# Magic Shapes for SHACL Validation 

Shqiponja Ahmetaj<br>TU Wien (Austria), WU Wien (Austria)<br>shqiponja.ahmetaj@tuwien.ac.at

Bianca Löhnert<br>TU Wien (Austria)<br>bianca@loehnert-net.at

Magdalena Ortiz<br>TU Wien (Austria)<br>ortiz@kr.tuwien.ac.at

Mantas Šimkus<br>Umeå University (Sweden), TU Wien (Austria)<br>simkus@cs.umu.se


#### Abstract

A key prerequisite for the successful adoption of the Shapes Constraint Language (SHACL)-the W3C standardized constraint language for RDF graphs-is the availability of automated tools that efficiently validate targeted constraints (known as shapes graphs) over possibly very large RDF graphs. There are already significant efforts to produce optimized engines for SHACL validation, but they focus on restricted fragments of SHACL. For unrestricted SHACL, that is SHACL with unrestricted recursion and negation, there is no validator beyond a proof-of-concept prototype, and existing techniques are inherently incompatible with the goal-driven approaches being pursued by existing validators. Instead they require a global computation on the entire data graph that is not only computationally very costly, but also brittle, and can easily result in validation failures due to conflicts that are irrelevant to the validation targets.

To address these challenges, we present a 'magic' transformationbased on Magic Sets as known from Logic Programming-that transforms a SHACL shapes graph $S$ into a new shapes graph $S^{\prime}$ whose validation considers only the relevant neighbourhood of the targeted nodes. The new $S^{\prime}$ is equivalent to $S$ whenever there are no conflicts between the constraints and the data, and in case the validation of $S$ fails due to conflicts that are irrelevant to the target, $S^{\prime}$ may still admit a lazy, target-oriented validation. We implement the algorithm and run preliminary experiments, showing our approach can be a stepping stone towards validators for full SHACL, and that it can significantly improve the performance of the only prototype validator that currently supports full recursion and negation.


## PVLDB Reference Format:

Shqiponja Ahmetaj, Bianca Löhnert, Magdalena Ortiz, and Mantas Šimkus. Magic Shapes for SHACL Validation. PVLDB, 15(10): 2284-2296, 2022.
doi:10.14778/3547305.3547329

## PVLDB Artifact Availability:

The source code, data, and/or other artifacts have been made available at https://github.com/biziloehnert/magicSHACL.

## 1 INTRODUCTION

The Shapes Constraint Language (SHACL) was recently standardized by the W3C as a formalism for checking the quality of RDF

[^0]graphs; we refer to [15] for an introduction. In SHACL, the main problem is to check whether a given RDF graph $G$ validates a SHACL shapes graph $(C, T)$, where $C$ is a set of constraints, each associated to a so-called shape name, and $T$ is a specification of nodes (targets) from the data graph which should validate certain shapes from $C$. For illustration, consider a SHACL shapes graph $(C, T)$, where $C=\{$ Queen $\leftarrow \exists$ married $\cdot \top \wedge \exists$ has.crown\} contains one constraint stating that each Queen must be married and have a crown, and in $T$ we have the shape atom Queen (Sissi). Here Queen is a shape name, married and has are data predicates, i.e., properties, and crown is a constant. The graph $G=\{$ married(Sissi, Franz) $\}$ does not validate $(C, T)$, but the extended graph $G^{\prime}=G \cup\{$ has(Sissi, crown) $\}$ does.

The standard specifies a syntax for expressing SHACL constraints and describes when they are validated by RDF graphs. It allows for recursive constraints, that is, constraints that involve cyclic dependencies, but their semantics is not defined in the specification and left open to implementers. This has motivated some recent logicbased proposals to formalize the semantics of full SHACL [4, 8].

All formalizations of SHACL validation-with and without recursion-are based on assignments of shapes to nodes in the data graph, which have to comply with the constraints and contain the targets. Such assignments are global in the sense that they cover all nodes in the data graph. For tractable fragments that restrict the interaction of recursion and negation, it has been shown that validation under global (a.k.a. strict faithful) assignments coincides with validation under a weaker form of faithful assignments that allow to decide only some shape at some nodes, and to leave others undefined. Moreover, such faithful assignments can be obtained starting from the nodes mentioned in the targets and visiting only their relevant neighbourhood, while deciding the value of assignments only for the shape names occurring in the constraints.

Unfortunately, the picture is significantly different in the presence of negation and recursion, whose interaction may hinder the existence of a valid shape assignment and thus make a data graph inconsistent with a set of SHACL constraints. Consider the following example shapes graph $(C, T)$ and data graph $G$ :

$$
\begin{aligned}
& C=\{\text { Crowned } \leftarrow \exists \text { crownedBy. } \neg \text { Crowned }\} \\
& T=\{\text { Crowned }(\text { Sissi })\} \\
& G=\{\text { crownedBy }(\text { Sissi, Archbishop }), \text { crownedBy }(\text { Tim }, \text { Tim })\}
\end{aligned}
$$

Crowned(Sissi) satisfies the constraint for Crowned since Sissi is crowned by the Archbishop, who himself is not Crowned. Although one may expect the target to be valid, no valid global assignment exists, since there is no consistent way to assign or not assign node Tim to the shape name Crowned. This may not be desired


Figure 1: Using Magic Shapes for SHACL validation. The input set $C$ and targets $T$ are transformed into a (typically smaller and simpler) shapes graph $C_{\text {magic }}$ relevant for the targets, which can be validated with off-the-shelf validators
since the part of the data graph relevant to validating the target Crowned (Sissi) is independent from Tim being crowned or not, and indeed, a faithful assignment could leave Crowned (Tim) undefined. This is a toy example, but in real life RDF graphs may be huge and errors may be unavoidable. Simply discarding the entire graph because of a problematic fact unrelated to our targets may be highly undesirable. Moreover, finding a global assignment over the full graph is costly. It makes the worst-case complexity of validation to go up from P to NP-hard [8], and additionally, it is incompatible with the target-guided top-down approaches. Indeed, recent and ongoing efforts to implement scalable SHACL validators use this type of goal-oriented approaches [7, 14], but focus on tractable fragments of SHACL, and do not handle unrestricted SHACL, that is with unrestricted recursion and negation. Existing algorithms search for global assignment, and only one of them is implemented in a proof-of-concept prototype [4]. To our knowledge, no targetoriented way to compute faithful assignments looking only at the relevant part of the graph has been proposed.

