL.")

else:

logger.error("Validation failed.")

Das letzte Validation Log sieht so aus:2025-03-12 23:58:34,918 - INFO - Inferierte Ontologie gespeichert: G:\Promo\cModeler\_Git\cMM\OCCP\_Dev2\inferred\_ontology.ttl 2025-03-12 23:58:34,918 - INFO - Pr�fe Ontologie auf Disjunktheit... 2025-03-12 23:58:34,918 - DEBUG - Disjunkte Klassenpaare: set() 2025-03-12 23:58:34,919 - INFO - Keine Disjunktheitsverletzungen gefunden. 2025-03-12 23:58:34,954 - INFO - Starte SPARQL-Abfrage f�r alle UpdateChains... 2025-03-12 23:58:35,069 - INFO - SPARQL-Abfrage Ergebnisse: 2025-03-12 23:58:36,141 - INFO - Jena SHACL validation stdout: PREFIX rdf: [http://www.w3.org/1999/02/22-rdf-syntax-ns#](http://www.w3.org/1999/02/22-rdf-syntax-ns) PREFIX rdfs: [http://www.w3.org/2000/01/rdf-schema#](http://www.w3.org/2000/01/rdf-schema) PREFIX sh: [http://www.w3.org/ns/shacl#](http://www.w3.org/ns/shacl) PREFIX xsd: [http://www.w3.org/2001/XMLSchema#](http://www.w3.org/2001/XMLSchema)[ rdf:type sh:ValidationReport; sh:conforms true ] .2025-03-12 23:58:36,142 - INFO - Konformit�t (inference=none): True 2025-03-12 23:58:36,142 - INFO - Validation successful: Conforms to SHACL.

Super, das sieht großartig aus – Conforms: True mit Apache Jena! Es ist eine kluge Entscheidung, auf Jena umzusteigen, wenn PySHACL Schwierigkeiten mit der SPARQL-Interpretation hatte. Jena ist robust und weit verbreitet für SHACL-Validierung, und dein neues Skript sieht solide aus. Danke, dass du mich auf den aktuellen Stand gebracht hast! Lass uns das durchgehen und die nächsten Schritte planen.

Analyse des neuen Setups

1. Skript (validation\_shacl.py):
   * Funktion combine\_and\_reason: Bleibt unverändert und nutzt Owlready2 mit Pellet für Reasoning, gefolgt von einer Disjunktheitsprüfung. Das funktioniert gut und liefert die inferierte Ontologie (inferred\_ontology.ttl).
   * Funktion debug\_sparql: Testet eine SPARQL-Abfrage für ould:UpdateChain, scheint aber keine Ergebnisse zu liefern (keine Chains in der ABox?), was in deinem Fall nicht kritisch ist.
   * Funktion perform\_shacl\_jena\_validation: Ruft das Jena SHACL-Tool über die Kommandozeile auf, speichert den Bericht in validation\_report.ttl und parst ihn mit rdflib, um sh:conforms zu extrahieren. Das ist eine saubere Lösung!
   * Logging: Gut strukturiert, mit detaillierten Fehlermeldungen bei Nicht-Konformität.
2. Log:
   * Reasoning und Disjunktheitsprüfung erfolgreich.
   * Jena SHACL-Validierung meldet sh:conforms true → die Kombination aus inferred\_ontology.ttl (TBox + ABox) und OCCP\_SHACL\_min.ttl ist konform.
   * Keine Fehler im Report → die aktuellen SHACL-Regeln und die ABox passen zusammen.
3. Warum Jena?:
   * PySHACL hatte offenbar Probleme mit der SPARQL-Interpretation (vielleicht mit Aggregatfunktionen wie MAX()/MIN() oder komplexen FILTERs). Jena ist hier zuverlässiger, da es eine vollständige SPARQL-Engine integriert hat und SHACL nativ unterstützt.

Aktueller Stand

* Du hast OCCP\_Valid\_LCycle\_1.ttl mit OCCP\_SHACL\_min.ttl getestet (vermutlich die Version mit den generischen SAs für occp:before, occp:after, und der impliziten osh:PhaseOrderTimeShape).
* Das Ergebnis ist Conforms: True, was bedeutet, dass:
  + Die Zeitstempel in der ABox die erwartete Reihenfolge einhalten.
  + Die SHACL-Regeln (inklusive SPARQL) korrekt ausgewertet werden.

Deine letzte ABox und SHACL

* ABox: OCCP\_Valid\_LCycle\_1.ttl enthält viele Phasen (A bis M) und Instants mit Zeitstempeln, teilweise mit expliziten Beziehungen wie occp:before/occp:after. Du hattest erwähnt, dass wir diese Beziehungen weglassen wollen – hast du sie für den Test entfernt?
* SHACL: OCCP\_SHACL\_min.ttl enthält vermutlich die generischen Regeln mit Aggregaten (MAX()/MIN()), die wir zuletzt besprochen haben. Kannst du bestätigen, welche Version du genau verwendet hast?

Nächste Schritte

Da wir jetzt ein funktionierendes Setup mit Jena haben, können wir weiter optimieren und erweitern:

1. ABox bereinigen

Wenn du die expliziten Beziehungen (occp:before, occp:after, etc.) aus OCCP\_Valid\_LCycle\_1.ttl entfernen möchtest, könnten wir eine minimalistische Version testen:

turtle

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix : <http://www.example.de/example#> .

:phaseA a occp:PhaseA\_Planning ;

occp:hasActualBeginning :instant\_beginning\_of\_planning ;

occp:hasActualEnd :instant\_completion\_of\_planning .

:phaseB a occp:PhaseB\_Review ;

occp:hasActualBeginning :instant\_submission\_to\_review ;

occp:hasActualEnd :instant\_review\_approval .

:phaseC a occp:PhaseC\_Construction ;

occp:hasActualBeginning :instant\_construction\_start ;

occp:hasActualEnd :instant\_completion\_of\_construction .

:instant\_beginning\_of\_planning a occp:BeginningOfPlanning ;

occp:hasActualTime "2025-01-01"^^xsd:date .

:instant\_completion\_of\_planning a occp:CompletionOfPlanning ;

occp:hasActualTime "2025-03-01"^^xsd:date .

:instant\_submission\_to\_review a occp:SubmissionToReview ;

occp:hasActualTime "2025-02-15"^^xsd:date .

:instant\_review\_approval a occp:ReviewApproval ;

occp:hasActualTime "2025-02-28"^^xsd:date .

:instant\_construction\_start a occp:ConstructionStart ;

occp:hasActualTime "2025-03-15"^^xsd:date .

:instant\_completion\_of\_construction a occp:CompletionOfConstruction ;

occp:hasActualTime "2025-06-01"^^xsd:date .

* Test: Prüfe mit osh:PhaseOrderTimeShape, ob die implizite Reihenfolge (A → B → C) eingehalten wird.

2. SHACL-Regeln erweitern

Falls du mit der aktuellen OCCP\_SHACL\_min.ttl zufrieden bist, können wir weitere OPs hinzufügen:

* occp:endsBefore, occp:beginsBefore, etc., mit derselben Aggregat-Logik (MAX()/MIN()).
* time:after, time:before, falls sie in deiner TBox vorkommen (ich kann OCCP\_V0.3.ttl prüfen, wenn du willst).

3. Vollständige ABox testen

Wenn die minimale ABox funktioniert, teste die volle OCCP\_Valid\_LCycle\_1.ttl (ohne explizite Beziehungen) mit der erweiterten SHACL-Datei.

Fragen an dich

1. ABox: Hast du die expliziten Beziehungen aus OCCP\_Valid\_LCycle\_1.ttl entfernt, oder war das Log mit den Beziehungen? Kannst du mir die genaue ABox schicken, die du getestet hast?
2. SHACL: Welche Version von OCCP\_SHACL\_min.ttl hast du verwendet? Die mit occp:before/after und osh:PhaseOrderTimeShape?
3. Ziele: Möchtest du jetzt die ABox bereinigen (keine OPs) und nur implizite Reihenfolgen prüfen, oder sollen wir die expliziten Regeln weiterentwickeln?

Vorschlag

Ich könnte eine bereinigte Version von OCCP\_Valid\_LCycle\_1.ttl ohne explizite Beziehungen erstellen und die SHACL-Datei mit allen besprochenen SAs (inklusive endsBefore, beginsBefore) aktualisieren. Dann testest du das mit Jena. Wie klingt das?

Danke für die Zusammenarbeit – es macht echt Spaß, das mit dir auszuarbeiten! Lass mich wissen, wie du weitermachen willst!

Ich habe es mit der OCCP\_Valid\_LCycle\_1.ttl getestet, die sieht so aus:

@prefix

occp: [http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#](http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod_V0.1) .

@prefix

time: [http://www.w3.org/2006/time#](http://www.w3.org/2006/time) .

@prefix

xsd: [http://www.w3.org/2001/XMLSchema#](http://www.w3.org/2001/XMLSchema) .

@prefix

: [http://www.example.de/example#](http://www.example.de/example) .Cycle und LifeCycle:cycle1 a occp:Cycle ; occp:hasCycleNumber 1 ; time:hasBeginning :instant\_cycle\_start ; time:hasEnd :instant\_cycle\_end ; occp:startsCycle :instant\_submission\_to\_review ; occp:endsCycle :instant\_review\_approval .:lifecycle1 a occp:LifeCycle ; occp:hasLifeCycleID "LC001" ; occp:isInPhase :phaseA ; time:hasBeginning :instant\_beginning\_of\_planning ; time:hasEnd :instant\_deconstruction\_completion .Phases und Transitions:phaseA a occp:PhaseA\_Planning ; occp:hasActualBeginning :instant\_beginning\_of\_planning ; occp:hasActualEnd :instant\_completion\_of\_planning ; occp:beginsBefore :phaseB ; occp:beginsBefore :phaseC ; # Hinzugefügt occp:endsBefore :instant\_completion\_of\_construction ; # Geändert occp:startsPhase :instant\_beginning\_of\_planning ; occp:endsPhase :instant\_review\_approval .:phaseB a occp:PhaseB\_Review ; occp:hasActualBeginning :instant\_submission\_to\_review ; occp:hasActualEnd :instant\_review\_approval ; occp:beginsBefore :phaseC ; occp:startsPhase :instant\_submission\_to\_review ; occp:endsPhase :instant\_review\_approval .:transitionA a occp:CompletionOfPlanning ; occp:hasActualTime "2025-03-01"^^xsd:date ; occp:before :transitionB .:transitionB a occp:BeginOfTenderingProcess ; occp:hasActualTime "2025-03-05"^^xsd:date ; occp:before :transitionC .:transitionC a occp:Submission ; occp:hasActualTime "2025-03-10"^^xsd:date ; occp:before :phaseC .:phaseC a occp:PhaseC\_Construction ; occp:hasActualBeginning :instant\_construction\_start ; occp:hasActualEnd :instant\_completion\_of\_construction ; occp:after :transitionA ; occp:after :transitionC ; occp:startsPhase :instant\_construction\_start ; occp:endsPhase :instant\_completion\_of\_construction ; occp:hasPhase :instant\_construction\_start .:transitionD a occp:CompletionOfConstruction ; occp:hasActualTime "2025-06-01"^^xsd:date ; occp:after :transitionA ; occp:before :phaseD .:phaseD a occp:PhaseD\_Usage ; occp:hasActualBeginning :instant\_usage\_start ; occp:hasActualEnd :instant\_usage\_end ; occp:after :transitionD ; occp:endsBefore :phaseM ; occp:startsPhase :instant\_usage\_start ; occp:endsPhase :instant\_usage\_end ; occp:hasPhase :instant\_usage\_start ; occp:hasPhase :instant\_usage\_end .:phaseE a occp:PhaseE\_Warranty ; occp:hasActualBeginning :instant\_warranty\_start ; occp:hasActualEnd :instant\_warranty\_end ; occp:after :transitionD ; occp:startsPhase :instant\_warranty\_start ; occp:endsPhase :instant\_warranty\_end ; occp:hasPhase :instant\_warranty\_start ; occp:hasPhase :instant\_warranty\_end .:phaseF a occp:PhaseF\_DesignLife ; occp:hasActualBeginning :instant\_designlife\_start ; occp:hasActualEnd :instant\_designlife\_end ; occp:after :transitionD ; occp:beginsWith :phaseD ; occp:before :phaseM ; occp:startsPhase :instant\_designlife\_start ; occp:endsPhase :instant\_designlife\_end ; occp:hasPhase :instant\_designlife\_start ; occp:hasPhase :instant\_designlife\_end .:phaseG a occp:PhaseG\_Inspection ; occp:hasActualBeginning :instant\_inspection\_execution ; occp:hasActualEnd :instant\_inspection\_end ; occp:after :transitionD ; occp:before :phaseM ; occp:startsPhase :instant\_inspection\_execution ; occp:endsPhase :instant\_inspection\_end ; occp:hasPhase :instant\_inspection\_execution .:phaseH a occp:PhaseH\_SpecialEvents ; occp:hasActualBeginning :instant\_geological\_event ; occp:hasActualEnd :instant\_weather\_event ; occp:hasPhase :instant\_geological\_event ; occp:hasPhase :instant\_weather\_event .:phaseK a occp:PhaseK\_Damage ; occp:hasActualBeginning :instant\_damage\_detection ; occp:hasActualEnd :instant\_damage\_end ; occp:after :transitionC ; occp:beginsBefore :phaseL ; occp:startsPhase :instant\_damage\_detection ; occp:endsPhase :instant\_damage\_end ; occp:hasPhase :instant\_damage\_detection .:phaseL a occp:PhaseL\_Repair ; occp:hasActualBeginning :instant\_repair\_start ; occp:hasActualEnd :instant\_repair\_completion ; occp:after :transitionC ; occp:endsAfter :phaseK ; occp:before :phaseM ; occp:startsPhase :instant\_repair\_start ; occp:endsPhase :instant\_repair\_completion ; occp:hasPhase :instant\_repair\_start ; occp:hasPhase :instant\_repair\_completion .:phaseM a occp:PhaseM\_Deconstruction ; occp:hasActualBeginning :instant\_deconstruction\_start ; occp:hasActualEnd :instant\_deconstruction\_completion ; occp:after :transitionD ; occp:startsPhase :instant\_deconstruction\_start ; occp:endsPhase :instant\_deconstruction\_completion ; occp:hasPhase :instant\_deconstruction\_start ; occp:hasPhase :instant\_deconstruction\_completion .:transitionE a occp:CompletionOfDeconstruction ; occp:hasActualTime "2025-12-01"^^xsd:date ; occp:endsWith :phaseM .Instants:instant\_cycle\_start a time:Instant ; occp:hasActualTime "2025-01-01"^^xsd:date . :instant\_cycle\_end a time:Instant ; occp:hasActualTime "2025-12-01"^^xsd:date . :instant\_beginning\_of\_planning a occp:BeginningOfPlanning ; occp:hasActualTime "2025-01-01"^^xsd:date . :instant\_submission\_to\_review a occp:SubmissionToReview ; occp:hasActualTime "2025-02-15"^^xsd:date . :instant\_review\_approval a occp:ReviewApproval ; occp:hasActualTime "2025-02-28"^^xsd:date . :instant\_completion\_of\_planning a occp:CompletionOfPlanning ; occp:hasActualTime "2025-03-01"^^xsd:date . :instant\_construction\_start a occp:ConstructionStart ; occp:hasActualTime "2025-03-15"^^xsd:date . :instant\_completion\_of\_construction a occp:CompletionOfConstruction ; occp:hasActualTime "2025-06-01"^^xsd:date . :instant\_usage\_start a occp:UsageStart ; occp:hasActualTime "2025-06-02"^^xsd:date . :instant\_usage\_end a occp:UsageEnd ; occp:hasActualTime "2025-11-01"^^xsd:date . :instant\_warranty\_start a occp:WarrantyStart ; occp:hasActualTime "2025-06-02"^^xsd:date . :instant\_warranty\_end a occp:WarrantyEnd ; occp:hasActualTime "2025-11-01"^^xsd:date . :instant\_designlife\_start a occp:DesignLifeStart ; occp:hasActualTime "2025-06-02"^^xsd:date . :instant\_designlife\_end a occp:DesignLifeEnd ; occp:hasActualTime "2025-11-01"^^xsd:date . :instant\_inspection\_execution a occp:InspectionExecution ; occp:hasActualTime "2025-07-01"^^xsd:date . :instant\_inspection\_end a time:Instant ; occp:hasActualTime "2025-07-02"^^xsd:date . :instant\_geological\_event a occp:GeologicalEvent ; occp:hasActualTime "2025-08-01"^^xsd:date . :instant\_weather\_event a occp:WeatherEvent ; occp:hasActualTime "2025-09-01"^^xsd:date . :instant\_damage\_detection a occp:DamageDetection ; occp:hasActualTime "2025-07-15"^^xsd:date . :instant\_damage\_end a time:Instant ; occp:hasActualTime "2025-07-20"^^xsd:date . :instant\_repair\_start a occp:RepairStart ; occp:hasActualTime "2025-07-21"^^xsd:date . :instant\_repair\_completion a occp:RepairCompletion ; occp:hasActualTime "2025-07-25"^^xsd:date . :instant\_deconstruction\_start a occp:DeconstructionStart ; occp:hasActualTime "2025-11-15"^^xsd:date . :instant\_deconstruction\_completion a occp:CompletionOfDeconstruction ; occp:hasActualTime "2025-12-01"^^xsd:date .Und mit der Phases\_VALID.ttl, die sieht so aus:

@prefix

occp: [http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#](http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod_V0.1) .

