QKG2026 - ESWC 2026
May 10, 2026

Graphwise

A two-layered Approach to Cope with Recursion in SHACL Repairs

Robert David

Repairing SHACL Violations

GRAPHWISE
AI THRIVES ON WHOLE DATA

Shapes graph:

```
:StudentShape a sh:NodeShape;
    sh:targetNode :Ben;
    sh:property [
        sh:path :enrolledIn;
        sh:minCount 1;
        sh:class :Course; ] .
```

```
:CourseShape a sh:NodeShape;
    sh:targetClass :Course;
    sh:property [
        sh:path :courseID;
        sh:minCount 1;
        sh:maxCount 1; ] .
```

Data graph:

```
:Ben :enrolledIn :C1 .
```

Repairing SHACL Violations

GRAPHWISE
AI THRIVES ON WHOLE DATA

Shapes graph:

```
:StudentShape a sh:NodeShape;
    sh:targetNode :Ben;
    sh:property [
        sh:path :enrolledIn;
        sh:minCount 1;
        sh:class :Course; ] .
```

???

```
:CourseShape a sh:NodeShape;
    sh:targetClass :Course;
    sh:property [
        sh:path :courseID;
        sh:minCount 1;
        sh:maxCount 1; ] .
```

Repaired data graph:

```
:Ben :enrolledIn :C1 .
:C1 a :Course .
```

Repairing SHACL Violations

GRAPHWISE
AI THRIVES ON WHOLE DATA

Shapes graph:

```
:StudentShape a sh:NodeShape;
    sh:targetNode :Ben;
    sh:property [
        sh:path :enrolledIn;
        sh:minCount 1;
        sh:class :Course; ] .
```

Repaired data graph:

```
:Ben :enrolledIn :C1 .
:C1 a :Course .
```

New target
:C1

```
:CourseShape a sh:NodeShape;
    sh:targetClass :Course;
    sh:property [
        sh:path :courseID;
        sh:minCount 1;
        sh:maxCount 1; ] .
```

```
:C1 :courseID :someID .
```

Problem of Repair Target Recursion

GRAPHWISE
AI THRIVES ON WHOLE DATA

- SHACL violations can be **repaired**
  - by changing the data graph
  - **adding** and/or **removing triples**
  - so that it conforms to the constraints.

- Recent work: **SHACL repair program** to to compute repairs.
  - Previously determined target nodes in a preprocessing step.
  - Adapt program to rules to “dynamically” determine target nodes.
  - Can solve repair target recursion as part of repair computation.
  - **But, infinite cycles may occur!**

Infinite Repair Target Recursion

GRAPHWISE
AI THRIVES ON WHOLE DATA

Shapes graph:

```
:StudentShape a sh:NodeShape;
    sh:targetClass :Student;
    sh:property [
        sh:path :enrolledIn;
        sh:minCount 1;
        sh:class :Course; ] .
```

```
:CourseShape a sh:NodeShape;
    sh:targetClass :Course;
    sh:property [
        sh:path :enrolledStudent;
        sh:minCount 1;
        sh:class :Student; ] .
```

Data graph:

```
:Ben :enrolledIn :C1 .      :C1 a :Course .
:S1 :enrolledIn :C2 .
```

```
:C1 :enrolledStudent :S1 .      :S1 a :Student .
```

… continues infinitely

When does infinite recursion occur?

GRAPHWISE
AI THRIVES ON WHOLE DATA

Define basic scenarios where infinite recursion occurs:

1. Let (W, s) be a target with W a class target for class B and either s is a shape expression of the form $s \leftarrow \geq i\ p.B,\ i \geq 1$ or s is a shape expression of the form $s \leftarrow \geq i\ p.(c \wedge B),\ i \geq 0$. (W, s) is a class target recursion R_C .

2. Let (W, s) be a target with W a subjects-of target for property p and either s is a shape expression of the form $s \leftarrow \geq i\ p.s',\ i \geq 1$ or s is a shape expression of the form $s \leftarrow \geq i\ p.c,\ i \geq 0$. (W, s) is a subjects-of target recursion R_{PS} .

3. Likewise for objects-of target and (W, s) is a objects-of target recursion R_{PO}.

7

A two-layered Solution Approach

GRAPHWISE
AI THRIVES ON WHOLE DATA

Goal: allow for a maximum of scenarios without limitation. E.g., allow finite recursion.

- Run **static cycle detection**
  - Only additions can cause infinite cycles. Deletions are limited by the data.
  - Check if infinite recursion can occur.
    - If no, then run program as is.
    - If yes, then modify repair program before running.
- Add **repair program guards**
  - Repair program is based on Answer Set Programming (ASP).
  - Implemented guard is a **finite domain** for fresh values.
  - Additions are limited by picking from the set of fresh values.

8

Finite Domain Guards for ASP

GRAPHWISE
AI THRIVES ON WHOLE DATA

Initial data graph:

```
new(1..3)     % generates 1 to 3 new atoms, i.e., new(1), new(2), new(3)

:Ben :enrolledIn :C1
```

Repaired data graph:

```
:C1 :enrolledStudent new(1) .          :Student(new(1)) .

new(1) :enrolledIn new(2) .            :Course(new(2)) .

new(2) :enrolledStudent new(3) .       :Student(new(3)) .
```

Next, the program should add another :enrolledIn for new(3). However, there are no new atoms left. Therefore, the repair process is reported to be unsatisfiable.

What about other implementations?

- Static cycle detection can be done with many programming languages.
- Runtime guards are specific to the programming language:
  - Finite domain is required by ASP because of how ASP grounding works and has the drawback of pre-defining the size.
  - Implementation available at: https://github.com/robert-david/shacl-repairs/tree/target-recursion-guard
- Some other guards can be:
  - Explicit depth limit, Explicit resource limit, …

What is the best option, i.e. guard+repair solution, **in practice?**

10

Repair Target Recursion

GRAPHWISE
AI THRIVES ON WHOLE DATA

Problem: Repair target recursion can cause infinite cycles.

Approach: We propose a two-layered approach to allow a maximum of finite recursion cases to be supported.

Future work: Look in into extended scenarios.
Look at alternative implementations.