The focus of this work is SHACL validation in the presence of unrestricted negation and recursion. We build on Magic Sets [2, 3, 6], a well-known technique developed in the context of deductive databases to combine the advantages of top-down and bottom-up evaluation. In a nutshell, it uses the query goal to adorn the input program with binding information, obtaining a new program whose bottom-up evaluation, analogously to the top-down one, involves only the part of the data relevant to answering the query. We adapt this technique to SHACL shape graphs. Our main contribution is a Magic Shapes technique that takes as input a shapes graph $S=(C, T)$, and produces a new, potentially significantly smaller 'magic shapes graph' $S_{m}=\left(C_{\text {magic }}, T\right)$. On the one hand, $C_{\text {magic }}$ discards the constraints in $C$ that are not necessary for validating the target $T$. At the same time, it generates a set of 'magic constraints' that, in a nutshell, ensure that validation is always restricted to a relevant fragment of the input data graph, starting from the nodes directly mentioned in $T$, and iteratively identifying their relevant neighborhood. The output $S_{m}$ can then be passed to existing validators instead of the input $S$, see Figure 1. Whenever the data graph is consistent with the constraints and a global assignment exists, $S_{m}$ and $S$ are equivalent. Otherwise, if there is no global assignment, the data graph may still validate the 'magic shapes graph' if the problematic part of the data graph and set of constraints is not relevant to the current target. In this way, we can find faithful assignments that leave irrelevant shape adornments undefined, enabling a more fine-grained validation than under current semantics. For tractable fragments, it also provides an easy way to do validation on a potentially much smaller fragment of the
input RDF graph, while ignoring constraints that do not affect validation of the targets, independently of the algorithms implemented by the specific validator at hand.

Structure of the paper. In Section 2, we present the syntax of SHACL and the notion of validation under both the classical (or supported) and stable model semantics. We describe the Magic Shapes algorithm in Section 3 and Section 4. To illustrate the general ideas of the algorithm, we first concentrate in Section 3 on the positive fragment of SHACL. The full algorithm for SHACL constraints with arbitrary negation and recursion is described in Section 4. Section 5 is dedicated to showing the correctness of the Magic Shapes algorithm. We then discuss in Section 6 how the Magic Shapes technique allows us to find faithful assignments and define an inconsistency-tolerant version of the semantics. We implemented the Magic Shapes technique and did some preliminary experiments; the results are presented in Section 7.
Related Work. Logic-based proposals to formalize the semantics of full SHACL-with recursive constraints-have emerged recently. Andresel et al. [4] proposed a semantics based on the stable models semantics for logic programs, stricter than the semantics based on classical logic due to Corman el al. [8]. Both semantics coincide with the official recommendation for non-recursive constraints. There is a close connection between SHACL and the more established family of Description Logics [5], which has been used to gain new insights into SHACL (see, e.g., [16, 20]). There has also been increasing interest in scalable SHACL validation. Shacl2Spargl [7] is a validation engine that checks conformance of RDF graphs with SHACL constraints by evaluating SPARQL queries against the data, which optimizes the order in which shapes are processed. Further optimization techniques are implemented in Trav-Shacl [14]. However, these works focus on tractable fragments of SHACL, and do not handle unrestricted interaction of recursion and negation in constraints. To our knowledge, the only implementation that validates unrestricted SHACL is the Shacl-Asp prototype from [4], which translates SHACL constraints into answer set programs (ASP) [12] and evaluates them using the DLV system [1]. Note that [7] proposes an algorithm for unrestricted SHACL that involves a SAT solver, but to our knowledge, with no available implementation.

Our Magic Shapes technique is inspired by the well-known Magic Set transformation for optimizing evaluation of Datalog programs in bottom-up systems [6, 18], and its extensions for Datalog with disjunction [3, 9] and (possibly stratified) negation [2, 6, 13, 21]. In paticular, we build on the ideas of the Magic Set method presented in [13] for Datalog with unstratified negation, which goes beyond optimization, and also addresses the challenge of meaningful semanics in the face of possible inconsistencies.

## 2 PRELIMINARIES

We present here SHACL and the notion of validation by RDF graphs under the supported model semantics and the stable model semantics. We follow the formalization from [4].

Data graph Let $\mathbf{N}$ and $\mathbf{P}$ denote infinite, disjoint sets of nodes and property names, respectively. A (data) graph $G$ is a finite set of atoms of the form $p(a, b)$, where $p \in \mathbf{P}$ and $a, b \in \mathbf{N}$. The set of nodes appearing in $G$ is denoted with $V(G)$.

Syntax of SHACL Let $S$ be an infinite set of shape names, disjoint from $\mathbf{N}$ and $\mathbf{P}$. A shape atom is an expression of the form $s(a)$, where $s \in \mathrm{~S}$ and $a \in \mathrm{~N}$. A path expression $E$ is a regular expression build from the operators $*, \cdot, \cup$, and symbols $p$ or expressions $p^{-}$, where $p \in \mathbf{P}$. A (shape) expression is an expression $\phi$ of the form:

$$
\phi:=\top|s| a\left|\phi_{1} \wedge \phi_{2}\right| \phi_{1} \vee \phi_{2}|\neg \phi| \geq_{n} E . \phi \mid E=E^{\prime}
$$

where $s \in \mathrm{~S}, a \in \mathrm{~N}, n \in \mathbb{N}$, and $E, E^{\prime}$ are path expressions. We use the abbreviations $\exists E . \phi$ for $\geq_{1} E . \phi$, and $\leq_{n} E . \phi$ for $\neg\left(\geq_{n+1} E . \phi\right)$. A (shape) constraint is of the form $s \leftarrow \phi$, where $s \in \mathrm{~S}$ and $\phi$ is a shape expression. We may sometimes call $s$ the head and $\phi$ the body of the constraint. A target set (or simply target) is a set of shape atoms of the form $s(a)$ where $s \in \mathrm{~S}$ and $a \in \mathrm{~N}$. A shapes graph is a pair $(C, T)$, where $C$ is a set of constraints and $T$ is a target. The definition $\phi_{s, C}$ of a shape name $s$ in a set of constraints $C$ is the disjunction of all shape expressions in the body of a shape constraint in $C$ whose head shape is $s$. That is, $\phi_{s, C}=\bigvee_{s \leftarrow \phi \in C} \phi$. If $C$ is clear from the context, we may simply write $\phi_{s}$ instead of $\phi_{s, C}$. For the theoretical part of this work, we only consider ground targets given as shape atoms, since the other types of targets (like the so-called class targets) can be converted to ground targets using SPARQL queries. The SHACL recommendation ${ }^{1}$ provides a possible definition in SPARQL for each target type.