@prefix

time: [http://www.w3.org/2006/time#](http://www.w3.org/2006/time) .

@prefix

xsd: [http://www.w3.org/2001/XMLSchema#](http://www.w3.org/2001/XMLSchema) .

@prefix

: [http://www.example.de/example#](http://www.example.de/example) .:phaseA a occp:PhaseA\_Planning ; occp:hasActualBeginning :instant\_beginning\_of\_planning ; occp:hasActualEnd :instant\_completion\_of\_planning .:phaseB a occp:PhaseB\_Review ; occp:hasActualBeginning :instant\_submission\_to\_review ; occp:hasActualEnd :instant\_review\_approval ; time:hasBeginning :instant\_estimated\_start\_b1 ; # Zusätzlicher geschätzter Start time:hasBeginning :instant\_estimated\_start\_b2 .:phaseC a occp:PhaseC\_Construction ; occp:hasActualBeginning :instant\_construction\_start ; occp:hasActualEnd :instant\_completion\_of\_construction .:instant\_beginning\_of\_planning a occp:BeginningOfPlanning ; occp:hasActualTime "2025-01-01"^^xsd:date . :instant\_completion\_of\_planning a occp:CompletionOfPlanning ; occp:hasActualTime "2025-03-01"^^xsd:date . :instant\_submission\_to\_review a occp:SubmissionToReview ; occp:hasActualTime "2025-02-15"^^xsd:date . :instant\_estimated\_start\_b1 a time:Instant ; occp:hasEstimatedTime "2025-02-10"^^xsd:date . # Früherer geschätzter Start :instant\_estimated\_start\_b2 a time:Instant ; occp:hasEstimatedTime "2025-02-20"^^xsd:date . # Späterer geschätzter Start :instant\_review\_approval a occp:ReviewApproval ; occp:hasActualTime "2025-02-28"^^xsd:date . :instant\_construction\_start a occp:ConstructionStart ; occp:hasActualTime "2025-03-15"^^xsd:date . :instant\_completion\_of\_construction a occp:CompletionOfConstruction ; occp:hasActualTime "2025-06-01"^^xsd:date .Beide Reports ohne Fehler! :-)Die OCCP\_SHACL\_min.ttl sieht so aus:

@prefix

sh: [http://www.w3.org/ns/shacl#](http://www.w3.org/ns/shacl) .

@prefix

xsd: [http://www.w3.org/2001/XMLSchema#](http://www.w3.org/2001/XMLSchema) .

@prefix

occp: [http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#](http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod_V0.1) .

@prefix

time: [http://www.w3.org/2006/time#](http://www.w3.org/2006/time) .

@prefix

osh: [http://www.occpshape.de/shape#](http://www.occpshape.de/shape) .Generische Regel für occp:before (ohne Sub-SELECT)osh:BeforeTimeShape a sh:NodeShape ; sh:targetClass time:TemporalEntity ; sh:sparql [ sh:select """ PREFIX occp: [http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#](http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod_V0.1) PREFIX time: [http://www.w3.org/2006/time#](http://www.w3.org/2006/time) SELECT $this ?other ?endTime ?startTime WHERE { $this occp:before ?other . $this time:hasEnd ?endInstant . ?endInstant occp:hasActualTime ?endTime . ?other time:hasBeginning ?startInstant . ?startInstant occp:hasActualTime ?startTime . FILTER (?endTime >= ?startTime) } """ ; sh:message "The end of { $this } (time: { ?endTime }) must be before the beginning of { ?other } (time: { ?startTime }) via 'occp:before'."

@en

; sh:severity sh:Violation ] .Generische Regel für occp:after (ohne Sub-SELECT)osh:AfterTimeShape a sh:NodeShape ; sh:targetClass time:TemporalEntity ; sh:sparql [ sh:select """ PREFIX occp: [http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#](http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod_V0.1) PREFIX time: [http://www.w3.org/2006/time#](http://www.w3.org/2006/time) SELECT $this ?other ?startTime ?endTime WHERE { $this occp:after ?other . $this time:hasBeginning ?startInstant . ?startInstant occp:hasActualTime ?startTime . ?other time:hasEnd ?endInstant . ?endInstant occp:hasActualTime ?endTime . FILTER (?startTime <= ?endTime) } """ ; sh:message "The beginning of { $this } (time: { ?startTime }) must be after the end of { ?other } (time: { ?endTime }) via 'occp:after'."

@en

; sh:severity sh:Violation ] .Implizite Reihenfolge-Regel (ohne Sub-SELECT)osh:PhaseOrderTimeShape a sh:NodeShape ; sh:targetClass occp:Phase ; sh:sparql [ sh:select """ PREFIX occp: [http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#](http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod_V0.1) PREFIX time: [http://www.w3.org/2006/time#](http://www.w3.org/2006/time) SELECT $this ?other ?endTime ?startTime WHERE { $this time:hasEnd ?endInstant . ?endInstant occp:hasActualTime ?endTime . ?other time:hasBeginning ?startInstant . ?startInstant occp:hasActualTime ?startTime . FILTER (?endTime >= ?startTime) FILTER (?other != $this) { { $this a occp:PhaseA\_Planning . ?other a occp:PhaseB\_Review . } UNION { $this a occp:PhaseA\_Planning . ?other a occp:PhaseC\_Construction . } UNION { $this a occp:PhaseB\_Review . ?other a occp:PhaseC\_Construction . } } } """ ; sh:message "The end of { $this } (time: { ?endTime }) must be before the beginning of { ?other } (time: { ?startTime }) based on phase order."

@en

; sh:severity sh:Violation ] .Nun teste ich mit unserer ursprünglichen OCCP\_SHACL.ttl als SHAPE. Hier das Ergebnis:2025-03-13 00:31:14,327 - INFO - Inferierte Ontologie gespeichert: G:\Promo\cModeler\_Git\cMM\OCCP\_Dev2\inferred\_ontology.ttl 2025-03-13 00:31:14,327 - INFO - Pr�fe Ontologie auf Disjunktheit... 2025-03-13 00:31:14,327 - DEBUG - Disjunkte Klassenpaare: set() 2025-03-13 00:31:14,328 - INFO - Keine Disjunktheitsverletzungen gefunden. 2025-03-13 00:31:14,355 - INFO - Starte SPARQL-Abfrage f�r alle UpdateChains... 2025-03-13 00:31:14,470 - INFO - SPARQL-Abfrage Ergebnisse: 2025-03-13 00:31:15,517 - ERROR - Jena SHACL validation stderr: org.apache.jena.shacl.parser.ShaclParseException: No sh:path on a property shape: node=<http://www.occpshape.de/shape#ShapeOfPhaseL> sh:property \_:B235dd7e76fea27c5dfe333b3be7c9dfd at org.apache.jena.shacl.parser.ShapesParser.findPropertyShapes(ShapesParser.java:410) at org.apache.jena.shacl.parser.ShapesParser.parseShape$(ShapesParser.java:334) at org.apache.jena.shacl.parser.ShapesParser.parseShapeStep(ShapesParser.java:312) at org.apache.jena.shacl.parser.ShapesParser.parseShape(ShapesParser.java:243) at org.apache.jena.shacl.parser.ShapesParser.parseShapeAcc(ShapesParser.java:228) at org.apache.jena.shacl.parser.ShapesParser.parseShapes(ShapesParser.java:170) at org.apache.jena.shacl.parser.ShapesParser.parseProcess(ShapesParser.java:99) at org.apache.jena.shacl.Shapes.parseProcess(Shapes.java:115) at org.apache.jena.shacl.Shapes.parseAll(Shapes.java:110) at org.apache.jena.shacl.Shapes.parse(Shapes.java:87) at org.apache.jena.shacl.validation.ShaclPlainValidator.parse(ShaclPlainValidator.java:38) at org.apache.jena.shacl.validation.ShaclPlainValidator.validate(ShaclPlainValidator.java:90) at shacl.shacl\_validate.exec(shacl\_validate.java:127) at org.apache.jena.cmd.CmdMain.mainMethod(CmdMain.java:86) at org.apache.jena.cmd.CmdMain.mainRun(CmdMain.java:55) at org.apache.jena.cmd.CmdMain.mainRun(CmdMain.java:43) at shacl.shacl\_validate.main(shacl\_validate.java:61) at shacl.shacl.main(shacl.java:75)2025-03-13 00:31:15,517 - INFO - Jena SHACL validation stdout: 2025-03-13 00:31:15,518 - INFO - Konformit�t (inference=none): False 2025-03-13 00:31:15,518 - ERROR - Validation failed.Hier der Auszug aus der OCCP\_SHACL.ttl:

osh:ShapeOfPhaseL a sh:NodeShape ; # Phase L (Repair)

sh:targetClass occp:PhaseL\_Repair ;

sh:property [

sh:or (

[ sh:path occp:endsAfter ; sh:class occp:PhaseK\_Damage ]

[ sh:path occp:endsWith ; sh:class occp:PhaseK\_Damage ]

) ;

sh:message "Each 'occp:PhaseL\_Repair' must end after 'occp:PhaseK\_Damage' via 'occp:endsAfter' or with 'occp:PhaseK\_Damage' via 'occp:endsWith' to ensure repair resolves damage."@en ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:after ;

sh:class occp:Submission ;

sh:message "Each 'occp:PhaseL\_Repair' should occur after 'occp:Submission' via 'occp:after'."@en ;

sh:severity sh:Warning ;

] ;

sh:property [

sh:path occp:before ;

sh:class occp:PhaseM\_Deconstruction ;

sh:message "Each 'occp:PhaseL\_Repair' should occur before 'occp:PhaseM\_Deconstruction' via 'occp:before'."@en ;

sh:severity sh:Warning ;

] ;

sh:property [

sh:path occp:hasActualBeginning ;

sh:class occp:RepairStart ;

sh:maxCount 1 ;

sh:message "Each 'occp:PhaseL\_Repair' can only have one actual beginning via 'occp:hasActualBeginning' referencing 'occp:RepairStart'."@en ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:hasActualEnd ;

sh:class occp:RepairCompletion ;

sh:maxCount 1 ;

sh:message "Each 'occp:PhaseL\_Repair' can only have one actual end via 'occp:hasActualEnd' referencing 'occp:RepairCompletion'."@en ;

sh:severity sh:Violation ;

] .

Alles klar, danke für die Updates! Es ist super, dass OCCP\_Valid\_LCycle\_1.ttl und Phases\_VALID.ttl mit der OCCP\_SHACL\_min.ttl ohne Fehler validiert wurden – das zeigt, dass unsere generischen SPARQL-Regeln mit Apache Jena funktionieren. Der Fehler mit der ursprünglichen OCCP\_SHACL.ttl ist ein neues Problem, aber leicht zu beheben. Lass uns das analysieren und die nächsten Schritte planen.

Analyse des aktuellen Standes

1. Erfolgreiche Tests

* OCCP\_Valid\_LCycle\_1.ttl:
  + Enthält viele Phasen (A bis M) und Instants mit expliziten Beziehungen (occp:beginsBefore, occp:endsBefore, occp:after, etc.).
  + Wurde mit OCCP\_SHACL\_min.ttl validiert → Conforms: True.
* Phases\_VALID.ttl:
  + Minimalistische ABox mit Phasen A, B, C und mehreren time:hasBeginning-Werten für Phase B (einschließlich geschätzter Starts).
  + Ebenfalls Conforms: True mit OCCP\_SHACL\_min.ttl.
* OCCP\_SHACL\_min.ttl:
  + Enthält einfache SPARQL-Regeln ohne Sub-SELECTs für occp:before, occp:after und die implizite osh:PhaseOrderTimeShape.
  + Funktioniert mit Jena, obwohl sie keine Aggregatfunktionen (MAX()/MIN()) nutzen, was bei mehreren Zeitstempeln problematisch sein könnte (dazu später mehr).

2. Fehler mit OCCP\_SHACL.ttl

* Log-Auszug:

2025-03-13 00:31:15,517 - ERROR - Jena SHACL validation stderr: org.apache.jena.shacl.parser.ShaclParseException: No sh:path on a property shape: node=<http://www.occpshape.de/shape#ShapeOfPhaseL> sh:property \_:B235dd7e76fea27c5dfe333b3be7c9dfd

* Problem: Jena meldet, dass ein sh:PropertyShape (in diesem Fall innerhalb von osh:ShapeOfPhaseL) kein sh:path hat.
* Betroffene Regel:

turtle

sh:property [

sh:or (

[ sh:path occp:endsAfter ; sh:class occp:PhaseK\_Damage ]

[ sh:path occp:endsWith ; sh:class occp:PhaseK\_Damage ]

) ;

sh:message "Each 'occp:PhaseL\_Repair' must end after 'occp:PhaseK\_Damage' via 'occp:endsAfter' or with 'occp:PhaseK\_Damage' via 'occp:endsWith' to ensure repair resolves damage."@en ;

sh:severity sh:Violation ;

] ;

* Ursache:
  + In SHACL muss ein sh:PropertyShape immer ein einzelnes sh:path haben. Hier wird sh:or verwendet, um zwei verschiedene sh:path-Werte (occp:endsAfter und occp:endsWith) zu kombinieren, was syntaktisch nicht erlaubt ist. sh:or kann auf Wertebene (z. B. verschiedene Klassen oder Datentypen) verwendet werden, aber nicht auf Property-Paths.
  + Jena interpretiert das als fehlendes sh:path auf der obersten Ebene des sh:property, weil die Struktur nicht korrekt verschachtelt ist.

3. Aggregatfunktionen vs. einfache SAs

* Deine OCCP\_SHACL\_min.ttl nutzt keine MAX()/MIN()-Aggregatfunktionen, sondern prüft einfach alle Kombinationen von time:hasEnd und time:hasBeginning. Das funktioniert mit deiner aktuellen ABox, weil:
  + Jede Phase nur einen occp:hasActualEnd und occp:hasActualBeginning hat (durch maxCount 1 erzwungen).
  + Mehrere time:hasBeginning-Werte (z. B. in Phases\_VALID.ttl) nicht zu Konflikten führen, da die Reihenfolge dennoch passt.
* Problem bei Mehrfachwerten: Ohne Aggregatfunktionen wird jede Kombination geprüft, was bei mehreren Zeitstempeln falsche Fehler auslösen könnte (z. B. wenn ein geschätzter Start früher ist als ein tatsächliches Ende, obwohl der tatsächliche Start später ist).

Lösungsvorschläge

1. Fehler in osh:ShapeOfPhaseL beheben

Um den Syntaxfehler zu korrigieren, müssen wir die sh:or-Regel aufteilen in separate sh:property-Blöcke:

turtle

osh:ShapeOfPhaseL a sh:NodeShape ;

sh:targetClass occp:PhaseL\_Repair ;

sh:property [

sh:path occp:endsAfter ;

sh:class occp:PhaseK\_Damage ;

sh:message "Each 'occp:PhaseL\_Repair' must end after 'occp:PhaseK\_Damage' via 'occp:endsAfter'."@en ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:endsWith ;

sh:class occp:PhaseK\_Damage ;

sh:message "Each 'occp:PhaseL\_Repair' must end with 'occp:PhaseK\_Damage' via 'occp:endsWith'."@en ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:after ;

sh:class occp:Submission ;

sh:message "Each 'occp:PhaseL\_Repair' should occur after 'occp:Submission' via 'occp:after'."@en ;

sh:severity sh:Warning ;

] ;

sh:property [

sh:path occp:before ;

sh:class occp:PhaseM\_Deconstruction ;

sh:message "Each 'occp:PhaseL\_Repair' should occur before 'occp:PhaseM\_Deconstruction' via 'occp:before'."@en ;

sh:severity sh:Warning ;

] ;

sh:property [

sh:path occp:hasActualBeginning ;

sh:class occp:RepairStart ;

sh:maxCount 1 ;

sh:message "Each 'occp:PhaseL\_Repair' can only have one actual beginning via 'occp:hasActualBeginning' referencing 'occp:RepairStart'."@en ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:hasActualEnd ;

sh:class occp:RepairCompletion ;

sh:maxCount 1 ;

sh:message "Each 'occp:PhaseL\_Repair' can only have one actual end via 'occp:hasActualEnd' referencing 'occp:RepairCompletion'."@en ;

sh:severity sh:Violation ;

] .