Evaluation of shape expressions An assignment $A$ for a graph $G$ is a set of shape atoms such that $v \in V(G)$ for each $s(v) \in A$. The set $G \cup A$ is called decorated graph for $G$. The evaluation of a (complex) shape expression w.r.t. a decorated graph $I$ is given in Table 1 in terms of a function $\llbracket \cdot \rrbracket^{I}$ that maps a (complex) shape expression $\phi$ to a set of nodes, and a path expression $E$ to a set of pairs of nodes. Throughout the paper, when validating a graph $G$ against a shapes graph $S$, we assume that all constants that occur in $S$ also appear in $G$.

Supported and stable model semantics We first present the supported model semantics introduced in [8].

Definition 2.1. Given a set of shape constraints $C$, a decorated graph $I$ is called a supported model of $C$, if $\llbracket \phi_{s, C} \rrbracket^{I}=\llbracket s \rrbracket^{I}$ holds for each shape name $s$ in $C$. A graph $G$ validates a shapes graph $(C, T)$ under the supported model semantics if there exists an assignment $A$ for $G$ such that (i) $G \cup A$ is a supported model, and (ii) $T \subseteq A$.

The stable model semantics restricts the supported model semantics by requiring that every shape atom of a decorated graph has a (non-circular) justification. This is achieved by requiring the existence of a level assignment [4].

[^1]Table 1: Evaluation of shape expressions

$$
\begin{array}{ll}
\llbracket \top \rrbracket^{I}=V(I) & \llbracket c \rrbracket^{I}=\{c\} \\
\llbracket p \rrbracket^{I}=\left\{\left(v, v^{\prime}\right) \mid p\left(v, v^{\prime}\right) \in I\right\} & \llbracket p^{-} \rrbracket^{I}=\left\{\left(v, v^{\prime}\right) \mid p\left(v^{\prime}, v\right) \in I\right\} \\
\llbracket E \cup E^{\prime} \rrbracket^{I}=\llbracket E \rrbracket^{I} \cup \llbracket E^{\prime} \rrbracket^{I} & \llbracket E \cdot E^{\prime} \rrbracket^{I}=\llbracket E \rrbracket^{I} \circ \llbracket E^{\prime} \rrbracket^{I} \\
\llbracket E^{*} \rrbracket^{I}=\{(v, v) \mid v \in V(I)\} \cup \llbracket E \rrbracket^{I} \cup \llbracket E \cdot E \rrbracket^{I} \cup \llbracket E \cdot E \cdot E \rrbracket^{I} \cup \cdots \\
\llbracket s \rrbracket^{I}=\{v \mid s(v) \in I\} & \llbracket \neg \phi \rrbracket^{I}=V(I) \backslash \llbracket \phi \rrbracket^{I} \\
\llbracket \phi_{1} \vee \phi_{2} \rrbracket^{I}=\llbracket \phi_{1} \rrbracket^{I} \cup \llbracket \phi_{2} \rrbracket^{I} & \llbracket \phi_{1} \wedge \phi_{2} \rrbracket^{I}=\llbracket \phi_{1} \rrbracket^{I} \cap \llbracket \phi_{2} \rrbracket^{I} \\
\llbracket \geq_{n} E \cdot \phi \rrbracket^{I}=\left\{v| |\left\{\left(v, v^{\prime}\right) \in \llbracket E \rrbracket^{I} \text { and } v^{\prime} \in \llbracket \phi \rrbracket^{I}\right\} \mid \geq n\right\} \\
\llbracket E=E^{\prime} \rrbracket^{I}=\left\{v \mid \forall v^{\prime}:\left(v, v^{\prime}\right) \in \llbracket E \rrbracket^{I} \text { iff }\left(v, v^{\prime}\right) \in \llbracket E^{\prime} \rrbracket^{I}\right\}
\end{array}
$$

Definition 2.2 (Level assignment). Let $I$ be supported model. A level assignment for $I$ is a function level that maps tuples in $\{(\phi, v) \mid$ $\left.v \in \llbracket \phi \rrbracket^{I}\right\}$ to integers, and satisfies the following conditions:
(i) $\operatorname{level}\left(\phi_{1} \wedge \phi_{2}, v\right)=\max \left(\left\{\operatorname{level}\left(\phi_{1}, v\right)\right.\right.$, level $\left.\left.\left(\phi_{2}, v\right)\right\}\right)$
(ii) $\operatorname{level}\left(\phi_{1} \vee \phi_{2}, v\right)=\min \left(\left\{\operatorname{level}\left(\phi_{1}, v\right)\right.\right.$, level $\left.\left.\left(\phi_{2}, v\right)\right\}\right)$
(iii) level $\left(\geq_{n} E . \phi, v\right)$ is the smallest $k \geq 0$ for which there exist $n$ nodes $v_{1}, v_{2}, \ldots, v_{n}$ such that for all $1 \leq i \leq n$
(a) $\left(v, v_{i}\right) \in \llbracket E \rrbracket^{I}, v_{i} \in \llbracket \phi \rrbracket^{I}$, and
(b) $\operatorname{level}\left(\phi, v_{i}\right) \leq k$.

Definition 2.3 (Stable Model Semantics). A decorated graph I is a stable model of a set $C$ of constraints, if (i) $I$ is a supported model of $C$, and (ii) there exists a level assignment such that for all $s(v) \in I$ level $\left(\phi_{s}, v\right)<\operatorname{level}(s, v)$. A set of shape constraints $C$ is called consistent with $G$ under the supported (stable) model semantics if there exists a supported (stable) model of $C$; otherwise $C$ is called inconsistent with $G$ (under the supported (stable) model semantics).

A data graph $G$ validates a shapes graph $(C, T)$ under the stable model semantics if there exists an assignment $A$ such that (i) $G \cup A$ is a stable model of $C$, and (ii) $T \subseteq A$.

Clearly, every stable model is also a supported model, but the converse may not be the case. We illustrate this on an example.

Example 2.4. Consider a shapes graph $(C, T)$ and a data graph $G$ defined as follows:

$$
\begin{aligned}
C= & \{\text { Queen } \leftarrow \exists \text { married. } \text { King } \vee \exists \text { has.crown }, \\
& \text { King } \leftarrow \exists \text { married. Queen } \vee \exists \text { has.crown }\} \\
T= & \{\text { King }(\text { Franz })\} \\
G= & \{\text { married }(\text { Harry, Meghan }), \text { married }(\text { Meghan, Harry }), \\
& \text { married }(\text { Franz, Sissi }), \text { married }(\text { Sissi, Franz }), \\
& \text { has }(\text { Franz, crown })\}
\end{aligned}
$$

Consider the following shape assignments:

```
\(A_{1}=\{\operatorname{King}(\) Harry \()\), Queen \((\) Meghan \(), \operatorname{King}(\) Franz \()\), Queen \((\) Sissi \()\}\)
\(A_{2}=\{\operatorname{King}(\) Meghan \()\), Queen(Harry), King(Franz), Queen(Sissi) \(\}\)
\(A_{3}=\{\operatorname{King}(\) Harry \()\), Queen(Meghan), King(Sissi), Queen(Franz) \(\}\)
\(A_{4}=\{\operatorname{King}(\) Meghan \()\), Queen(Harry), King(Sissi), Queen(Franz) \(\}\)
\(A_{5}=\{\operatorname{King}(\) Franz \()\), Queen \((\) Sissi \()\}\)
\(A_{6}=\{\operatorname{King}(\) Sissi \()\), Queen \((\) Franz \()\}\)
```

The decorated graphs $I_{i}=G \cup A_{i}$ with $1 \leq i \leq 6$ all satisfy the condition $\llbracket \phi_{s} \rrbracket^{I_{i}}=\llbracket s \rrbracket^{I_{i}}$ for each shape name $s$ occurring in $C$, and hence, each $I_{i}$ is a supported model of $C$. The models $I_{i}$ with $1 \leq i \leq$ 4 have no level assignment that fulfills the conditions of Definition 2.3 item (ii), hence they are not stable models. However, $I_{5}$ and $I_{6}$ are stable models. To show that $I_{5}$ is a stable model we give the following level assignment, with $v \in\{$ Harry, Megan, Franz, Sissi $\}$ :

$$
\begin{array}{ll}
\operatorname{level}(\mathrm{T}, v)=0 & \operatorname{level}(\exists \text { has.crown, Franz })=0 \\
\operatorname{level}(\text { King, Franz })=1 & \operatorname{level}(\text { Queen, Sissi })=1
\end{array}
$$

Note that $\operatorname{level}\left(\phi_{s}, v\right)<\operatorname{level}(s, v)$ for all $s(v) \in G \cup A_{5}$ as required. The level assignment for $I_{6}$ is similar.

Brave and Cautious validation The above notions of validation only check for the existence of a supported- or stable model, which can be viewed as brave validation. Andresel et al. [4] also study cautious validation, which aims to validate the targets that are true in every stable model. We recall this notion for both semantics.

Definition 2.5. A data graph $G$ cautiously validates a shapes graph ( $C, T$ ) under the supported (stable) model semantics if $T \subseteq A$ for every supported (stable) model $G \cup A$ of $C$.

We illustrate brave and cautious validation with an example.
Example 2.6. Consider Example 2.4. The target $T$ is contained in the supported models with assignment $A_{1}, A_{2}$ and $A_{5}$, but not $A_{3}, A_{4}$ and $A_{5}$. Hence, the data graph $G$ bravely, but not cautiously, validates the shapes graph $(C, T)$ under the supported model semantics. For the stable model semantics there exist two models, namely $G \cup A_{5}$ and $G \cup A_{6}$. Since $T \subseteq A_{5}$, but $T \nsubseteq A_{6}$ the data graph $G$ bravely, but not cautiously validates the shapes graph $(C, T)$ under the stable model semantics.

To ease presentation, we use the following normal form for SHACL constraints.

Definition 2.7 (Normal form for constraints). A constraint is in normal form if it has one of the following forms:

$$
\begin{array}{lll}
\text { (NF1) } s \leftarrow T & \text { (NF2) } s \leftarrow a & \text { (NF3) } s \leftarrow E=E^{\prime} \\
(N F 4) s \leftarrow \neg s^{\prime} & \text { (NF5) } s \leftarrow s_{1} \wedge \cdots \wedge s_{n} & (N F 6) s \leftarrow \geq_{n} E . s^{\prime}
\end{array}
$$

It was shown in [4] (Proposition 4.2) that a set of constrains $C$ can be transformed in polynomial time into a normalized set of constraints $C^{\prime}$ such that for every graph $G$ and target set $T, G$ validates $(C, T)$ iff $G$ validates ( $C^{\prime}, T$ ) under supported- and stable model semantics. Further, it can be shown that every model $I$ of $(C, T)$ can be transformed into a model $I^{\prime}$ of $\left(C^{\prime}, T\right)$, so that both brave- and cautious validation are preserved.

## 3 MAGIC SHAPES ALGORITHM FOR SHACL

Now we describe the Magic Shapes algorithm for SHACL validation. For simplicity, we first present it for the positive fragment of SHACL, that is, constraints in normal form that do not include expressions of the form (NF4). In the next section, we extend it to constraints with unrestricted use of negation. The transformation follows ideas from the Magic Sets algorithm for DATALoG $\urcorner$ in [13].

We first identify the shape names that are reachable from a target and the constraints that are relevant for validating it.