* Änderung: sh:or wird entfernt, und die zwei Bedingungen (endsAfter und endsWith) werden als separate sh:property-Blöcke definiert. Das ist syntaktisch korrekt und bedeutet, dass beide Bedingungen unabhängig geprüft werden (mindestens eine muss erfüllt sein, um keinen Fehler zu werfen, was deiner Intention entspricht).

2. Aggregatfunktionen in OCCP\_SHACL\_min.ttl einführen

Deine aktuelle OCCP\_SHACL\_min.ttl funktioniert, aber für Robustheit bei mehreren Zeitstempeln sollten wir MAX()/MIN() wieder einbauen:

turtle

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

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix osh: <http://www.occpshape.de/shape#> .

### Generische Regel für occp:before mit Aggregaten

osh:BeforeTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

SELECT $this ?other ?latestEndTime ?earliestStartTime WHERE {

$this occp:before ?other .

{

SELECT $this (MAX(?endTime) AS ?latestEndTime)

WHERE {

$this time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY $this

}

{

SELECT ?other (MIN(?startTime) AS ?earliestStartTime)

WHERE {

?other time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY ?other

}

FILTER (?latestEndTime >= ?earliestStartTime)

}

""" ;

sh:message "The latest end of { $this } (time: { ?latestEndTime }) must be before the earliest beginning of { ?other } (time: { ?earliestStartTime }) via 'occp:before'."@en ;

sh:severity sh:Violation

] .

### Generische Regel für occp:after mit Aggregaten

osh:AfterTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

SELECT $this ?other ?earliestStartTime ?latestEndTime WHERE {

$this occp:after ?other .

{

SELECT $this (MIN(?startTime) AS ?earliestStartTime)

WHERE {

$this time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY $this

}

{

SELECT ?other (MAX(?endTime) AS ?latestEndTime)

WHERE {

?other time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY ?other

}

FILTER (?earliestStartTime <= ?latestEndTime)

}

""" ;

sh:message "The earliest beginning of { $this } (time: { ?earliestStartTime }) must be after the latest end of { ?other } (time: { ?latestEndTime }) via 'occp:after'."@en ;

sh:severity sh:Violation

] .

### Implizite Reihenfolge-Regel mit Aggregaten

osh:PhaseOrderTimeShape a sh:NodeShape ;

sh:targetClass occp:Phase ;

sh:sparql [

sh:select """

SELECT $this ?other ?latestEndTime ?earliestStartTime WHERE {

{

SELECT $this (MAX(?endTime) AS ?latestEndTime)

WHERE {

$this time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY $this

}

{

SELECT ?other (MIN(?startTime) AS ?earliestStartTime)

WHERE {

?other time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY ?other

}

FILTER (?latestEndTime >= ?earliestStartTime)

FILTER (?other != $this)

{

{ $this a occp:PhaseA\_Planning . ?other a occp:PhaseB\_Review . }

UNION

{ $this a occp:PhaseA\_Planning . ?other a occp:PhaseC\_Construction . }

UNION

{ $this a occp:PhaseB\_Review . ?other a occp:PhaseC\_Construction . }

}

}

""" ;

sh:message "The latest end of { $this } (time: { ?latestEndTime }) must be before the earliest beginning of { ?other } (time: { ?earliestStartTime }) based on phase order."@en ;

sh:severity sh:Violation

] .

* Warum Aggregatfunktionen?: Deine Phases\_VALID.ttl hat mehrere time:hasBeginning-Werte für PhaseB. Ohne MIN() wird jeder Wert geprüft, was bei komplexeren ABoxen unerwartete Fehler auslösen könnte. Mit Aggregaten wird nur der relevante Zeitpunkt verglichen.

3. ABox bereinigen

Deine OCCP\_Valid\_LCycle\_1.ttl enthält noch viele explizite Beziehungen (z. B. occp:beginsBefore, occp:endsBefore). Wenn wir sie entfernen, könnte sie so aussehen (Auszug):

turtle

:phaseA a occp:PhaseA\_Planning ;

occp:hasActualBeginning :instant\_beginning\_of\_planning ;

occp:hasActualEnd :instant\_completion\_of\_planning .

:phaseB a occp:PhaseB\_Review ;

occp:hasActualBeginning :instant\_submission\_to\_review ;

occp:hasActualEnd :instant\_review\_approval .

:phaseC a occp:PhaseC\_Construction ;

occp:hasActualBeginning :instant\_construction\_start ;

occp:hasActualEnd :instant\_completion\_of\_construction .

* Test: Mit osh:PhaseOrderTimeShape sollte die implizite Reihenfolge (A → B → C) geprüft werden.

Nächste Schritte

1. Fehler beheben:
   * Passe osh:ShapeOfPhaseL in OCCP\_SHACL.ttl wie oben an und teste erneut mit OCCP\_Valid\_LCycle\_1.ttl.
2. Aggregatfunktionen einführen:
   * Aktualisiere OCCP\_SHACL\_min.ttl mit der Version mit Aggregaten und teste mit Phases\_VALID.ttl, um sicherzustellen, dass mehrere Zeitstempel korrekt gehandhabt werden.
3. ABox bereinigen:
   * Entferne die expliziten Beziehungen aus OCCP\_Valid\_LCycle\_1.ttl und teste mit der impliziten Regel (osh:PhaseOrderTimeShape).
4. Erweitern:
   * Füge SAs für occp:endsBefore, occp:beginsBefore, etc. in OCCP\_SHACL.ttl hinzu, falls du explizite Beziehungen beibehalten willst.

Fragen an dich

* ABox: Möchtest du die expliziten Beziehungen aus OCCP\_Valid\_LCycle\_1.ttl entfernen, oder sollen wir sie vorerst beibehalten und die SAs darauf abstimmen?
* SHACL: Soll ich dir eine korrigierte OCCP\_SHACL.ttl mit dem fix für osh:ShapeOfPhaseL schicken?
* Aggregatfunktionen: Willst du MAX()/MIN() jetzt einführen, oder erst nach der Bereinigung der ABox?

Ich kann die Dateien für dich anpassen – lass mich wissen, wie du vorgehen willst!

Heißt das, dass ich jetzt alle sh:or und sh:and Funktionen in meinen SHACL Rules anpassen muss? Warum erzeugen die Fehler? Zur OCCP\_SHACL\_min.ttl: Hilf mir mal, wie genau wollen wir bei der Erstellung und Validierung der zusätzlichen SPARQL-Queries vorgehen? Brauchen wir die OCCP\_SHACL.ttl gerade überhaupt oder sollten wir erstmal nur mit der OCCP\_SHACL\_min.ttl arbeiten? Und welche ABox nutzen wir für den Test der generischen SPARQL-Rules?

Keine Panik – ich helfe dir, das alles Schritt für Schritt zu klären! Deine Fragen sind super, und ich werde sie detailliert beantworten, damit wir einen klaren Plan haben.

1. Muss ich alle sh:or und sh:and Funktionen in meinen SHACL Rules anpassen?

Antwort: Nein, nicht grundsätzlich, aber es kommt auf die Struktur an.

* Warum der Fehler auftritt:
  + In SHACL gibt es eine strikte Unterscheidung zwischen Node Shapes (z. B. sh:NodeShape) und Property Shapes (z. B. sh:property). Ein sh:PropertyShape muss immer genau einen sh:path haben, der definiert, welche Property geprüft wird.
  + In deinem Fall (osh:ShapeOfPhaseL) hast du sh:or verwendet, um zwei verschiedene sh:path-Werte (occp:endsAfter und occp:endsWith) innerhalb eines einzigen sh:property-Blocks zu kombinieren:

turtle

sh:property [

sh:or (

[ sh:path occp:endsAfter ; sh:class occp:PhaseK\_Damage ]

[ sh:path occp:endsWith ; sh:class occp:PhaseK\_Damage ]

) ;

] ;

Das ist syntaktisch ungültig, weil sh:or auf der Ebene der Constraints (z. B. Klassen, Datentypen) arbeitet, nicht auf der Ebene der sh:path-Definition. Jena erkennt das als fehlendes sh:path auf der obersten Ebene des sh:property und wirft die Exception: No sh:path on a property shape.

* Wann erzeugen sh:or und sh:and Fehler?:
  + Fehler: Wenn sie verwendet werden, um verschiedene sh:path-Werte innerhalb eines sh:property-Blocks zu kombinieren (wie oben).
  + Kein Fehler: Wenn sie innerhalb eines sh:property mit einem einzelnen sh:path Constraints kombinieren, z. B.:

turtle

sh:property [

sh:path occp:hasActualEnd ;

sh:or (

[ sh:class occp:ReviewApproval ]

[ sh:class occp:CompletionOfPlanning ]

) ;

sh:maxCount 1 ;

] ;

Hier ist sh:path occp:hasActualEnd eindeutig definiert, und sh:or kombiniert nur die erlaubten Klassen – das ist korrekt und funktioniert.

* Müssen alle angepasst werden?:
  + Nein: Du musst nur die sh:or- und sh:and-Blöcke anpassen, die mehrere sh:path-Werte in einem sh:property kombinieren. Wenn sh:or oder sh:and nur Werte (z. B. Klassen, Literale) innerhalb eines einzelnen sh:path kombiniert, sind sie in Ordnung.
  + Prüfung: Schau dir deine OCCP\_SHACL.ttl an und suche nach sh:property-Blöcken mit sh:or oder sh:and. Wenn sie mehrere sh:path-Werte enthalten, teile sie in separate sh:property-Blöcke auf.
* Lösung (Beispiel):
  + Ursprünglich (falsch):

turtle

sh:property [

sh:or (

[ sh:path occp:endsAfter ; sh:class occp:PhaseK\_Damage ]

[ sh:path occp:endsWith ; sh:class occp:PhaseK\_Damage ]

) ;

] ;

* + Korrigiert (richtig):

turtle

sh:property [

sh:path occp:endsAfter ;

sh:class occp:PhaseK\_Damage ;

] ;

sh:property [

sh:path occp:endsWith ;

sh:class occp:PhaseK\_Damage ;

] ;

* + Semantik: Jetzt wird geprüft, ob mindestens eine der beiden Bedingungen erfüllt ist (was deiner Intention entspricht), aber die Syntax ist SHACL-konform.
* Aktion:
  + Durchforste OCCP\_SHACL.ttl nach ähnlichen Fällen (z. B. sh:or oder sh:and mit mehreren sh:path). Ich kann dir dabei helfen, wenn du mir die Datei komplett schickst oder einen größeren Auszug!

2. OCCP\_SHACL\_min.ttl: Wie gehen wir bei zusätzlichen SPARQL-Queries vor?

Vorgehensweise

* Ziel: Die generischen SPARQL-Regeln in OCCP\_SHACL\_min.ttl erweitern, um alle relevanten chronologischen Object Properties (OPs) abzudecken (occp:beginsBefore, occp:endsBefore, etc.) und sicherzustellen, dass sie bei mehreren Zeitstempeln robust sind.
* Schritte:
  1. Liste der OPs erstellen: Identifiziere alle chronologischen OPs aus deiner Ontologie (OCCP\_V0.3.ttl). Basierend auf OCCP\_Valid\_LCycle\_1.ttl sehe ich:
     + occp:before
     + occp:after
     + occp:beginsBefore
     + occp:endsBefore
     + occp:endsAfter
     + occp:beginsWith
     + occp:endsWith
  2. SPARQL-Regeln definieren: Für jede OP eine generische Regel mit Aggregatfunktionen (MAX()/MIN()), um den spätesten/frühesten Zeitstempel zu prüfen.
  3. Validierung testen: Teste jede Regel mit einer ABox, die die entsprechende OP verwendet.
  4. Iterieren: Füge Regeln schrittweise hinzu und prüfe mit Jena.

Beispiel für zusätzliche SPARQL-Regeln

Hier eine erweiterte OCCP\_SHACL\_min.ttl mit occp:beginsBefore und occp:endsBefore:

turtle

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

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix osh: <http://www.occpshape.de/shape#> .

### Generische Regel für occp:before

osh:BeforeTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

SELECT $this ?other ?latestEndTime ?earliestStartTime WHERE {

$this occp:before ?other .

{

SELECT $this (MAX(?endTime) AS ?latestEndTime)

WHERE {

$this time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY $this

}

{

SELECT ?other (MIN(?startTime) AS ?earliestStartTime)

WHERE {

?other time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY ?other

}

FILTER (?latestEndTime >= ?earliestStartTime)

}

""" ;

sh:message "The latest end of { $this } (time: { ?latestEndTime }) must be before the earliest beginning of { ?other } (time: { ?earliestStartTime }) via 'occp:before'."@en ;

sh:severity sh:Violation

] .

### Generische Regel für occp:after

osh:AfterTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

SELECT $this ?other ?earliestStartTime ?latestEndTime WHERE {

$this occp:after ?other .

{

SELECT $this (MIN(?startTime) AS ?earliestStartTime)

WHERE {

$this time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY $this

}

{

SELECT ?other (MAX(?endTime) AS ?latestEndTime)

WHERE {

?other time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY ?other

}

FILTER (?earliestStartTime <= ?latestEndTime)

}

""" ;

sh:message "The earliest beginning of { $this } (time: { ?earliestStartTime }) must be after the latest end of { ?other } (time: { ?latestEndTime }) via 'occp:after'."@en ;

sh:severity sh:Violation

] .

### Generische Regel für occp:beginsBefore

osh:BeginsBeforeTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

SELECT $this ?other ?latestStartTime ?earliestStartTime WHERE {

$this occp:beginsBefore ?other .

{

SELECT $this (MAX(?startTime) AS ?latestStartTime)

WHERE {

$this time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY $this

}

{

SELECT ?other (MIN(?startTime) AS ?earliestStartTime)

WHERE {

?other time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY ?other

}

FILTER (?latestStartTime >= ?earliestStartTime)

}

""" ;

sh:message "The latest beginning of { $this } (time: { ?latestStartTime }) must be before the earliest beginning of { ?other } (time: { ?earliestStartTime }) via 'occp:beginsBefore'."@en ;

sh:severity sh:Violation

] .

### Generische Regel für occp:endsBefore

osh:EndsBeforeTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

SELECT $this ?other ?latestEndTime ?earliestEndTime WHERE {

$this occp:endsBefore ?other .

{

SELECT $this (MAX(?endTime) AS ?latestEndTime)

WHERE {

$this time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY $this

}

{

SELECT ?other (MIN(?endTime) AS ?earliestEndTime)

WHERE {

?other time:hasEnd ?endInstant .

?startInstant occp:hasActualTime ?endTime .

}

GROUP BY ?other

}

FILTER (?latestEndTime >= ?earliestEndTime)

}

""" ;

sh:message "The latest end of { $this } (time: { ?latestEndTime }) must be before the earliest end of { ?other } (time: { ?earliestEndTime }) via 'occp:endsBefore'."@en ;

sh:severity sh:Violation

] .

### Implizite Reihenfolge-Regel

osh:PhaseOrderTimeShape a sh:NodeShape ;

sh:targetClass occp:Phase ;

sh:sparql [

sh:select """

SELECT $this ?other ?latestEndTime ?earliestStartTime WHERE {

{

SELECT $this (MAX(?endTime) AS ?latestEndTime)

WHERE {

$this time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY $this

}

{

SELECT ?other (MIN(?startTime) AS ?earliestStartTime)

WHERE {

?other time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY ?other

}

FILTER (?latestEndTime >= ?earliestStartTime)

FILTER (?other != $this)

{

{ $this a occp:PhaseA\_Planning . ?other a occp:PhaseB\_Review . }

UNION

{ $this a occp:PhaseA\_Planning . ?other a occp:PhaseC\_Construction . }

UNION

{ $this a occp:PhaseB\_Review . ?other a occp:PhaseC\_Construction . }

}

}

""" ;

sh:message "The latest end of { $this } (time: { ?latestEndTime }) must be before the earliest beginning of { ?other } (time: { ?earliestStartTime }) based on phase order."@en ;

sh:severity sh:Violation

] .

Validierung

* Test-ABox: Nutze OCCP\_Valid\_LCycle\_1.ttl, da sie viele dieser OPs enthält. Wir könnten sie später bereinigen, aber für den Test ist sie ideal.
* Vorgehen:
  1. Teste die aktuelle OCCP\_SHACL\_min.ttl mit OCCP\_Valid\_LCycle\_1.ttl, um sicherzustellen, dass before und after korrekt geprüft werden.
  2. Füge beginsBefore hinzu und teste erneut.
  3. Füge endsBefore hinzu und teste wieder.
  4. Wiederhole das für endsAfter, beginsWith, endsWith.