Definition 3.1. Given a shapes graph $(C, T)$, where $C$ is positive, we let $\operatorname{reach}(C, T)$ be the smallest set of shape names such that:

- if $s(a) \in T$, then $s \in \operatorname{reach}(C, T)$, and
- if $s \leftarrow \phi \in C$ and $s \in \operatorname{reach}(C, T)$, then every shape name occurring in $\phi$ is also in $\operatorname{reach}(C, T)$.
The set of constraints $C_{T}=\{s \leftarrow \phi \in C \mid s \in \operatorname{reach}(C, T)\}$ is called the module of $C$ w.r.t. $T$.

Example 3.2. Consider a shapes graph $(C, T)$, where $C$ contains:

$$
\begin{array}{lll}
s_{1} \leftarrow s_{2} \wedge s_{3} & s_{2} \leftarrow \exists r . s_{4} & s_{3} \leftarrow s_{2} \\
s_{4} \leftarrow \exists r & s_{5} \leftarrow s_{6} & s_{6} \leftarrow \exists r . s_{2}
\end{array}
$$

and $T=\left\{s_{1}(a)\right\}$. Then, $\operatorname{reach}(C, T)=\left\{s_{1}, s_{2}, s_{3}, s_{4}\right\}$, resulting in the module $C_{T}=\left\{s_{1} \leftarrow s_{2} \wedge s_{3}, s_{2} \leftarrow \exists r . s_{4}, s_{3} \leftarrow s_{2}, s_{4} \leftarrow \exists r\right\}$.

The algorithm takes as input a shapes graphs $(C, T)$ and outputs a new optimized ( $\left.C_{\text {magic }}, T\right)$ against which data graphs can be validated. The algorithm consists of four main steps, which are summarized in Algorithm 1 and described in more detail below.

```
Input :SHACL shapes graph \((C, T)\)
Output: Optimized shapes graph \(\left(C_{\text {magic }}, T\right)\).
begin
    \(C_{\text {generate }}:=\{\) magic_s \(\leftarrow v \mid s(v) \in T\} ;\)
    \(C_{T}:=\{s \leftarrow \phi \in C \mid s \in \operatorname{reach}(C, T)\} ;\)
    foreach \(s \leftarrow \phi \in C_{T}\) do
        \(C_{\text {generated }}:=C_{\text {generated }} \cup\{\) Generate \((s \leftarrow \phi)\}\);
    end
    \(C_{\text {modified }}:=\left\{s \leftarrow\right.\) magic_s \(\left.\wedge \phi \mid s \leftarrow \phi \in C_{T}\right\}\);
    \(C_{\text {magic }}:=C_{\text {generated }} \cup C_{\text {modified }} ;\)
    return ( \(C_{\text {magic }}, T\) );
10 end
Algorithm 1: Magic Shape Algorithm for positive SHACL
```

Modularize. The step modularize in line 3 extracts the set of constraints relevant for the targets, i.e. the module $C_{T}$. Note that this depends on the shape names in $T$, but not on the targeted nodes.

Create magic seeds. In line 2, constraints of the form magic_s $\leftarrow$ $v$, called magic seeds, are created for each shape atom $s(v)$ in $T$ and added to the set $C_{\text {generate }}$ of generated constraints.

Example 3.3. Consider again Example 3.2. The algorithm adds the magic seed magic_ $s_{1} \leftarrow a$ in $C_{\text {generate }}$. Intuitively, this constraint says that the validation of $s_{1}$ at $a$ is relevant.

Generate. The module $C_{T}$ is used to generate magic constraints in lines 4-6 through the procedure Generate, described in Algorithm 2. In a nutshell, for each $r \in C_{T}$ of the form $s \leftarrow \phi$ and for each shape name $s^{\prime}$ in $\phi$, it creates a new constraint with magic_s in the body and magic_s in the head. If in the constraint two shapes are connected by a path expression $E$, then $E$ is inverted. Intuitively, these constraints 'mark' the nodes and shapes that are relevant for validating the current target. The constraint magic_s $s^{\prime} \leftarrow$ magic_s can be read as 'if the $s$-assignment of a node $v$ is marked as relevant, then the $s^{\prime}$-assignment of $v$ is also marked as relevant'.

```
function Generate(s\leftarrow\phi) is
    if }\phi=\mp@subsup{s}{1}{}\wedge\cdots\wedge\mp@subsup{s}{n}{}\mathrm{ then
        return
            {magic_s}\mp@subsup{s}{1}{}\leftarrow\mathrm{ magic_s;...;magic_s_
    if }\phi=\mp@subsup{\geq}{n}{n}E.\mp@subsup{s}{}{\prime}\mathrm{ then
        return {magic_s's}\leftarrow\mp@subsup{\geq}{n}{}\mp@subsup{E}{}{-}.\mathrm{ .magic_s};
    else
        return \emptyset;
    end
end
```

Algorithm 2: Generation

Example 3.4. In our running example, procedure Generate adds four constraints to $C_{\text {generate }}$, namely

$$
\begin{array}{ll}
\text { magic_} s_{2} \leftarrow \text { magic_s } s_{1} & \text { magic_ } s_{3} \leftarrow \text { magic_s } s_{1} \\
\text { magic_s } s_{4} \leftarrow \exists r^{-} . \text {magic_s } s_{2} & \text { magic_ } s_{2} \leftarrow \text { magic_s }
\end{array}
$$

These constraints together with the magic seeds in Example 3.3 comprise now $C_{\text {generate }}$.

Modification. In line 7, the body of each constraint in the module $C_{T}$ is enhanced with a magic version of the shape name in its head. Intuitively, this ensures that the resoner does not continue with the validation of shapes assignments at nodes that are not marked as relevant by the magic rules, and restricts models to only the relevant atoms.

Example 3.5. Consider again $C_{T}$ from Example 3.2. The set of constraints $C_{\text {modified }}$ contains:

$$
\begin{array}{ll}
s_{1} \leftarrow \text { magic_s } s_{1} \wedge s_{2} \wedge s_{3} & s_{2} \leftarrow \text { magic_s } s_{2} \wedge \exists r . s_{4} \\
s_{3} \leftarrow \text { magic_s } s_{3} \wedge s_{2} & s_{4} \leftarrow \text { magic_s } s_{4} \wedge \exists r
\end{array}
$$

The output $C_{\text {magic }}$ of the magic shapes algorithm consists of the union of the sets $C_{\text {generated }}$ and $C_{\text {modified }}$, see line 8 of Algorithm 1.

Example 3.6. To demonstrate the magic shapes algorithm, consider a data graph $G=\{r(a, a), r(a, c)\}$. When validating $G$ against the shapes graph $(C, T)$ from Example 3.2, there exists a stable model $G \cup A$, with the assignment being

$$
\begin{aligned}
A=\{ & s_{1}(a), s_{2}(a), s_{3}(a), s_{4}(a), s_{5}(a), s_{6}(a) \\
& \left.s_{1}(c), s_{2}(c), s_{3}(c), s_{4}(c)\right\}
\end{aligned}
$$

The target $s_{1}(a)$ is in $A$, and therefore, $G$ validates $(C, T)$. When validating $G$ against the magic shapes graph $\left(C_{\text {magic }}, T\right)$, then we get a supported model $G \cup M \cup A^{\prime}$, with

$$
\begin{aligned}
M & =\left\{\operatorname{magic} s_{1}(a), \text { magic_} s_{2}(a), \text { magic_} s_{3}(a), \text { magic_ } s_{4}(a)\right\} \\
A^{\prime} & =\left\{s_{1}(a), s_{2}(a), s_{3}(a), s_{4}(a)\right\} .
\end{aligned}
$$

Again, it is the case that $s_{1}(a) \in A^{\prime}$. Note that the assignment $A^{\prime} \subseteq A$ contains only 4 shape atoms. Indeed, the module $C_{T}$ does not contain the constraints regarding $s_{5}$ and $s_{6}$, which are not reachable from the target, and hence, $s_{5}(a)$ and $s_{6}(a)$ are not in $A^{\prime}$. Moreover, the set $M$, obtained from $C_{\text {generate }}$ and $G$, discards the irrelevant magic shape atoms concerning node $c$, which together with $C_{\text {modified }}$, restrict the assignments of the shape names $s_{1}$ to $s_{4}$ to the relevant node $a$ only.

## 4 ALGORITHM TO SUPPORT NEGATION

The presence of arbitrary negation and recursion in SHACL constraints may cause inconsistency with the input data graph, making every target inherently invalid as there cannot exist a stable- or supported model that satisfies the constraints. E.g., for a constraint of the form $s \leftarrow \neg s$, there is no way to assign or not assign a node to the shape name $s$ so that the constraint is satisfied. It is well-known from the database literature [19] that parts of a logic program that may give rise to inconsistency can be caused by this kind of cyclic dependencies with an odd number of negated predicates. An analogous situation arises in SHACL, where odd cycles may make some assignments invalid. We need to identify such so-called dangerous constraints that appear under the scope of negation, and if they are relevant for validating the target, they should participate in the magic algorithm. In this way, the magic shapes graph preserves all valid shape assignments, and only allows validation of inconsistent inputs if the contradictions don't interact with validating the target.

Definition 4.1. [Dangerous constraints] Assume a set of constraints $C$. The shape dependency graph of $C$ is the marked directed graph $D G_{C}$ with an edge from $s_{1}$ to $s_{2}$ if there is a constraint $s_{2} \leftarrow \phi$ in $C$ such that $s_{1}$ occurs in $\phi$, and the edge ( $s_{1}, s_{2}$ ) is marked if $\phi=\neg s_{2}$. A cycle of $D G_{C}$ is a sequence of nodes $S=n_{1}, \ldots, n_{k}$, such that $n_{1}=n_{k}$, each $n_{i}$ for $1<i<k$ occurs exactly once in $S$, and $\left(n_{i}, n_{i+1}\right)$ for $(1 \leq i<k)$ is an edge in $D G_{C}$. An odd cycle in $D G_{C}$ is a cycle, where an odd number of edges are marked.

A shape name $s$ is dangerous in $C$ if
(i) $s$ occurs in an odd cycle of $D G_{C}$, or
(ii) s occurs in $\phi$ for some constraint $s^{\prime} \leftarrow \phi \in C$ such that $s^{\prime}$ is dangerous.
We say a constraint $s^{\prime} \leftarrow \phi$ is dangerous of $s^{\prime}$ is dangerous.
Example 4.2. Consider the set of shape constraints $C$ :

$$
\begin{array}{lll}
s_{1} \leftarrow \neg s_{1} \wedge s_{2} & s_{2} \leftarrow s_{3} & s_{3} \leftarrow \exists r . \neg s_{4} \\
s_{4} \leftarrow \exists r . \neg s_{3} & s_{5} \leftarrow s_{4} \wedge \neg s_{6} & s_{6} \leftarrow s_{4} \wedge \neg s_{5}
\end{array}
$$

This set has one odd cycle involving the shape name $s_{1}$. Since $s_{1}$ is dangerous and affects $s_{2}, s_{2}$ affects $s_{3}$, and $s_{3}$ affects $s_{4}$, the shape names $s_{1}$ to $s_{4}$ are dangerous. The figure below shows the shape dependency graph with its dangerous shapes and odd cycle in red.


Extended modules For consistent sets of constraints and data graphs, we want to ensure cautious and brave soundness and completeness, and for inconsistent ones, we want to ensure brave completeness and cautious soundness. For both cautious and brave validation, the completeness property means that if $G$ validates $(C, T)$, then $G$ validates ( $\left.C_{\text {magic }}, T\right)$, and soundness means that if $G$ validates ( $C_{\text {magic }}, T$ ), then $G$ validates ( $C, T$ ). Clearly, inconsistent inputs trivially cautiously validate the shapes graph, and it may be that $G$ bravely validates $\left(C_{\text {magic }}, T\right)$. Thus brave soundness and cautious completeness cannot be ensured for inconsistent inputs.

In the presence of unrestricted negation in constraints, it may not be sufficient to consider reachability of shape names only from
left to right as for the positive case. This may leave out some relevant dangerous constraint which may be triggered by shape names that are reachable from the target, and which can only be included by propagating from right to left. It is important to detect and appropriately include such constraints in the Magic Shapes algorithm. We illustrate the problem with the following example.