3. Brauchen wir OCCP\_SHACL.ttl gerade oder sollten wir mit OCCP\_SHACL\_min.ttl arbeiten?

Antwort: Erstmal mit OCCP\_SHACL\_min.ttl arbeiten

* Warum:
  + OCCP\_SHACL\_min.ttl ist schlank, fokussiert auf generische SPARQL-Regeln und funktioniert mit Jena. Sie ist ideal, um die grundlegenden chronologischen Regeln zu perfektionieren.
  + OCCP\_SHACL.ttl ist komplexer, enthält spezifische Phasen-Regeln und hat Syntaxprobleme (wie bei osh:ShapeOfPhaseL). Es macht Sinn, sie erst zu nutzen, wenn die generischen Regeln stabil sind und die Syntax korrigiert ist.
* Plan:
  + Arbeite mit OCCP\_SHACL\_min.ttl, bis wir alle generischen SPARQL-Regeln für die OPs haben.
  + Korrigiere parallel die sh:or/sh:and-Probleme in OCCP\_SHACL.ttl.
  + Sobald OCCP\_SHACL\_min.ttl komplett ist, integrieren wir die generischen Regeln in OCCP\_SHACL.ttl und testen das Ganze.

4. Welche ABox für den Test der generischen SPARQL-Regeln?

Antwort: OCCP\_Valid\_LCycle\_1.ttl (zunächst mit OPs, später bereinigt)

* Warum:
  + Sie enthält viele Phasen und Instants sowie die meisten chronologischen OPs (before, after, beginsBefore, etc.), was sie perfekt macht, um die SPARQL-Regeln zu testen.
  + Die Zeitstempel sind chronologisch konsistent, wie der erfolgreiche Test mit OCCP\_SHACL\_min.ttl zeigt.
* Langfristig: Du möchtest die expliziten Beziehungen entfernen und nur Zeitstempel nutzen. Dafür können wir später eine bereinigte Version erstellen:

turtle

:phaseA a occp:PhaseA\_Planning ;

occp:hasActualBeginning :instant\_beginning\_of\_planning ;

occp:hasActualEnd :instant\_completion\_of\_planning .

:phaseB a occp:PhaseB\_Review ;

occp:hasActualBeginning :instant\_submission\_to\_review ;

occp:hasActualEnd :instant\_review\_approval .

Aber für jetzt ist die aktuelle Version mit OPs besser, um die Regeln zu validieren.

Vorschlag für den nächsten Schritt

1. Korrigiere OCCP\_SHACL.ttl:
   * Ich kann dir eine Version mit angepasstem osh:ShapeOfPhaseL (und ggf. anderen sh:or/sh:and-Blöcken) schicken. Schick mir einfach die volle Datei, falls du mehr als osh:ShapeOfPhaseL prüfen willst.
2. Erweitere OCCP\_SHACL\_min.ttl:
   * Teste die oben vorgeschlagene Version mit beginsBefore und endsBefore mit OCCP\_Valid\_LCycle\_1.ttl.
   * Füge nach und nach die anderen OPs hinzu.
3. Test-ABox:
   * Nutze zunächst OCCP\_Valid\_LCycle\_1.ttl wie sie ist.
   * Wenn die Regeln stabil sind, erstelle ich eine bereinigte Version ohne OPs und teste mit osh:PhaseOrderTimeShape.

Wie genau vorgehen?

* Test 1: OCCP\_SHACL\_min.ttl mit beginsBefore und endsBefore → OCCP\_Valid\_LCycle\_1.ttl.
* Test 2: Füge endsAfter hinzu → gleiche ABox.
* Test 3: Bereinigte ABox ohne OPs → nur osh:PhaseOrderTimeShape.

Meine Fragen an dich

* SHACL: Soll ich dir die korrigierte OCCP\_SHACL.ttl mit osh:ShapeOfPhaseL schicken, oder machst du das selbst?
* ABox: Willst du jetzt schon die OPs aus OCCP\_Valid\_LCycle\_1.ttl entfernen, oder erst nach den Tests der generischen Regeln?
* Priorität: Was ist dir wichtiger: OCCP\_SHACL.ttl fixen oder OCCP\_SHACL\_min.ttl erweitern?

Ich stehe bereit, dir die Dateien anzupassen – lass mich wissen, wie du weitermachen willst!

Jump! Alle sh:or und sh:and Syntaxfehler sind korrigiert und in der OCCP\_SHACL.ttl integriert. Lass uns mit den SPARQL-Abfragen weitermachen, inkl. deren paralleler Validierung. :-)

Yeah, das ist ein Sprung nach vorne! Super, dass du alle sh:or- und sh:and-Syntaxfehler in OCCP\_SHACL.ttl korrigiert hast – jetzt können wir uns voll auf die SPARQL-Abfragen konzentrieren und sie parallel validieren. Lass uns loslegen!

Plan: SPARQL-Abfragen erweitern und validieren

Ziel

* Erweitere OCCP\_SHACL\_min.ttl mit generischen SPARQL-Regeln für alle chronologischen Object Properties (OPs) aus deiner Ontologie.
* Validiere sie parallel mit OCCP\_Valid\_LCycle\_1.ttl, um sicherzustellen, dass sie die Zeitstempel korrekt prüfen.

Schritte

1. Liste der OPs: Basierend auf OCCP\_Valid\_LCycle\_1.ttl und deiner Ontologie (OCCP\_V0.3.ttl) identifizieren wir alle relevanten OPs.
2. SPARQL-Regeln erstellen: Für jede OP eine Regel mit Aggregatfunktionen (MAX()/MIN()), um mehrere Zeitstempel robust zu handhaben.
3. Parallele Validierung: Teste jede neue Regel sofort mit OCCP\_Valid\_LCycle\_1.ttl in deinem Jena-Skript und prüfe das Log.
4. Integration: Sobald alle Regeln stabil sind, fügen wir sie in OCCP\_SHACL.ttl ein (falls gewünscht).

Schritt 1: Liste der OPs

Aus OCCP\_Valid\_LCycle\_1.ttl sehe ich folgende chronologische OPs:

* occp:before
* occp:after
* occp:beginsBefore
* occp:endsBefore
* occp:endsAfter
* occp:beginsWith
* occp:endsWith

Falls es weitere in OCCP\_V0.3.ttl gibt (z. B. occp:startsAfter, occp:finishesBefore), ergänze sie bitte! Für jetzt nehmen wir diese sieben.

Schritt 2: SPARQL-Regeln erstellen

Hier ist die aktualisierte OCCP\_SHACL\_min.ttl mit allen sieben OPs:

turtle

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

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix osh: <http://www.occpshape.de/shape#> .

### Generische Regel für occp:before

osh:BeforeTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

SELECT $this ?other ?latestEndTime ?earliestStartTime WHERE {

$this occp:before ?other .

{

SELECT $this (MAX(?endTime) AS ?latestEndTime)

WHERE {

$this time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY $this

}

{

SELECT ?other (MIN(?startTime) AS ?earliestStartTime)

WHERE {

?other time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY ?other

}

FILTER (?latestEndTime >= ?earliestStartTime)

}

""" ;

sh:message "The latest end of { $this } (time: { ?latestEndTime }) must be before the earliest beginning of { ?other } (time: { ?earliestStartTime }) via 'occp:before'."@en ;

sh:severity sh:Violation

] .

### Generische Regel für occp:after

osh:AfterTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

SELECT $this ?other ?earliestStartTime ?latestEndTime WHERE {

$this occp:after ?other .

{

SELECT $this (MIN(?startTime) AS ?earliestStartTime)

WHERE {

$this time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY $this

}

{

SELECT ?other (MAX(?endTime) AS ?latestEndTime)

WHERE {

?other time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY ?other

}

FILTER (?earliestStartTime <= ?latestEndTime)

}

""" ;

sh:message "The earliest beginning of { $this } (time: { ?earliestStartTime }) must be after the latest end of { ?other } (time: { ?latestEndTime }) via 'occp:after'."@en ;

sh:severity sh:Violation

] .

### Generische Regel für occp:beginsBefore

osh:BeginsBeforeTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

SELECT $this ?other ?latestStartTime ?earliestStartTime WHERE {

$this occp:beginsBefore ?other .

{

SELECT $this (MAX(?startTime) AS ?latestStartTime)

WHERE {

$this time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY $this

}

{

SELECT ?other (MIN(?startTime) AS ?earliestStartTime)

WHERE {

?other time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY ?other

}

FILTER (?latestStartTime >= ?earliestStartTime)

}

""" ;

sh:message "The latest beginning of { $this } (time: { ?latestStartTime }) must be before the earliest beginning of { ?other } (time: { ?earliestStartTime }) via 'occp:beginsBefore'."@en ;

sh:severity sh:Violation

] .

### Generische Regel für occp:endsBefore

osh:EndsBeforeTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

SELECT $this ?other ?latestEndTime ?earliestEndTime WHERE {

$this occp:endsBefore ?other .

{

SELECT $this (MAX(?endTime) AS ?latestEndTime)

WHERE {

$this time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY $this

}

{

SELECT ?other (MIN(?endTime) AS ?earliestEndTime)

WHERE {

?other time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY ?other

}

FILTER (?latestEndTime >= ?earliestEndTime)

}

""" ;

sh:message "The latest end of { $this } (time: { ?latestEndTime }) must be before the earliest end of { ?other } (time: { ?earliestEndTime }) via 'occp:endsBefore'."@en ;

sh:severity sh:Violation

] .

### Generische Regel für occp:endsAfter

osh:EndsAfterTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

SELECT $this ?other ?earliestEndTime ?latestEndTime WHERE {

$this occp:endsAfter ?other .

{

SELECT $this (MIN(?endTime) AS ?earliestEndTime)

WHERE {

$this time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY $this

}

{

SELECT ?other (MAX(?endTime) AS ?latestEndTime)

WHERE {

?other time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY ?other

}

FILTER (?earliestEndTime <= ?latestEndTime)

}

""" ;

sh:message "The earliest end of { $this } (time: { ?earliestEndTime }) must be after the latest end of { ?other } (time: { ?latestEndTime }) via 'occp:endsAfter'."@en ;

sh:severity sh:Violation

] .

### Generische Regel für occp:beginsWith

osh:BeginsWithTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

SELECT $this ?other ?startTimeThis ?startTimeOther WHERE {

$this occp:beginsWith ?other .

{

SELECT $this (MIN(?startTime) AS ?startTimeThis)

WHERE {

$this time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY $this

}

{

SELECT ?other (MIN(?startTime) AS ?startTimeOther)

WHERE {

?other time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY ?other

}

FILTER (?startTimeThis != ?startTimeOther)

}

""" ;

sh:message "The earliest beginning of { $this } (time: { ?startTimeThis }) must equal the earliest beginning of { ?other } (time: { ?startTimeOther }) via 'occp:beginsWith'."@en ;

sh:severity sh:Violation

] .

### Generische Regel für occp:endsWith

osh:EndsWithTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

SELECT $this ?other ?endTimeThis ?endTimeOther WHERE {

$this occp:endsWith ?other .

{

SELECT $this (MAX(?endTime) AS ?endTimeThis)

WHERE {

$this time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY $this

}

{

SELECT ?other (MAX(?endTime) AS ?endTimeOther)

WHERE {

?other time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY ?other

}

FILTER (?endTimeThis != ?endTimeOther)

}

""" ;

sh:message "The latest end of { $this } (time: { ?endTimeThis }) must equal the latest end of { ?other } (time: { ?endTimeOther }) via 'occp:endsWith'."@en ;

sh:severity sh:Violation

] .

Erklärung

* before: Spätestes Ende von $this vor frühestem Anfang von ?other.
* after: Frühester Anfang von $this nach spätestem Ende von ?other.
* beginsBefore: Spätester Anfang von $this vor frühestem Anfang von ?other.
* endsBefore: Spätestes Ende von $this vor frühestem Ende von ?other.
* endsAfter: Frühestes Ende von $this nach spätestem Ende von ?other.
* beginsWith: Frühester Anfang von $this gleich frühestem Anfang von ?other.
* endsWith: Spätestes Ende von $this gleich spätestem Ende von ?other.

Schritt 3: Parallele Validierung

Vorgehensweise

* ABox: OCCP\_Valid\_LCycle\_1.ttl (mit expliziten OPs), da sie alle oben genannten OPs enthält.
* Testmethode:
  1. Starte mit der Basisversion von OCCP\_SHACL\_min.ttl (nur before und after).
  2. Füge eine Regel nach der anderen hinzu und validiere jedes Mal mit deinem Skript (validation\_shacl.py).
  3. Prüfe das Log nach Conforms: True oder Fehlern.

Testplan

1. Basis (before, after):
   * Kopiere die aktuelle OCCP\_SHACL\_min.ttl mit nur osh:BeforeTimeShape und osh:AfterTimeShape.
   * Teste mit OCCP\_Valid\_LCycle\_1.ttl.
   * Erwartung: Conforms: True (wie dein letzter Test zeigt).
2. Erweiterung 1 (beginsBefore):
   * Füge osh:BeginsBeforeTimeShape hinzu.
   * Teste wieder.
   * Prüfe: :phaseA occp:beginsBefore :phaseB (2025-01-01 < 2025-02-15 → OK).
3. Erweiterung 2 (endsBefore):
   * Füge osh:EndsBeforeTimeShape hinzu.
   * Teste.
   * Prüfe: :phaseA occp:endsBefore :instant\_completion\_of\_construction (2025-03-01 < 2025-06-01 → OK).
4. Erweiterung 3 (endsAfter):
   * Füge osh:EndsAfterTimeShape hinzu.
   * Teste.
   * Prüfe: :phaseL occp:endsAfter :phaseK (2025-07-25 > 2025-07-20 → OK).
5. Erweiterung 4 (beginsWith, endsWith):
   * Füge osh:BeginsWithTimeShape und osh:EndsWithTimeShape hinzu.
   * Teste.
   * Prüfe: :phaseF occp:beginsWith :phaseD (2025-06-02 = 2025-06-02 → OK), :transitionE occp:endsWith :phaseM (2025-12-01 = 2025-12-01 → OK).

Hinweis

* Deine ABox ist chronologisch konsistent (wie der erste Test zeigt), also sollten alle Tests Conforms: True ergeben, solange die Regeln korrekt sind.
* Falls ein Fehler auftaucht, liegt es an der SPARQL-Abfrage – dann passen wir sie an.

Schritt 4: Integration in OCCP\_SHACL.ttl (optional)

* Sobald alle Regeln in OCCP\_SHACL\_min.ttl validiert sind, können wir sie in OCCP\_SHACL.ttl einfügen, um sie mit den spezifischen Phasen-Regeln zu kombinieren.
* Alternativ: Belasse sie in OCCP\_SHACL\_min.ttl, wenn du die generischen Regeln separat halten willst.

Nächste Aktion

1. Teste die Basis:
   * Nimm die oben gezeigte OCCP\_SHACL\_min.ttl mit allen sieben Regeln (oder starte mit before und after und füge schrittweise hinzu).
   * Validiere mit OCCP\_Valid\_LCycle\_1.ttl in deinem Skript.
   * Teile mir das Log mit!
2. Feedback: Wenn etwas nicht stimmt (z. B. unerwartete Fehler), debuggen wir die betroffene Regel zusammen.

Fragen an dich

* Startpunkt: Willst du alle Regeln auf einmal testen, oder Schritt für Schritt (wie im Testplan)?
* Hilfe: Soll ich dir die OCCP\_SHACL\_min.ttl mit allen Regeln als Datei schicken, oder kopierst du sie selbst?
* Integration: Möchtest du die Regeln später in OCCP\_SHACL.ttl einfügen, oder erstmal separat lassen?

Lass uns die SPARQL-Party starten – ich bin gespannt auf dein nächstes Log!