Example 4.3. Consider the data graph $G=\{r(a, a)\}$ and shapes graph $(C, T)$, where $T=\left\{s_{4}(a)\right\}$ and $C$ is the set of constraints from Example 4.2. Note that for simplicity of presentation in this particular example the constraints are not in normal form, but can be easily normalized. There are 2 supported models of $C$ and $G$, which also coincide with its stable models, namely $I_{1}=\left\{r(a, a), s_{4}(a), s_{5}(a)\right\}$ and $I_{2}=\left\{r(a, a), s_{4}(a), s_{6}(a)\right\}$. Thus $G$ validates $(C, T)$ both bravely and cautiously. Note that $s_{3}(a)$ does not participate in any model since they are filtered out by the first constraint.

Now let us consider the shapes graph resulting from the Magic Shapes algorithm. The only shape name that is reachable from the target shape name $s_{4}$ is $s_{3}$, and hence, the module $C_{T}$ contains only the constraints for these shape names, namely $s_{3} \leftarrow \exists r . \neg s_{4}$ and $s_{4} \leftarrow \exists r . \neg s_{3}$, which will participate in the algorithm. The result is the set $C_{\text {magic }}$ consisting of the constraints:

$$
\begin{aligned}
\text { magic_s }_{4} & \leftarrow a, \\
\text { magic_s } & \leftarrow \exists r^{-} . \text {magic_s } s_{4}, \\
\text { magic_s } & \leftarrow \exists r^{-} . \text {magic_s } s_{3} \\
s_{3} & \leftarrow \text { magic_s } s_{3} \wedge \exists r . \neg s_{4}, \\
s_{4} & \leftarrow \text { magic_s } s_{4} \wedge \exists r . \neg s_{3}
\end{aligned}
$$

The only models of $C_{\text {magic }}$ are

$$
\begin{aligned}
& I_{1}^{\prime}=\left\{r(a, a), \text { magic } s_{-} s_{3}(a), \text { magic } \_s_{4}(a), s_{3}(a)\right\}, \text { and } \\
& I_{2}^{\prime}=\left\{r(a, a), \text { magic_s3}(a), \text { magic_s } s_{4}(a), s_{4}(a)\right\}
\end{aligned}
$$

Of these two models, only $I_{1}^{\prime}$ contains the target shape atom $s_{4}(a)$, so $G$ validates ( $C_{\text {magic }}, T$ ) bravely, but not cautiously.

There are also cases where $G$ bravely validates $\left(C_{\text {magic }}, T\right)$, but not $(C, T)$. For instance, consider again the graph $G$ and the shapes graph $\left(C, T^{\prime}\right)$, where $T^{\prime}=\left\{s_{3}(a)\right\}$. The module $C_{T^{\prime}}$ remains the same as $C_{T}$, and thus the stable models $I_{1}$ and $I_{2}$ remain the same. The target $T^{\prime}$ is neither contained in $I_{1}$ nor in $I_{2}$, therefore $G$ neither bravely nor cautiously validates the shapes graph $\left(C, T^{\prime}\right)$. When applying the magic shape algorithm on $\left(C, T^{\prime}\right)$, the output $C_{\text {magic }}^{\prime}$ contains the same constraints as above, except for the magic seed, which is now magic_s3 $\leftarrow a$. Then the supported and the stable models of $C_{\text {magic }}^{\prime}$ are:

$$
\begin{aligned}
& I_{3}^{\prime}=\left\{r(a, a), \text { magic }^{2} s_{3}(a), \text { magic }_{-} s_{4}(a), s_{4}(a)\right\} \text { and } \\
& I_{4}^{\prime}=\left\{r(a, a), \text { magic_s } s_{3}(a), \text { magic_s } s_{4}(a), s_{3}(a)\right\}
\end{aligned}
$$

Now the target $T^{\prime}$ is contained in $I_{4}^{\prime}$, so $G$ bravely validates the shapes graph $\left(C_{\text {magic }}^{\prime}, T^{\prime}\right)$ although $\left(C, T^{\prime}\right)$ does not.

As illustrated through the above example, on consistent inputs, the current version of the Magic Shapes algorithm for shapes graphs with negation is not cautious-complete and brave-sound. In the above example, the reason for the difference in validation with the original shapes graph is the presence of the second model $I_{1}^{\prime}$ which contains $s_{3}$. Intuitively, the first constraint of $C$ filters out
every model that contain a shape atom over the shape name $s_{1}$, and together with the second constraint do not allow also shape atoms over $s_{3}$. In fact, the first and second constraint are both dangerous.

To address this issue and to ensure both brave and cautious validation (soundness and completeness) for consistent inputs, we extend the notion of reachable shape names and module to take into consideration dangerous constraints. If they are relevant for validating the target, reachability should be propagated also from the body to the head.

Definition 4.4. [Extended reachable set, module] Given a shapes graph $(C, T)$, exReach $(C, T)$ is the set of extended reachable shape names in $C$ from shape names in $T$ such that:
(i) if $s(a) \in T$, then $s \in \operatorname{exReach}(C, T)$,
(ii) if $s(a) \in T$ and there exists a directed path in $D G_{C}$ from $s$ to $s^{\prime}$, then $s^{\prime} \in \operatorname{exReach}(C, T)$, and
(iii) if $s \in \operatorname{exReach}(C, T), s^{\prime}$ is dangerous and there exists a directed path in $D G_{C}$ from $s^{\prime}$ to $s$, then $s^{\prime} \in \operatorname{exReach}(C, T)$.
The set of constraints $C_{T}=\{s \leftarrow \phi \in C \mid s \in \operatorname{exreach}(C, T)\}$ from $C$ is called a module of $C$ w.r.t. $T$.

Complete Algorithm for SHACL with negation We now describe how to adapt the algorithm to take into account the effect of dangerous constraints. The only function that is extended is Generate, which will produce more constraints for dangerous constraints. Algorithm 1 will remain the same.

```
function Generate \((s \leftarrow \phi)\) is
    if \(\phi=\geq_{n} E . s^{\prime}\) then
        return magic_s \(s^{\prime} \leftarrow \geq_{n} E^{-}\).magic_s;
    else
        foreach shape name \(s^{\prime}\) in \(\phi\) do
            return magic_ \(s^{\prime} \leftarrow\) magic_s;
        end
    end
    if \(s \leftarrow \phi\) is dangerous then
        if \(\phi=\geq_{n} E . s^{\prime}\) then
            return magic_s \(\leftarrow \geq_{n}\) E.magic_s';
        else
            foreach shape name \(s \neq s_{i}\) in \(\phi\) do
                        return magic_s \(\leftarrow\) magic_si;
            end
        end
end
```

Algorithm 3: Generation

Roughly, $C_{\text {generate }}$ now contains for each dangerous constraints a set of 'magic' constraints that consider all possible alternations of the shape names in the head and the body of the constraint (see Algorithm 3 lines 9-16). These constraints propagate in assignments the mapping of nodes also from the shape names in the body to the shape name in the head of constraints.