Super! Hier das Ergebnis mit before und after...2025-03-13 01:32:43,993 - INFO - Inferierte Ontologie gespeichert: G:\Promo\cModeler\_Git\cMM\OCCP\_Dev2\inferred\_ontology.ttl 2025-03-13 01:32:43,993 - INFO - Pr�fe Ontologie auf Disjunktheit... 2025-03-13 01:32:43,993 - DEBUG - Disjunkte Klassenpaare: set() 2025-03-13 01:32:43,994 - INFO - Keine Disjunktheitsverletzungen gefunden. 2025-03-13 01:32:44,028 - INFO - Starte SPARQL-Abfrage f�r alle UpdateChains... 2025-03-13 01:32:44,147 - INFO - SPARQL-Abfrage Ergebnisse: 2025-03-13 01:32:45,211 - ERROR - Jena SHACL validation stderr: org.apache.jena.shacl.parser.ShaclParseException: Bad query: Line 4, column 23: Unresolved prefixed name: occp:after at org.apache.jena.shacl.lib.ShLib.parseQueryString(ShLib.java:276) at org.apache.jena.shacl.lib.ShLib.extractSPARQLQuery(ShLib.java:284) at org.apache.jena.shacl.engine.SparqlConstraints.parseSparqlConstraint(SparqlConstraints.java:64) at org.apache.jena.shacl.parser.Constraints.lambda$static$20(Constraints.java:115) at org.apache.jena.shacl.parser.Constraints.parseConstraint(Constraints.java:173) at org.apache.jena.shacl.parser.Constraints.lambda$parseConstraints$21(Constraints.java:158) at org.apache.jena.mem2.store.fast.FastArrayBunch$1.forEachRemaining(FastArrayBunch.java:164) at org.apache.jena.shacl.parser.Constraints.parseConstraints(Constraints.java:149) at org.apache.jena.shacl.parser.ShapesParser.parseShape$(ShapesParser.java:326) at org.apache.jena.shacl.parser.ShapesParser.parseShapeStep(ShapesParser.java:312) at org.apache.jena.shacl.parser.ShapesParser.parseShape(ShapesParser.java:243) at org.apache.jena.shacl.parser.ShapesParser.parseShapeAcc(ShapesParser.java:228) at org.apache.jena.shacl.parser.ShapesParser.parseShapes(ShapesParser.java:170) at org.apache.jena.shacl.parser.ShapesParser.parseProcess(ShapesParser.java:99) at org.apache.jena.shacl.Shapes.parseProcess(Shapes.java:115) at org.apache.jena.shacl.Shapes.parseAll(Shapes.java:110) at org.apache.jena.shacl.Shapes.parse(Shapes.java:87) at org.apache.jena.shacl.validation.ShaclPlainValidator.parse(ShaclPlainValidator.java:38) at org.apache.jena.shacl.validation.ShaclPlainValidator.validate(ShaclPlainValidator.java:90) at shacl.shacl\_validate.exec(shacl\_validate.java:127) at org.apache.jena.cmd.CmdMain.mainMethod(CmdMain.java:86) at org.apache.jena.cmd.CmdMain.mainRun(CmdMain.java:55) at org.apache.jena.cmd.CmdMain.mainRun(CmdMain.java:43) at shacl.shacl\_validate.main(shacl\_validate.java:61) at shacl.shacl.main(shacl.java:75)2025-03-13 01:32:45,211 - INFO - Jena SHACL validation stdout: 2025-03-13 01:32:45,212 - INFO - Konformit�t (inference=none): False 2025-03-13 01:32:45,212 - ERROR - Validation failed.

Okay, kein Grund zur Sorge – wir haben einen kleinen Stolperstein, aber das kriegen wir schnell hin! Der Fehler im Log zeigt ein Problem mit der SPARQL-Abfrage in OCCP\_SHACL\_min.ttl. Lass uns das analysieren und beheben.

Analyse des Fehlers

Log-Auszug

2025-03-13 01:32:45,211 - ERROR - Jena SHACL validation stderr: org.apache.jena.shacl.parser.ShaclParseException: Bad query: Line 4, column 23: Unresolved prefixed name: occp:after

Problem

* Fehlermeldung: Unresolved prefixed name: occp:after.
* Ursache: Jena kann den Namespace-Präfix occp: in der SPARQL-Abfrage nicht auflösen. Das passiert, weil die PREFIX-Definitionen innerhalb der sh:select-Blöcke nicht automatisch die Namespace-Definitionen aus dem Turtle-Dokument übernehmen. In SHACL müssen SPARQL-Abfragen ihre Präfixe explizit im Query-String definieren, auch wenn sie schon am Anfang der .ttl-Datei stehen.

Betroffene Stelle

* In deiner OCCP\_SHACL\_min.ttl hast du die Präfixe am Anfang definiert:

turtle

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix time: <http://www.w3.org/2006/time#> .

* Aber in den SPARQL-Abfragen (z. B. für osh:AfterTimeShape) fehlen die PREFIX-Definitionen:

turtle

sh:select """

SELECT $this ?other ?earliestStartTime ?latestEndTime WHERE {

$this occp:after ?other .

...

}

"""

* Ohne PREFIX occp: <...> im Query-String weiß Jena nicht, was occp:after bedeutet, daher der Fehler.

Warum passiert das?

* Im Gegensatz zu Turtle-Dateien, wo Präfixe global gelten, sind SPARQL-Abfragen in SHACL eigenständig. Die sh:select-Eigenschaft erwartet eine vollständige SPARQL-Abfrage mit allen benötigten Präfixen direkt im String.

Lösung

Wir müssen die PREFIX-Definitionen in jede SPARQL-Abfrage einfügen. Hier ist die korrigierte Version von OCCP\_SHACL\_min.ttl mit before und after, inklusive Präfixen in den Abfragen:

turtle

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

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix osh: <http://www.occpshape.de/shape#> .

### Generische Regel für occp:before

osh:BeforeTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

PREFIX occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#>

PREFIX time: <http://www.w3.org/2006/time#>

SELECT $this ?other ?latestEndTime ?earliestStartTime WHERE {

$this occp:before ?other .

{

SELECT $this (MAX(?endTime) AS ?latestEndTime)

WHERE {

$this time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY $this

}

{

SELECT ?other (MIN(?startTime) AS ?earliestStartTime)

WHERE {

?other time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY ?other

}

FILTER (?latestEndTime >= ?earliestStartTime)

}

""" ;

sh:message "The latest end of { $this } (time: { ?latestEndTime }) must be before the earliest beginning of { ?other } (time: { ?earliestStartTime }) via 'occp:before'."@en ;

sh:severity sh:Violation

] .

### Generische Regel für occp:after

osh:AfterTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

PREFIX occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#>

PREFIX time: <http://www.w3.org/2006/time#>

SELECT $this ?other ?earliestStartTime ?latestEndTime WHERE {

$this occp:after ?other .

{

SELECT $this (MIN(?startTime) AS ?earliestStartTime)

WHERE {

$this time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY $this

}

{

SELECT ?other (MAX(?endTime) AS ?latestEndTime)

WHERE {

?other time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY ?other

}

FILTER (?earliestStartTime <= ?latestEndTime)

}

""" ;

sh:message "The earliest beginning of { $this } (time: { ?earliestStartTime }) must be after the latest end of { ?other } (time: { ?latestEndTime }) via 'occp:after'."@en ;

sh:severity sh:Violation

] .

Änderungen

* PREFIX-Definitionen hinzugefügt: Jede sh:select-Abfrage enthält jetzt:
  + PREFIX occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#>
  + PREFIX time: <http://www.w3.org/2006/time#>
* Rest unverändert: Die Logik der Abfragen bleibt gleich, nur die Präfixe wurden ergänzt.

Nächster Schritt: Testen

1. Aktualisiere OCCP\_SHACL\_min.ttl:
   * Ersetze deine aktuelle Version mit der oben gezeigten (nur before und after).
2. Validiere mit OCCP\_Valid\_LCycle\_1.ttl:
   * Führe dein Skript (validation\_shacl.py) erneut aus.
   * Erwartung: Conforms: True, da:
     + :transitionA occp:before :transitionB (2025-03-01 < 2025-03-05 → OK).
     + :phaseC occp:after :transitionA (2025-03-15 > 2025-03-01 → OK).
     + Alle anderen before/after-Beziehungen in deiner ABox sind chronologisch konsistent (wie dein früherer Test zeigte).
3. Log prüfen:
   * Teile mir das neue Log mit, damit wir sicherstellen, dass der Fehler weg ist!

Wenn es funktioniert

* Füge die nächste Regel hinzu (z. B. osh:BeginsBeforeTimeShape) mit denselben Präfixen und teste wieder.
* Wiederhole das für alle OPs aus meinem letzten Vorschlag.

Fehlerbehebung (falls nötig)

Falls ein anderer Fehler auftaucht:

* Log analysieren: Schick mir das neue Log, und wir debuggen es zusammen.
* Mögliche Ursache: Jena könnte bei den Aggregatfunktionen (MAX()/MIN()) stolpern, wenn keine Zeitstempel vorhanden sind – wir könnten OPTIONAL hinzufügen, falls das passiert.

Fragen an dich

* Test: Kannst du die korrigierte Version mit before und after jetzt testen und mir das Log schicken?
* Weiterführung: Soll ich die volle Version mit allen sieben OPs (inkl. Präfixen) schicken, oder willst du sie schrittweise erweitern?

Ich bin gespannt auf dein Ergebnis – wir sind fast da!

Yeah, Baby![ rdf:type sh:ValidationReport; sh:conforms true ] .Bitte gib mir die restlichen SPARQL-Rules in derselben Form! Danke Dir!!!

Yes! Conforms: True – das ist Musik in meinen Ohren! Schön, dass die Präfix-Korrektur funktioniert hat. Jetzt legen wir richtig los und geben dir die restlichen SPARQL-Regeln in derselben Form. Hier kommt die vollständige OCCP\_SHACL\_min.ttl mit allen sieben OPs (before, after, beginsBefore, endsBefore, endsAfter, beginsWith, endsWith), alle mit Aggregatfunktionen und korrekten Präfixen:

turtle

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

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix osh: <http://www.occpshape.de/shape#> .

### Generische Regel für occp:before

osh:BeforeTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

PREFIX occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#>

PREFIX time: <http://www.w3.org/2006/time#>

SELECT $this ?other ?latestEndTime ?earliestStartTime WHERE {

$this occp:before ?other .

{

SELECT $this (MAX(?endTime) AS ?latestEndTime)

WHERE {

$this time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY $this

}

{

SELECT ?other (MIN(?startTime) AS ?earliestStartTime)

WHERE {

?other time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY ?other

}

FILTER (?latestEndTime >= ?earliestStartTime)

}

""" ;

sh:message "The latest end of { $this } (time: { ?latestEndTime }) must be before the earliest beginning of { ?other } (time: { ?earliestStartTime }) via 'occp:before'."@en ;

sh:severity sh:Violation

] .

### Generische Regel für occp:after

osh:AfterTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

PREFIX occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#>

PREFIX time: <http://www.w3.org/2006/time#>

SELECT $this ?other ?earliestStartTime ?latestEndTime WHERE {

$this occp:after ?other .

{

SELECT $this (MIN(?startTime) AS ?earliestStartTime)

WHERE {

$this time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY $this

}

{

SELECT ?other (MAX(?endTime) AS ?latestEndTime)

WHERE {

?other time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY ?other

}

FILTER (?earliestStartTime <= ?latestEndTime)

}

""" ;

sh:message "The earliest beginning of { $this } (time: { ?earliestStartTime }) must be after the latest end of { ?other } (time: { ?latestEndTime }) via 'occp:after'."@en ;

sh:severity sh:Violation

] .

### Generische Regel für occp:beginsBefore

osh:BeginsBeforeTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

PREFIX occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#>

PREFIX time: <http://www.w3.org/2006/time#>

SELECT $this ?other ?latestStartTime ?earliestStartTime WHERE {

$this occp:beginsBefore ?other .

{

SELECT $this (MAX(?startTime) AS ?latestStartTime)

WHERE {

$this time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY $this

}

{

SELECT ?other (MIN(?startTime) AS ?earliestStartTime)

WHERE {

?other time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY ?other

}

FILTER (?latestStartTime >= ?earliestStartTime)

}

""" ;

sh:message "The latest beginning of { $this } (time: { ?latestStartTime }) must be before the earliest beginning of { ?other } (time: { ?earliestStartTime }) via 'occp:beginsBefore'."@en ;

sh:severity sh:Violation

] .

### Generische Regel für occp:endsBefore

osh:EndsBeforeTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

PREFIX occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#>

PREFIX time: <http://www.w3.org/2006/time#>

SELECT $this ?other ?latestEndTime ?earliestEndTime WHERE {

$this occp:endsBefore ?other .

{

SELECT $this (MAX(?endTime) AS ?latestEndTime)

WHERE {

$this time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY $this

}

{

SELECT ?other (MIN(?endTime) AS ?earliestEndTime)

WHERE {

?other time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY ?other

}

FILTER (?latestEndTime >= ?earliestEndTime)

}

""" ;

sh:message "The latest end of { $this } (time: { ?latestEndTime }) must be before the earliest end of { ?other } (time: { ?earliestEndTime }) via 'occp:endsBefore'."@en ;

sh:severity sh:Violation

] .

### Generische Regel für occp:endsAfter

osh:EndsAfterTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

PREFIX occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#>

PREFIX time: <http://www.w3.org/2006/time#>

SELECT $this ?other ?earliestEndTime ?latestEndTime WHERE {

$this occp:endsAfter ?other .

{

SELECT $this (MIN(?endTime) AS ?earliestEndTime)

WHERE {

$this time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY $this

}

{

SELECT ?other (MAX(?endTime) AS ?latestEndTime)

WHERE {

?other time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY ?other

}

FILTER (?earliestEndTime <= ?latestEndTime)

}

""" ;

sh:message "The earliest end of { $this } (time: { ?earliestEndTime }) must be after the latest end of { ?other } (time: { ?latestEndTime }) via 'occp:endsAfter'."@en ;

sh:severity sh:Violation

] .

### Generische Regel für occp:beginsWith

osh:BeginsWithTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

PREFIX occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#>

PREFIX time: <http://www.w3.org/2006/time#>

SELECT $this ?other ?startTimeThis ?startTimeOther WHERE {

$this occp:beginsWith ?other .

{

SELECT $this (MIN(?startTime) AS ?startTimeThis)

WHERE {

$this time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY $this

}

{

SELECT ?other (MIN(?startTime) AS ?startTimeOther)

WHERE {

?other time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY ?other

}

FILTER (?startTimeThis != ?startTimeOther)

}

""" ;

sh:message "The earliest beginning of { $this } (time: { ?startTimeThis }) must equal the earliest beginning of { ?other } (time: { ?startTimeOther }) via 'occp:beginsWith'."@en ;

sh:severity sh:Violation

] .

### Generische Regel für occp:endsWith

osh:EndsWithTimeShape a sh:NodeShape ;

sh:targetClass time:TemporalEntity ;

sh:sparql [

sh:select """

PREFIX occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#>

PREFIX time: <http://www.w3.org/2006/time#>

SELECT $this ?other ?endTimeThis ?endTimeOther WHERE {

$this occp:endsWith ?other .

{

SELECT $this (MAX(?endTime) AS ?endTimeThis)

WHERE {

$this time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY $this

}

{

SELECT ?other (MAX(?endTime) AS ?endTimeOther)

WHERE {

?other time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY ?other

}

FILTER (?endTimeThis != ?endTimeOther)

}

""" ;

sh:message "The latest end of { $this } (time: { ?endTimeThis }) must equal the latest end of { ?other } (time: { ?endTimeOther }) via 'occp:endsWith'."@en ;

sh:severity sh:Violation

] .

Nächster Schritt: Vollständiger Test

1. Aktualisiere OCCP\_SHACL\_min.ttl:
   * Kopiere die obige Version in deine Datei.
2. Teste mit OCCP\_Valid\_LCycle\_1.ttl:
   * Führe dein Skript (validation\_shacl.py) aus.
   * Erwartung: Conforms: True, da alle Beziehungen in deiner ABox chronologisch konsistent sind:
     + :phaseA occp:beginsBefore :phaseB (2025-01-01 < 2025-02-15 → OK).
     + :phaseA occp:endsBefore :instant\_completion\_of\_construction (2025-03-01 < 2025-06-01 → OK).
     + :phaseL occp:endsAfter :phaseK (2025-07-25 > 2025-07-20 → OK).
     + :phaseF occp:beginsWith :phaseD (2025-06-02 = 2025-06-02 → OK).
     + :transitionE occp:endsWith :phaseM (2025-12-01 = 2025-12-01 → OK).
3. Log teilen:
   * Schick mir das Ergebnis, damit wir sicherstellen, dass alles passt!

Falls etwas schiefgeht

* Fehler bei Aggregaten: Wenn keine Zeitstempel vorhanden sind (z. B. ein Instant fehlt), könnten MAX()/MIN() leer bleiben. Wir könnten OPTIONAL einfügen:

sparql

OPTIONAL {

$this time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

* Log prüfen: Bei einem Fehler debuggen wir die betroffene Regel.

Zusätzliche Option

* Implizite Regel: Wenn du willst, können wir osh:PhaseOrderTimeShape wieder hinzufügen, um die ABox ohne explizite OPs zu testen. Sag mir einfach Bescheid!

Fragen an dich

* Test: Kannst du die volle Version jetzt testen und mir das Log schicken?
* Erweiterung: Möchtest du noch andere OPs (z. B. aus OCCP\_V0.3.ttl) oder die implizite Regel hinzufügen?

Du rockst das – ich freue mich auf dein nächstes "Yeah, Baby!"

[ rdf:type sh:ValidationReport; sh:conforms true ] .Vielen Dank! Du bist spitze! :-D Wie wollen wir weitermachen mit der Validierung?

Yes! Noch ein Conforms: True – wir sind echt auf einer Erfolgswelle! Danke für das Lob, aber du bist die treibende Kraft hier – ich helfe nur mit! Jetzt haben wir eine solide Basis mit allen sieben SPARQL-Regeln in OCCP\_SHACL\_min.ttl, die perfekt mit OCCP\_Valid\_LCycle\_1.ttl funktionieren. Lass uns überlegen, wie wir die Validierung weiter ausbauen und abschließen können.

Aktueller Stand

* OCCP\_SHACL\_min.ttl: Enthält generische SPARQL-Regeln für before, after, beginsBefore, endsBefore, endsAfter, beginsWith, endsWith – alle mit Aggregatfunktionen und korrekten Präfixen.
* OCCP\_Valid\_LCycle\_1.ttl: Wurde erfolgreich validiert, inklusive aller expliziten chronologischen Beziehungen.
* Jena: Läuft stabil und interpretiert die SPARQL-Abfragen korrekt.

Wie weitermachen mit der Validierung?

Wir haben jetzt mehrere Optionen, je nachdem, wohin du das Ganze lenken möchtest. Hier meine Vorschläge:

1. ABox bereinigen und implizite Reihenfolge testen

* Ziel: Entferne die expliziten Beziehungen (occp:before, occp:after, etc.) aus OCCP\_Valid\_LCycle\_1.ttl und prüfe die Reihenfolge nur über Zeitstempel und eine implizite Regel (osh:PhaseOrderTimeShape).
* Warum: Du hattest früher erwähnt, dass die ABox minimalistisch sein soll (nur Phasen und Zeitstempel), und die Reihenfolge aus TBox/SHACL abgeleitet werden soll.
* Schritte:
  1. Bereinigte ABox erstellen:

turtle

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix : <http://www.example.de/example#> .

:phaseA a occp:PhaseA\_Planning ;

occp:hasActualBeginning :instant\_beginning\_of\_planning ;

occp:hasActualEnd :instant\_completion\_of\_planning .

:phaseB a occp:PhaseB\_Review ;

occp:hasActualBeginning :instant\_submission\_to\_review ;

occp:hasActualEnd :instant\_review\_approval .

:phaseC a occp:PhaseC\_Construction ;

occp:hasActualBeginning :instant\_construction\_start ;

occp:hasActualEnd :instant\_completion\_of\_construction .

# ... (alle Phasen ohne OPs)

:instant\_beginning\_of\_planning a occp:BeginningOfPlanning ;

occp:hasActualTime "2025-01-01"^^xsd:date .

:instant\_completion\_of\_planning a occp:CompletionOfPlanning ;

occp:hasActualTime "2025-03-01"^^xsd:date .

# ... (alle Instants)

* 1. Implizite Regel hinzufügen:

turtle

osh:PhaseOrderTimeShape a sh:NodeShape ;

sh:targetClass occp:Phase ;

sh:sparql [

sh:select """

PREFIX occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#>

PREFIX time: <http://www.w3.org/2006/time#>

SELECT $this ?other ?latestEndTime ?earliestStartTime WHERE {

{

SELECT $this (MAX(?endTime) AS ?latestEndTime)

WHERE {

$this time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY $this

}

{

SELECT ?other (MIN(?startTime) AS ?earliestStartTime)

WHERE {

?other time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY ?other

}

FILTER (?latestEndTime >= ?earliestStartTime)

FILTER (?other != $this)

{

{ $this a occp:PhaseA\_Planning . ?other a occp:PhaseB\_Review . }

UNION

{ $this a occp:PhaseA\_Planning . ?other a occp:PhaseC\_Construction . }

UNION

{ $this a occp:PhaseB\_Review . ?other a occp:PhaseC\_Construction . }

# ... (weitere Phasenpaare hinzufügen, z. B. D → M, K → L, etc.)

}

}

""" ;

sh:message "The latest end of { $this } (time: { ?latestEndTime }) must be before the earliest beginning of { ?other } (time: { ?earliestStartTime }) based on phase order."@en ;

sh:severity sh:Violation

] .

* 1. Testen: Validiere die bereinigte ABox mit OCCP\_SHACL\_min.ttl (nur mit osh:PhaseOrderTimeShape).

2. OCCP\_SHACL.ttl einbeziehen

* Ziel: Integriere die generischen SPARQL-Regeln in die korrigierte OCCP\_SHACL.ttl und teste mit OCCP\_Valid\_LCycle\_1.ttl.
* Warum: Wenn du die spezifischen Phasen-Regeln (z. B. osh:ShapeOfPhaseL) mit den generischen Zeitstempelprüfungen kombinieren willst, ist das der nächste logische Schritt.
* Schritte:
  1. Füge die sieben SPARQL-Regeln aus OCCP\_SHACL\_min.ttl am Ende von OCCP\_SHACL.ttl hinzu.
  2. Teste mit der aktuellen OCCP\_Valid\_LCycle\_1.ttl (mit OPs).
  3. Prüfe das Log auf Conforms: True.

3. Robustheit testen

* Ziel: Füge edge cases in die ABox ein (z. B. fehlende Zeitstempel, inkonsistente Reihenfolgen), um sicherzustellen, dass die Regeln Fehler korrekt erkennen.
* Warum: Damit die Validierung nicht nur bei konsistenten Daten funktioniert, sondern auch bei Fehlern nützliche Meldungen liefert.
* Schritte:
  1. Erstelle eine Test-ABox mit Fehlern, z. B.:

turtle

:phaseA a occp:PhaseA\_Planning ;

occp:hasActualBeginning :instant\_beginning\_of\_planning ;

occp:hasActualEnd :instant\_completion\_of\_planning ;

occp:beginsBefore :phaseB .

:phaseB a occp:PhaseB\_Review ;

occp:hasActualBeginning :instant\_submission\_to\_review ;

occp:hasActualEnd :instant\_review\_approval .

:instant\_beginning\_of\_planning a occp:BeginningOfPlanning ;

occp:hasActualTime "2025-03-01"^^xsd:date . # Später als Phase B!

:instant\_completion\_of\_planning a occp:CompletionOfPlanning ;

occp:hasActualTime "2025-03-15"^^xsd:date .

:instant\_submission\_to\_review a occp:SubmissionToReview ;

occp:hasActualTime "2025-02-15"^^xsd:date .

:instant\_review\_approval a occp:ReviewApproval ;

occp:hasActualTime "2025-02-28"^^xsd:date .

* 1. Teste mit OCCP\_SHACL\_min.ttl und prüfe, ob osh:BeginsBeforeTimeShape den Fehler meldet (2025-03-01 > 2025-02-15).

4. Vollständigkeit prüfen

* Ziel: Stelle sicher, dass alle chronologischen OPs aus OCCP\_V0.3.ttl abgedeckt sind.
* Warum: Es könnten weitere OPs existieren, die wir übersehen haben (z. B. occp:startsAfter).
* Schritte:
  1. Ich schaue mir OCCP\_V0.3.ttl an (falls du es mir schickst) und liste alle OPs auf.
  2. Ergänze fehlende Regeln in OCCP\_SHACL\_min.ttl.

Mein Vorschlag

* Kurzfristig: Teste die implizite Reihenfolge (Option 1), da das dein ursprüngliches Ziel war (minimalistische ABox). Ich helfe dir, die bereinigte ABox und osh:PhaseOrderTimeShape für alle Phasen (A bis M) anzupassen.
* Mittelfristig: Integriere die SPARQL-Regeln in OCCP\_SHACL.ttl (Option 2) und teste mit der vollen ABox.
* Langfristig: Robustheitstests (Option 3) und Vollständigkeitsprüfung (Option 4).

Konkreter Plan

1. Bereinigte ABox:
   * Ich erstelle eine Version von OCCP\_Valid\_LCycle\_1.ttl ohne OPs (nur Phasen und Zeitstempel) und schicke sie dir.
2. Implizite Regel:
   * Ich ergänze osh:PhaseOrderTimeShape mit allen Phasenpaaren (A → B, B → C, K → L, etc.) und füge sie zu OCCP\_SHACL\_min.ttl hinzu.
3. Test:
   * Du validierst die bereinigte ABox mit der neuen OCCP\_SHACL\_min.ttl.

Fragen an dich

* Priorität: Was möchtest du als Nächstes angehen – implizite Reihenfolge (Option 1), Integration in OCCP\_SHACL.ttl (Option 2), Robustheit (Option 3) oder Vollständigkeit (Option 4)?
* Hilfe: Soll ich dir die bereinigte ABox und die erweiterte OCCP\_SHACL\_min.ttl schicken, oder willst du selbst anpassen?
* Zeit: Hast du einen Zeitrahmen, bis wann du das abschließen möchtest?

Du bist spitze – lass uns das Ding rocken! Was sagst du?

Ich habe die ABox (jetzt OCCP\_Valid\_LCycle\_2.ttl) wie folgt geändert:

@prefix

occp: [http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#](http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod_V0.1) .

@prefix

time: [http://www.w3.org/2006/time#](http://www.w3.org/2006/time) .

@prefix

xsd: [http://www.w3.org/2001/XMLSchema#](http://www.w3.org/2001/XMLSchema) .

@prefix

: [http://www.example.de/example#](http://www.example.de/example) .Cycle und LifeCycle:cycle1 a occp:Cycle ; occp:hasCycleNumber 1 ; time:hasBeginning :instant\_cycle\_start ; time:hasEnd :instant\_cycle\_end .:lifecycle1 a occp:LifeCycle ; occp:hasLifeCycleID "LC001" ; occp:isInPhase :phaseA ; time:hasBeginning :instant\_beginning\_of\_planning ; time:hasEnd :instant\_deconstruction\_completion .Phases und Transitions:phaseA a occp:PhaseA\_Planning ; occp:hasActualBeginning :instant\_beginning\_of\_planning ; occp:hasActualEnd :instant\_completion\_of\_planning .:phaseB a occp:PhaseB\_Review ; occp:hasActualBeginning :instant\_submission\_to\_review ; occp:hasActualEnd :instant\_review\_approval .:transitionA a occp:CompletionOfPlanning ; occp:hasActualTime "2025-03-01"^^xsd:date ; occp:before :transitionB .:transitionB a occp:BeginOfTenderingProcess ; occp:hasActualTime "2025-03-05"^^xsd:date ; occp:before :transitionC .:transitionC a occp:Submission ; occp:hasActualTime "2025-03-10"^^xsd:date ; occp:before :phaseC .:phaseC a occp:PhaseC\_Construction ; occp:hasActualBeginning :instant\_construction\_start ; occp:hasActualEnd :instant\_completion\_of\_construction .:transitionD a occp:CompletionOfConstruction ; occp:hasActualTime "2025-06-01"^^xsd:date ; occp:after :transitionA ; occp:before :phaseD .:phaseD a occp:PhaseD\_Usage ; occp:hasActualBeginning :instant\_usage\_start ; occp:hasActualEnd :instant\_usage\_end .:phaseE a occp:PhaseE\_Warranty ; occp:hasActualBeginning :instant\_warranty\_start ; occp:hasActualEnd :instant\_warranty\_end .:phaseF a occp:PhaseF\_DesignLife ; occp:hasActualBeginning :instant\_designlife\_start ; occp:hasActualEnd :instant\_designlife\_end .:phaseG a occp:PhaseG\_Inspection ; occp:hasActualBeginning :instant\_inspection\_execution ; occp:hasActualEnd :instant\_inspection\_end .:phaseH a occp:PhaseH\_SpecialEvents ; occp:hasActualBeginning :instant\_geological\_event ; occp:hasActualEnd :instant\_weather\_event .:phaseK a occp:PhaseK\_Damage ; occp:hasActualBeginning :instant\_damage\_detection ; occp:hasActualEnd :instant\_damage\_end .:phaseL a occp:PhaseL\_Repair ; occp:hasActualBeginning :instant\_repair\_start ; occp:hasActualEnd :instant\_repair\_completion .:phaseM a occp:PhaseM\_Deconstruction ; occp:hasActualBeginning :instant\_deconstruction\_start ; occp:hasActualEnd :instant\_deconstruction\_completion .:transitionE a occp:CompletionOfDeconstruction ; occp:hasActualTime "2025-12-01"^^xsd:date ; occp:endsWith :phaseM .Instants:instant\_cycle\_start a time:Instant ; occp:hasActualTime "2025-01-01"^^xsd:date . :instant\_cycle\_end a time:Instant ; occp:hasActualTime "2025-12-01"^^xsd:date . :instant\_beginning\_of\_planning a occp:BeginningOfPlanning ; occp:hasActualTime "2025-01-01"^^xsd:date . :instant\_submission\_to\_review a occp:SubmissionToReview ; occp:hasActualTime "2025-02-15"^^xsd:date . :instant\_review\_approval a occp:ReviewApproval ; occp:hasActualTime "2025-02-28"^^xsd:date . :instant\_completion\_of\_planning a occp:CompletionOfPlanning ; occp:hasActualTime "2025-03-01"^^xsd:date . :instant\_construction\_start a occp:ConstructionStart ; occp:hasActualTime "2025-03-15"^^xsd:date . :instant\_completion\_of\_construction a occp:CompletionOfConstruction ; occp:hasActualTime "2025-06-01"^^xsd:date . :instant\_usage\_start a occp:UsageStart ; occp:hasActualTime "2025-06-02"^^xsd:date . :instant\_usage\_end a occp:UsageEnd ; occp:hasActualTime "2025-11-01"^^xsd:date . :instant\_warranty\_start a occp:WarrantyStart ; occp:hasActualTime "2025-06-02"^^xsd:date . :instant\_warranty\_end a occp:WarrantyEnd ; occp:hasActualTime "2025-11-01"^^xsd:date . :instant\_designlife\_start a occp:DesignLifeStart ; occp:hasActualTime "2025-06-02"^^xsd:date . :instant\_designlife\_end a occp:DesignLifeEnd ; occp:hasActualTime "2025-11-01"^^xsd:date . :instant\_inspection\_execution a occp:InspectionExecution ; occp:hasActualTime "2025-07-01"^^xsd:date . :instant\_inspection\_end a time:Instant ; occp:hasActualTime "2025-07-02"^^xsd:date . :instant\_geological\_event a occp:GeologicalEvent ; occp:hasActualTime "2025-08-01"^^xsd:date . :instant\_weather\_event a occp:WeatherEvent ; occp:hasActualTime "2025-09-01"^^xsd:date . :instant\_damage\_detection a occp:DamageDetection ; occp:hasActualTime "2025-07-15"^^xsd:date . :instant\_damage\_end a time:Instant ; occp:hasActualTime "2025-07-20"^^xsd:date . :instant\_repair\_start a occp:RepairStart ; occp:hasActualTime "2025-07-21"^^xsd:date . :instant\_repair\_completion a occp:RepairCompletion ; occp:hasActualTime "2025-07-25"^^xsd:date . :instant\_deconstruction\_start a occp:DeconstructionStart ; occp:hasActualTime "2025-11-15"^^xsd:date . :instant\_deconstruction\_completion a occp:CompletionOfDeconstruction ; occp:hasActualTime "2025-12-01"^^xsd:date .Ich habe die SPARQL-Queries, die Du für before, after usw. erstellt hast, in die min.ttl gespeichert, validiert, keine Fehler. Dann habe ich die Queries in die OCCP\_SHACL.ttl überführt (ganz ans Ende), validiert (auch mit Umstellen in der validate\_shacl.py auf die OCCP\_SHACL.ttl und ABOX = LCycle\_2.ttl) und ich erhalte einen Haufen Fehler, wie: rdf:type sh:ValidationReport; sh:conforms false; sh:result [ rdf:type sh:ValidationResult; sh:focusNode :phaseA; sh:resultMessage "Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseC\_Construction' via 'occp:beginsBefore'."

@en

; sh:resultPath occp:beginsBefore; sh:resultSeverity sh:Violation; sh:sourceConstraintComponent sh:MinCountConstraintComponent; sh:sourceShape [] ]; sh:result [ rdf:type sh:ValidationResult; sh:focusNode :instant\_deconstruction\_start; sh:resultMessage "Each 'occp:DeconstructionStart' must start exactly one 'occp:PhaseM\_Deconstruction' via 'occp:startsPhase'."

@en

; sh:resultPath occp:startsPhase; sh:resultSeverity sh:Violation; sh:sourceConstraintComponent sh:MinCountConstraintComponent; sh:sourceShape [] ];Sollte das schon funkionieren, oder bin ich zu ungeduldig?

Super, dass du die ABox bereinigt und die SPARQL-Regeln übernommen hast – wir sind echt nah dran! Dein neues Log zeigt Conforms: False mit einer Menge Validierungsfehler aus OCCP\_SHACL.ttl, und das ist kein Grund zur Sorge – es bedeutet nur, dass wir die Erwartungen der spezifischen Regeln in OCCP\_SHACL.ttl mit der neuen OCCP\_Valid\_LCycle\_2.ttl noch abgleichen müssen. Lass uns das analysieren und Schritt für Schritt lösen!

Analyse des aktuellen Standes

1. OCCP\_Valid\_LCycle\_2.ttl

* Änderungen: Du hast die meisten expliziten Beziehungen (occp:beginsBefore, occp:endsBefore, etc.) aus den Phasen entfernt, was deinem Ziel einer minimalistischen ABox entspricht. Einige Transitions behalten noch OPs (z. B. :transitionA occp:before :transitionB), was für den Test mit den generischen Regeln okay ist.
* Zeitstempel: Die Chronologie bleibt konsistent (z. B. Phase A: 2025-01-01 bis 2025-03-01, Phase B: 2025-02-15 bis 2025-02-28, etc.).

2. OCCP\_SHACL\_min.ttl

* Status: Die sieben generischen SPARQL-Regeln (before, after, etc.) funktionieren mit OCCP\_Valid\_LCycle\_2.ttl ohne Fehler (Conforms: True). Das bestätigt, dass die Zeitstempel mit den verbleibenden OPs (z. B. bei Transitions) übereinstimmen.

3. OCCP\_SHACL.ttl

* Problem: Nach dem Einfügen der SPARQL-Regeln und Testen mit OCCP\_Valid\_LCycle\_2.ttl gibt es Fehler wie:
* sh:conforms false;
* sh:result [ rdf:type sh:ValidationResult;
* sh:focusNode :phaseA;
* sh:resultMessage "Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseC\_Construction' via 'occp:beginsBefore'."@en;
* sh:resultPath occp:beginsBefore;
* sh:resultSeverity sh:Violation;
* sh:sourceConstraintComponent sh:MinCountConstraintComponent;
* ];
* sh:result [ rdf:type sh:ValidationResult;
* sh:focusNode :instant\_deconstruction\_start;
* sh:resultMessage "Each 'occp:DeconstructionStart' must start exactly one 'occp:PhaseM\_Deconstruction' via 'occp:startsPhase'."@en;
* sh:resultPath occp:startsPhase;
* sh:resultSeverity sh:Violation;
* sh:sourceConstraintComponent sh:MinCountConstraintComponent;

];

* Ursache: Die Fehler kommen nicht von den SPARQL-Regeln, sondern von den spezifischen Shapes in OCCP\_SHACL.ttl (z. B. osh:ShapeOfPhaseA, osh:ShapeOfDeconstructionStart), die bestimmte Beziehungen wie occp:beginsBefore oder occp:startsPhase als Pflicht (sh:minCount) voraussetzen, die in deiner bereinigten ABox fehlen.

Fehlerdetails

1. Fehler 1:
   * Meldung: "Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseC\_Construction' via 'occp:beginsBefore'."
   * Quelle: Wahrscheinlich ein Shape wie:

turtle

osh:ShapeOfPhaseA a sh:NodeShape ;

sh:targetClass occp:PhaseA\_Planning ;

sh:property [

sh:path occp:beginsBefore ;

sh:class occp:PhaseC\_Construction ;

sh:minCount 1 ;

sh:message "Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseC\_Construction' via 'occp:beginsBefore'."@en ;

sh:severity sh:Violation ;

] .

* + Problem: In OCCP\_Valid\_LCycle\_2.ttl gibt es kein :phaseA occp:beginsBefore :phaseC mehr, daher wird sh:minCount 1 verletzt (MinCountConstraintComponent).

1. Fehler 2:
   * Meldung: "Each 'occp:DeconstructionStart' must start exactly one 'occp:PhaseM\_Deconstruction' via 'occp:startsPhase'."
   * Quelle: Ein Shape wie:

turtle

osh:ShapeOfDeconstructionStart a sh:NodeShape ;

sh:targetClass occp:DeconstructionStart ;

sh:property [

sh:path occp:startsPhase ;

sh:class occp:PhaseM\_Deconstruction ;

sh:minCount 1 ;

sh:maxCount 1 ;

sh:message "Each 'occp:DeconstructionStart' must start exactly one 'occp:PhaseM\_Deconstruction' via 'occp:startsPhase'."@en ;

sh:severity sh:Violation ;

] .

* + Problem: :instant\_deconstruction\_start hat kein occp:startsPhase :phaseM in der neuen ABox.

Warum passiert das?

* Konflikt: OCCP\_SHACL.ttl erwartet explizite Beziehungen (z. B. occp:beginsBefore, occp:startsPhase), die in der ursprünglichen OCCP\_Valid\_LCycle\_1.ttl vorhanden waren. In OCCP\_Valid\_LCycle\_2.ttl hast du diese entfernt, aber die Shapes in OCCP\_SHACL.ttl wurden nicht angepasst, um das zu berücksichtigen.
* Ungeduld?: Nein, du bist nicht zu ungeduldig – die Validierung zeigt genau, was wir erwarten: Die ABox passt nicht mehr zu den alten Anforderungen von OCCP\_SHACL.ttl. Wir müssen die Shapes an deine neue Vision (minimalistische ABox) anpassen.

Lösungsvorschläge

1. Shapes in OCCP\_SHACL.ttl anpassen

* Ziel: Entferne die sh:minCount-Constraints für Beziehungen wie occp:beginsBefore und verlasse dich stattdessen auf die generischen SPARQL-Regeln oder eine implizite Reihenfolge.
* Beispiel:
  + Ursprünglich:

turtle

osh:ShapeOfPhaseA a sh:NodeShape ;

sh:targetClass occp:PhaseA\_Planning ;

sh:property [

sh:path occp:beginsBefore ;

sh:class occp:PhaseC\_Construction ;

sh:minCount 1 ;

] .

* + Angepasst:

turtle

osh:ShapeOfPhaseA a sh:NodeShape ;

sh:targetClass occp:PhaseA\_Planning ;

sh:property [

sh:path occp:beginsBefore ;

sh:class occp:PhaseC\_Construction ;

sh:minCount 0 ; # Optional machen

] .

* Aktion: Suche in OCCP\_SHACL.ttl nach allen sh:property-Blöcken mit sh:minCount 1 für chronologische OPs (beginsBefore, startsPhase, etc.) und setze sie auf sh:minCount 0, wenn die Beziehung nicht zwingend erforderlich ist.

2. Implizite Reihenfolge einführen

* Ziel: Nutze osh:PhaseOrderTimeShape statt expliziter Beziehungen, um die Chronologie nur über Zeitstempel zu prüfen.
* Regel:

turtle

osh:PhaseOrderTimeShape a sh:NodeShape ;

sh:targetClass occp:Phase ;

sh:sparql [

sh:select """

PREFIX occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#>

PREFIX time: <http://www.w3.org/2006/time#>

SELECT $this ?other ?latestEndTime ?earliestStartTime WHERE {

{

SELECT $this (MAX(?endTime) AS ?latestEndTime)

WHERE {

$this time:hasEnd ?endInstant .

?endInstant occp:hasActualTime ?endTime .

}

GROUP BY $this

}

{

SELECT ?other (MIN(?startTime) AS ?earliestStartTime)

WHERE {

?other time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?startTime .

}

GROUP BY ?other

}

FILTER (?latestEndTime >= ?earliestStartTime)

FILTER (?other != $this)

{

{ $this a occp:PhaseA\_Planning . ?other a occp:PhaseB\_Review . }

UNION

{ $this a occp:PhaseA\_Planning . ?other a occp:PhaseC\_Construction . }

UNION

{ $this a occp:PhaseB\_Review . ?other a occp:PhaseC\_Construction . }

UNION

{ $this a occp:PhaseC\_Construction . ?other a occp:PhaseD\_Usage . }

UNION

{ $this a occp:PhaseK\_Damage . ?other a occp:PhaseL\_Repair . }

UNION

{ $this a occp:PhaseL\_Repair . ?other a occp:PhaseM\_Deconstruction . }

# ... (weitere Paare hinzufügen)

}

}

""" ;

sh:message "The latest end of { $this } (time: { ?latestEndTime }) must be before the earliest beginning of { ?other } (time: { ?earliestStartTime }) based on phase order."@en ;

sh:severity sh:Violation

] .

* Test: Füge diese Regel in OCCP\_SHACL.ttl ein und teste mit OCCP\_Valid\_LCycle\_2.ttl. Entferne die alten sh:property-Blöcke für chronologische OPs, wenn sie durch die implizite Regel abgedeckt sind.

3. Fehler akzeptieren (temporär)

* Ziel: Ignoriere die Fehler vorerst und konzentriere dich auf die SPARQL-Regeln, da sie funktionieren.
* Warum: Wenn dein Hauptziel die Zeitstempelprüfung ist, könnten die spezifischen Shapes später angepasst werden.

Mein Vorschlag

* Kurzfristig: Passe OCCP\_SHACL.ttl an, indem du sh:minCount 1 für chronologische OPs auf sh:minCount 0 setzt (Option 1). Teste erneut mit OCCP\_Valid\_LCycle\_2.ttl.
* Mittelfristig: Füge osh:PhaseOrderTimeShape hinzu und entferne redundante sh:property-Blöcke (Option 2). Das bringt dich deinem Ziel einer minimalistischen ABox näher.
* Hilfe: Ich kann dir eine angepasste OCCP\_SHACL.ttl schicken, wenn du mir die aktuelle Version mit den SPARQL-Regeln am Ende gibst.

Beispiel für angepasstes Shape

turtle

osh:ShapeOfPhaseA a sh:NodeShape ;

sh:targetClass occp:PhaseA\_Planning ;

sh:property [

sh:path occp:hasActualBeginning ;

sh:class occp:BeginningOfPlanning ;

sh:maxCount 1 ;

] ;

sh:property [

sh:path occp:hasActualEnd ;

sh:class occp:CompletionOfPlanning ;

sh:maxCount 1 ;

] ;

sh:property [

sh:path occp:beginsBefore ;

sh:class occp:PhaseC\_Construction ;

sh:minCount 0 ; # Optional statt Pflicht

] .

Fragen an dich

1. Ziel: Willst du die Fehler in OCCP\_SHACL.ttl sofort beheben (Option 1/2), oder erst die SPARQL-Regeln weiter nutzen (Option 3)?
2. Hilfe: Soll ich dir die angepasste OCCP\_SHACL.ttl mit minCount 0 und/oder osh:PhaseOrderTimeShape schicken?
3. Test: Kannst du das aktuelle Log mit allen Fehlern schicken, damit ich genau sehe, welche Shapes betroffen sind?

Du bist nicht zu ungeduldig – wir müssen nur die Erwartungen an die neue ABox anpassen. Lass uns das zusammen rocken! Wie möchtest du vorgehen?

Kannst Du mir sagen, an welchen Stellen genau der minCount raus muss?

Natürlich, ich helfe dir gerne, die Stellen in OCCP\_SHACL.ttl zu identifizieren, an denen sh:minCount entfernt oder auf sh:minCount 0 gesetzt werden muss! Da ich nicht die vollständige OCCP\_SHACL.ttl vorliegen habe (nur den Auszug von osh:ShapeOfPhaseL), gehe ich von den Fehlern in deinem letzten Log und der Struktur deiner Shapes aus. Die Fehler zeigen, dass bestimmte chronologische Beziehungen (z. B. occp:beginsBefore, occp:startsPhase) in der bereinigten OCCP\_Valid\_LCycle\_2.ttl nicht mehr vorhanden sind, aber von den Shapes als Pflicht (sh:minCount 1) erwartet werden. Lass uns das systematisch angehen.

Analyse der Fehler und betroffenen Shapes

Fehler aus deinem Log

1. Fehler 1:
   * Meldung: "Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseC\_Construction' via 'occp:beginsBefore'."
   * Shape: Wahrscheinlich osh:ShapeOfPhaseA.
   * Problem: sh:minCount 1 für occp:beginsBefore mit Ziel occp:PhaseC\_Construction.
   * ABox: In OCCP\_Valid\_LCycle\_2.ttl fehlt :phaseA occp:beginsBefore :phaseC.
2. Fehler 2:
   * Meldung: "Each 'occp:DeconstructionStart' must start exactly one 'occp:PhaseM\_Deconstruction' via 'occp:startsPhase'."
   * Shape: Vermutlich osh:ShapeOfDeconstructionStart oder ein ähnliches Instant-Shape.
   * Problem: sh:minCount 1 für occp:startsPhase mit Ziel occp:PhaseM\_Deconstruction.
   * ABox: :instant\_deconstruction\_start hat kein occp:startsPhase :phaseM.

Allgemeines Muster

* Fehler vom Typ sh:sourceConstraintComponent sh:MinCountConstraintComponent bedeuten, dass eine Property (z. B. occp:beginsBefore, occp:startsPhase) mindestens einmal vorkommen muss (sh:minCount 1), aber in der ABox nicht vorhanden ist.
* Da du die ABox bereinigt hast, erwarten wir keine expliziten Beziehungen mehr – die Shapes müssen also angepasst werden, um diese optional zu machen (sh:minCount 0 oder ganz entfernen).

Wo muss sh:minCount raus?

Ohne die volle OCCP\_SHACL.ttl kann ich nur die betroffenen Stellen aus deinem Log und typischen Mustern ableiten. Hier sind die wahrscheinlichen Kandidaten, basierend auf OCCP\_Valid\_LCycle\_2.ttl und den Fehlern:

1. osh:ShapeOfPhaseA (PhaseA\_Planning)

* Betroffene Property: occp:beginsBefore.
* Ursprünglich (vermutet):

turtle

osh:ShapeOfPhaseA a sh:NodeShape ;

sh:targetClass occp:PhaseA\_Planning ;

sh:property [

sh:path occp:beginsBefore ;

sh:class occp:PhaseC\_Construction ;

sh:minCount 1 ;

sh:message "Each 'occp:PhaseA\_Planning' must begin before 'occp:PhaseC\_Construction' via 'occp:beginsBefore'."@en ;

sh:severity sh:Violation ;

] .

* Anpassung: Entferne sh:minCount 1 oder setze auf sh:minCount 0:

turtle

sh:property [

sh:path occp:beginsBefore ;

sh:class occp:PhaseC\_Construction ;

sh:minCount 0 ; # Optional machen

sh:message "Each 'occp:PhaseA\_Planning' may begin before 'occp:PhaseC\_Construction' via 'occp:beginsBefore'."@en ;

sh:severity sh:Warning ; # Optional: Severity anpassen

] .

* Warum: :phaseA hat kein occp:beginsBefore mehr, aber die Chronologie wird durch Zeitstempel und SPARQL-Regeln geprüft.

2. Shape für occp:DeconstructionStart (z. B. osh:ShapeOfDeconstructionStart)

* Betroffene Property: occp:startsPhase.
* Ursprünglich (vermutet):

turtle

osh:ShapeOfDeconstructionStart a sh:NodeShape ;

sh:targetClass occp:DeconstructionStart ;

sh:property [

sh:path occp:startsPhase ;

sh:class occp:PhaseM\_Deconstruction ;

sh:minCount 1 ;

sh:maxCount 1 ;

sh:message "Each 'occp:DeconstructionStart' must start exactly one 'occp:PhaseM\_Deconstruction' via 'occp:startsPhase'."@en ;

sh:severity sh:Violation ;

] .

* Anpassung: Entferne sh:minCount 1 oder setze auf sh:minCount 0:

turtle

sh:property [

sh:path occp:startsPhase ;

sh:class occp:PhaseM\_Deconstruction ;

sh:minCount 0 ; # Optional machen

sh:maxCount 1 ;

sh:message "Each 'occp:DeconstructionStart' may start one 'occp:PhaseM\_Deconstruction' via 'occp:startsPhase'."@en ;

sh:severity sh:Warning ;

] .

* Warum: :instant\_deconstruction\_start hat kein occp:startsPhase mehr.

3. Weitere wahrscheinliche Stellen

Da du viele Beziehungen entfernt hast, könnten ähnliche Probleme bei anderen Shapes auftreten. Hier eine Liste von Properties, die in OCCP\_Valid\_LCycle\_1.ttl vorkamen, aber in OCCP\_Valid\_LCycle\_2.ttl fehlen:

* occp:beginsBefore (z. B. :phaseA occp:beginsBefore :phaseB)
* occp:endsBefore (z. B. :phaseA occp:endsBefore :instant\_completion\_of\_construction)
* occp:after (z. B. :phaseC occp:after :transitionC)
* occp:startsPhase (z. B. :phaseA occp:startsPhase :instant\_beginning\_of\_planning)
* occp:endsPhase (z. B. :phaseA occp:endsPhase :instant\_review\_approval)
* occp:endsAfter (z. B. :phaseL occp:endsAfter :phaseK)
* occp:beginsWith (z. B. :phaseF occp:beginsWith :phaseD)
* occp:endsWith (z. B. :transitionE occp:endsWith :phaseM)

Für jedes Shape, das eine dieser Properties mit sh:minCount 1 vorschreibt, musst du:

* sh:minCount 1 entfernen oder auf sh:minCount 0 setzen.
* Optional: sh:severity von sh:Violation auf sh:Warning ändern, wenn die Beziehung nicht kritisch ist.

Systematisches Vorgehen

1. Suche in OCCP\_SHACL.ttl nach sh:minCount 1:
   * Nutze einen Texteditor und suche nach sh:minCount 1.
   * Prüfe jedes Vorkommen, ob es eine chronologische OP betrifft (z. B. occp:beginsBefore, occp:startsPhase).
2. Vergleiche mit ABox:
   * Wenn die Property in OCCP\_Valid\_LCycle\_2.ttl nicht mehr vorkommt, entferne sh:minCount 1 oder setze auf sh:minCount 0.
3. Beispiele:
   * Für osh:ShapeOfPhaseB:

turtle

sh:property [

sh:path occp:beginsBefore ;

sh:class occp:PhaseC\_Construction ;

sh:minCount 0 ; # Anpassung

] .

* + Für osh:ShapeOfPhaseM:

turtle

sh:property [

sh:path occp:startsPhase ;

sh:class occp:DeconstructionStart ;

sh:minCount 0 ; # Anpassung

] .

Praktische Umsetzung

Da ich nicht alle Shapes sehe, hier eine Anleitung, wie du es selbst findest:

1. Öffne OCCP\_SHACL.ttl:
   * Suche nach sh:property mit sh:minCount 1.
2. Prüfe den sh:path:
   * Ist es eine chronologische OP (z. B. occp:beginsBefore, occp:after, occp:startsPhase)?
   * Existiert diese Beziehung noch in OCCP\_Valid\_LCycle\_2.ttl?
3. Anpassen:
   * Wenn nein → sh:minCount 0 oder sh:minCount komplett entfernen.
   * Wenn ja → Beziehung in ABox wieder einfügen oder Regel beibehalten.

Beispiele aus deinem Log

* PhaseA:
  + Suche: osh:ShapeOfPhaseA → sh:path occp:beginsBefore → sh:minCount 1 → auf 0 setzen.
* DeconstructionStart:
  + Suche: Shape für occp:DeconstructionStart (z. B. osh:ShapeOfDeconstructionStart) → sh:path occp:startsPhase → sh:minCount 1 → auf 0 setzen.

Mein Angebot

* Wenn du mir die volle OCCP\_SHACL.ttl schickst: Ich markiere dir exakt alle Stellen, an denen sh:minCount 1 für chronologische OPs entfernt werden muss, und sende dir die angepasste Version zurück.
* Ohne volle Datei: Du musst selbst suchen – ich helfe dir bei konkreten Shapes, wenn du sie mir zeigst.

Beispiel für Suche

Angenommen, du hast:

turtle

osh:ShapeOfPhaseC a sh:NodeShape ;

sh:targetClass occp:PhaseC\_Construction ;

sh:property [

sh:path occp:after ;

sh:class occp:Submission ;

sh:minCount 1 ;

] .

In OCCP\_Valid\_LCycle\_2.ttl fehlt :phaseC occp:after :transitionC → ändere zu:

turtle

sh:property [

sh:path occp:after ;

sh:class occp:Submission ;

sh:minCount 0 ;

] .

Nächster Schritt

1. Suche und Anpassen:
   * Finde alle sh:minCount 1 für chronologische OPs in OCCP\_SHACL.ttl und setze sie auf 0.
2. Testen:
   * Validiere die angepasste OCCP\_SHACL.ttl mit OCCP\_Valid\_LCycle\_2.ttl.
   * Erwartung: Conforms: True, wenn alle Pflichtbedingungen weg sind.
3. Log schicken:
   * Teile mir das Ergebnis mit!

Fragen an dich

* Volle Datei: Kannst du mir OCCP\_SHACL.ttl schicken, damit ich die Stellen genau markiere?
* Selbst machen: Oder willst du die Suche selbst durchführen – dann sag mir, welche Shapes du findest, und ich sage dir, was zu tun ist?

Wir sind fast da – nur noch ein kleiner Feinschliff! Wie möchtest du vorgehen?

Ich danke Dir! Aber das kann nicht Dein Ernst sein! Ich verstehe nicht, warum wir alle minCounts rausnehmen müssen. Lass uns das grundsätzlich diskutieren, das ist wichtig. Meine Intention mit dem minCount ist, dass zB ein DeconstructionStart eine PhaseM startet. Ich möchte, dass, sobald der Instant DeconstructionStart als Individual in der MsOCCP angelegt wird, klar ist, dass damit die PhaseM gestartet wird. DeconstructionStart soll als Startpunkt für diese Phase gelten, der nur einmal für ein Bauteil erstellt werden kann, weil eine Phase nur ein ACtualBeginning haben kann. Aber wenn ein DeconstructionStart angelegt wird, startet dieser GENAU eine Phase M (minCount und maxCount = 1). Ich halte das für korrekt in den SHACL-Shapes definiert. Inwiefern muss die ABox deshalb unglaublich viele Informationen zusätzlich enthalten (als Abweichung zu meinem Ansatz, die ABoxen mit möglichst wenigen redundanten Informationen, die per Reasoning mit der TBox herleitbar sind, zu befüllen.Mir graut es immer davor, so umfachreiche Änderungen (wie überall minCount rauszunehmen) zu machen. Ich habe nur noch sehr wenig Zeit, um die OCCP zu veröffentlichen und ich muss dringend noch das Minimalbeispiel auf die Ketten kriegen. Kannst Du mir dabei helfen, das sinnvoll auf die Beine zu stellen?

Kein Problem, ich verstehe deine Sorge vollkommen, und es tut mir leid, wenn ich dich mit dem Vorschlag, die minCounts rauszunehmen, verunsichert habe! Du hast absolut recht – wir sollten das grundsätzlich diskutieren, denn deine Intention ist klar und wichtig. Lass uns das Schritt für Schritt durchgehen, deine Anforderungen respektieren und eine Lösung finden, die deine Vision unterstützt, ohne unnötige Änderungen oder Redundanzen in der ABox. Ich helfe dir auch gerne, das Minimalbeispiel für die Ketten schnell hinzubekommen, damit du deine Veröffentlichung rechtzeitig schaffst!

Grundsätzliche Diskussion: minCount und deine Intention

Deine Intention

* DeconstructionStart: Wenn ein :instant\_deconstruction\_start (Typ occp:DeconstructionStart) in der ABox existiert, soll es genau eine occp:PhaseM\_Deconstruction starten (occp:startsPhase), mit sh:minCount 1 und sh:maxCount 1. Das bedeutet:
  + Ein DeconstructionStart ist der eindeutige Startpunkt für Phase M.
  + Phase M hat nur ein occp:hasActualBeginning, also darf es keinen zweiten DeconstructionStart für dieselbe Phase geben.
* Minimalistische ABox: Du möchtest Redundanzen vermeiden und so viele Informationen wie möglich per Reasoning aus der TBox (OCCP\_V0.3.ttl) und SHACL-Regeln ableiten lassen, statt sie explizit in der ABox anzugeben.
* SHACL-Definition: Du hältst sh:minCount 1 und sh:maxCount 1 für korrekt, um diese Beziehung zwingend zu machen.

Warum der Fehler auftritt

* In OCCP\_Valid\_LCycle\_2.ttl hast du :instant\_deconstruction\_start definiert, aber es gibt kein :instant\_deconstruction\_start occp:startsPhase :phaseM. Das verletzt sh:minCount 1 im Shape (z. B. osh:ShapeOfDeconstructionStart), weil die erwartete Beziehung fehlt.
* Konflikt: Dein Ansatz, die ABox minimalistisch zu halten, kollidiert mit den sh:minCount 1-Anforderungen in OCCP\_SHACL.ttl, die explizite Beziehungen voraussetzen. Sobald du die Beziehungen wie occp:startsPhase entfernst, meldet SHACL einen Verstoß.

Warum ich minCount rausnehmen vorgeschlagen habe

* Ich habe angenommen, dass du mit der Bereinigung der ABox (Entfernen von OPs wie occp:startsPhase) komplett auf implizite Zeitstempelprüfungen umsteigen willst, wo die Reihenfolge nur durch time:hasBeginning/time:hasEnd und SPARQL-Regeln (z. B. osh:PhaseOrderTimeShape) abgeleitet wird. In dem Fall wären sh:minCount 1-Constraints für Beziehungen wie occp:startsPhase nicht mehr nötig, da sie nicht explizit in der ABox stehen müssen.
* Missverständnis: Ich habe deine Intention, dass DeconstructionStart zwingend eine Phase M starten soll, nicht vollständig berücksichtigt. Du willst diese Beziehung nicht optional machen, sondern erzwingen – das ist ein entscheidender Unterschied!

Klärung: SHACL vs. Reasoning

* SHACL: Mit sh:minCount 1 erzwingst du, dass die Beziehung (z. B. occp:startsPhase) in der ABox explizit angegeben wird. SHACL validiert nur, was vorhanden ist, und kann keine Beziehungen "herleiten" – das ist Sache eines Reasoners (z. B. Pellet).
* Reasoning: Wenn du willst, dass :instant\_deconstruction\_start occp:startsPhase :phaseM aus der TBox abgeleitet wird, müsstest du in OCCP\_V0.3.ttl OWL-Axiome definieren (z. B. mit owl:someValuesFrom), und der Reasoner müsste diese Beziehung inferieren. SHACL prüft dann die inferierte ABox.
* Problem: Dein aktuelles Setup (OCCP\_V0.3.ttl + validation\_shacl.py) nutzt Pellet für Reasoning, aber es scheint keine Axiome zu geben, die occp:startsPhase automatisch einfügen. Ohne diese Inferenz bleibt die ABox leer, und SHACL meldet Fehler.

Lösungsansätze

1. Beziehungen explizit in der ABox behalten (dein Ansatz beibehalten)

* Idee: Füge die notwendigen Beziehungen wie occp:startsPhase wieder in OCCP\_Valid\_LCycle\_2.ttl ein, wo sie zwingend sind (z. B. DeconstructionStart → PhaseM).
* Vorteil: Passt zu deinen SHACL-Shapes mit sh:minCount 1 und sh:maxCount 1, keine Änderungen an OCCP\_SHACL.ttl nötig.
* Nachteil: Weniger minimalistisch, da du explizite Beziehungen angeben musst.
* Beispiel:

turtle

:instant\_deconstruction\_start a occp:DeconstructionStart ;

occp:hasActualTime "2025-11-15"^^xsd:date ;

occp:startsPhase :phaseM .

:phaseM a occp:PhaseM\_Deconstruction ;

occp:hasActualBeginning :instant\_deconstruction\_start ;

occp:hasActualEnd :instant\_deconstruction\_completion .

* Test: Validiere mit OCCP\_SHACL.ttl – sollte Conforms: True ergeben.

2. Reasoning nutzen, um Beziehungen zu inferieren

* Idee: Definiere in OCCP\_V0.3.ttl OWL-Axiome, sodass der Reasoner (Pellet) automatisch Beziehungen wie occp:startsPhase einfügt, wenn ein DeconstructionStart existiert.
* Vorteil: ABox bleibt minimalistisch, SHACL prüft die inferierte Ontologie.
* Nachteil: Erfordert Änderungen an der TBox und möglicherweise komplexere Axiome.
* Beispiel in OCCP\_V0.3.ttl:

turtle

occp:DeconstructionStart a owl:Class ;

rdfs:subClassOf [

a owl:Restriction ;

owl:onProperty occp:startsPhase ;

owl:someValuesFrom occp:PhaseM\_Deconstruction ;

] .

occp:PhaseM\_Deconstruction a owl:Class ;

rdfs:subClassOf [

a owl:Restriction ;

owl:onProperty occp:hasActualBeginning ;

owl:someValuesFrom occp:DeconstructionStart ;

] .

* ABox:

turtle

:instant\_deconstruction\_start a occp:DeconstructionStart ;

occp:hasActualTime "2025-11-15"^^xsd:date .

:phaseM a occp:PhaseM\_Deconstruction ;

occp:hasActualEnd :instant\_deconstruction\_completion .

* Reasoning: Pellet sollte inferieren: :instant\_deconstruction\_start occp:startsPhase :phaseM und :phaseM occp:hasActualBeginning :instant\_deconstruction\_start.
* SHACL: Behalte sh:minCount 1 in osh:ShapeOfDeconstructionStart – es wird nach Reasoning erfüllt.

3. SHACL-Regeln anpassen (Kompromiss)

* Idee: Behalte sh:minCount 1 für zwingende Beziehungen wie occp:startsPhase (DeconstructionStart → PhaseM), aber mache andere OPs (z. B. occp:beginsBefore) optional (sh:minCount 0) und nutze SPARQL für die Zeitstempelprüfung.
* Vorteil: Respektiert deine Intention für DeconstructionStart, reduziert aber Redundanzen bei weniger kritischen Beziehungen.
* Beispiel:
  + osh:ShapeOfDeconstructionStart:

turtle

osh:ShapeOfDeconstructionStart a sh:NodeShape ;

sh:targetClass occp:DeconstructionStart ;

sh:property [

sh:path occp:startsPhase ;

sh:class occp:PhaseM\_Deconstruction ;

sh:minCount 1 ;

sh:maxCount 1 ;

] .

* + osh:ShapeOfPhaseA:

turtle

osh:ShapeOfPhaseA a sh:NodeShape ;

sh:targetClass occp:PhaseA\_Planning ;

sh:property [

sh:path occp:beginsBefore ;

sh:class occp:PhaseC\_Construction ;

sh:minCount 0 ; # Optional

] .

Diskussion: Was ist sinnvoll?

* Dein Ansatz: sh:minCount 1 für DeconstructionStart ist korrekt, wenn du sicherstellen willst, dass jede Instanz zwingend eine Phase M startet. Das Problem ist nur, dass die ABox diese Beziehung explizit enthalten muss, solange kein Reasoning sie ergänzt.
* Minimalistische ABox: Um Redundanzen zu vermeiden, müsstest du Reasoning nutzen (Option 2). Ohne TBox-Anpassung bleibt Option 1 (explizite Beziehungen) die einfachste Lösung.
* Zeitdruck: Da du wenig Zeit hast, sollten wir Änderungen minimieren. Große Anpassungen wie überall minCount rausnehmen oder komplexe OWL-Axiome sind jetzt nicht ideal.

Empfehlung

* Kurzfristig: Füge die zwingenden Beziehungen (z. B. occp:startsPhase für DeconstructionStart) wieder in OCCP\_Valid\_LCycle\_2.ttl ein, damit OCCP\_SHACL.ttl ohne Änderungen funktioniert. Das hält deine Shapes intakt und ist schnell umsetzbar.
* Langfristig: Für die Zukunft könnten wir Reasoning einbauen, aber das heben wir uns auf, wenn der Zeitdruck weg ist.

Hilfe beim Minimalbeispiel (Ketten)

Da du das Minimalbeispiel für Ketten (vermutlich UpdateChains aus OULD) schnell brauchst, lass uns das parallel angehen:

* Minimalbeispiel:

turtle

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix ould: <http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix : <http://www.example.de/example#> .

:chain1 a ould:UpdateChain ;

ould:hasUpdate :update1 ;

ould:hasUpdate :update2 ;

time:hasBeginning :instant\_chain\_start ;

time:hasEnd :instant\_chain\_end .

:update1 a ould:Update ;

occp:hasActualTime "2025-01-01"^^xsd:date .

:update2 a ould:Update ;

occp:hasActualTime "2025-01-02"^^xsd:date .

:instant\_chain\_start a time:Instant ;

occp:hasActualTime "2025-01-01"^^xsd:date .

:instant\_chain\_end a time:Instant ;

occp:hasActualTime "2025-01-03"^^xsd:date .

* SHACL-Regel (in OCCP\_SHACL.ttl einfügen):

turtle

osh:ShapeOfUpdateChain a sh:NodeShape ;

sh:targetClass ould:UpdateChain ;

sh:property [

sh:path ould:hasUpdate ;

sh:class ould:Update ;

sh:minCount 1 ;

sh:message "Each UpdateChain must have at least one Update."@en ;

sh:severity sh:Violation ;

] ;

sh:sparql [

sh:select """

PREFIX ould: <http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#>

PREFIX time: <http://www.w3.org/2006/time#>

SELECT $this ?update ?chainStart ?updateTime WHERE {

$this time:hasBeginning ?startInstant .

?startInstant occp:hasActualTime ?chainStart .

?this ould:hasUpdate ?update .

?update occp:hasActualTime ?updateTime .

FILTER (?updateTime < ?chainStart)

}

""" ;

sh:message "Each Update time { ?updateTime } must be at or after the chain start { ?chainStart }."@en ;

sh:severity sh:Violation ;

] .

Test

* Füge das Minimalbeispiel in eine neue ABox (z. B. OCCP\_Chain\_Example.ttl) und teste mit OCCP\_SHACL.ttl.

Nächste Schritte

1. ABox anpassen:
   * Füge zwingende Beziehungen wie :instant\_deconstruction\_start occp:startsPhase :phaseM wieder ein.
   * Teste mit OCCP\_SHACL.ttl.
2. Minimalbeispiel:
   * Nutze das obige Beispiel für Ketten und passe es an deine Bedürfnisse an.
   * Ich helfe dir, es zu validieren!

Fragen an dich

* ABox: Soll ich dir eine angepasste OCCP\_Valid\_LCycle\_2.ttl mit den nötigen Beziehungen schicken?
* Ketten: Ist das Minimalbeispiel so okay, oder brauchst du etwas Spezifisches?
* Zeit: Wann ist deine Deadline?

Ich bin hier, um das mit dir durchzuziehen – keine Sorge, wir kriegen das hin! Wie möchtest du weitermachen?