Example 4.5. Consider again Example 4.3. The new $C_{\text {magic }}^{\prime \prime}$ resulting from the complete Magic Shapes algorithm extends the set of constraints $C_{\text {magic }}$ from Example 4.3 with the following constraints
generated by the extended procedure Generate from Algorithm 3:

$$
\begin{array}{rlrl}
\text { magic_s } s_{2} & \leftarrow \text { magic_s } s_{1} & & \text { magic_s } s_{1} \leftarrow \text { magic_s } \\
\text { magic_s } s_{3} & \leftarrow \text { magic_s } s_{2} & \text { magic_s } s_{2} \leftarrow \text { magic_s3 } \\
\text { magic_s3 } & \leftarrow \exists r . \text { magic_s } s_{4} & \text { magic_s } s_{4} \leftarrow \exists r . \text { magic_s3 } \\
s_{1} & \leftarrow \text { magic_s } s_{1} \wedge \neg s_{1} \wedge s_{2} & s_{2} \leftarrow \text { magic_ } s_{2} \wedge s_{3}
\end{array}
$$

Note that $C_{\text {magic }}^{\prime \prime}$ does not contain the constraints for $s_{5}$ and $s_{6}$ since they are not dangerous constraints. Indeed, they are not relevant for validating the target. The unique stable model of $C_{\text {magic }}^{\prime \prime}$ is $I=\left\{r(a, a)\right.$, magic_s $s_{4}(a)$, magic_s $_{3}(a)$, magic_s $s_{2}(a)$, magic_s $s_{2}(a)$, $\left.s_{4}(a)\right\}$.

Hence, $G$ both cautiously and bravely validates $\left(C_{\text {magic }}^{\prime \prime}, T\right)$ as it does with the input shapes graph $(C, T)$.

## 5 CORRECTNESS OF THE ALGORITHM

For a given data graph $G$ and a shapes graph $(C, T)$, the Magic Shapes algorithm outputs a new shapes graph $\left(C_{\text {magic }}, T\right)$, where $C_{\text {magic }}$ is composed of $C_{\text {generate }}$ and $C_{\text {modified }}$. Recall that the constraints in $C_{\text {generate }}$ define the magic version of the adorned shape names and $C_{\text {modified }}$ modifies constraints from the input $C$ by adding in the body the magic version of the shape name that is in the head of the constraint. Provided that $G$ is consistent with $C$, we want to ensure that $G$ validates $(C, T)$ iff $G$ validates $\left(C_{\text {magic }}, T\right)$. Otherwise, we want to guarantee cautious soundness and brave completeness. To this aim, we start with a simple fact:

Lemma 5.1. Let $G$ be a data graph and let $(C, T)$ be a shapes graph. Then there exists a unique assignment $A$ such that $G \cup A$ is a stable model of $C_{\text {generate }}$.

This holds because all the constraints in $C_{\text {generate }}$ are positive. Analogously to normal logic programs, the stable model $M$ of $C_{\text {generate }}$ is the unique, minimal assignment that is a supported model. Intuitively, $M$ identifies the relevant universe of shape atoms for validation, that is, it 'marks' with magic shapes all nodes of the graph that can participate in the validation of the target $C_{\text {modify }}$ under both the stable and supported model semantics. We will see that a shapes graph and its magic output coincide on the shape atoms that appear in $M$.

To show the correctness of the algorithm, we will show that $C_{\text {magic }}$ defines a kind of module that is independent from the rest of $C$ and $G$. To this aim we rely on the notion of ground constraints, which intuitively instantiate a constraint at a node or a pair of nodes.

Definition 5.2. A ground constraint is a tuple $(\rho, v)$, if $\rho$ is a constraint of the form (NF1)-(NF5) and a tuple $\left(\rho,\left(v, v^{\prime}\right)\right)$, if $\rho$ is a constraint of the form (NF6), where $v, v^{\prime}$ are nodes from N . We may say $(\rho, v)$ (or $\left(\rho,\left(v, v^{\prime}\right)\right)$ ) is the grounding of $\rho$ w.r.t. $v\left(\right.$ or $\left.\left(v, v^{\prime}\right)\right)$.

Next we define the notion of the grounding of a set of constraints w.r.t. an input data graph $G$.

Definition 5.3. Given a data graph $G$ and a set of constraints $C$, the grounding $\operatorname{gr}(C, G)$ of $C$ w.r.t. $G$ is a set of ground constraints obtained for each $\rho \in C$, such that

- if $\rho$ is of the form (NF1)-(NF5), then $(\rho, v)$ is in $C_{g r}$ for each $v \in V(G)$,
- if $\rho$ is of the form (NF6), then $\left(\rho,\left(v, v^{\prime}\right)\right)$ is in $C_{g r}$ for each $v, v^{\prime} \in V(G)$.

We lift the definition $\pi_{s}$ of $s$ in a set of constraints $C$ to ground constraints, and then adapt the definition of supported and stable models to ground constraints as follows.

Definition 5.4. A decorated graph $I$ is a supported model of a set of ground constraints $C_{g r}$ if $\llbracket \phi_{s} \rrbracket^{I}=\llbracket s \rrbracket^{I}$ for each shape name $s$ occurring in $C_{g r}$ and for each $v \in \llbracket s \rrbracket^{I}$ there exists in $C_{g r}$ a ground constraint $(\rho, v)$ or $\left(\rho,\left(v, v^{\prime}\right)\right)$ with $s$ in the head of $\rho$. $I$ is a stable model of $C_{g r}$ if it is a supported model of $C_{g r}$, and there exists a level assignment such that level $\left(\phi_{s}, v\right)<\operatorname{level}(s, v)$ for all $s(v) \in I$.

From the above definitions, the following lemma immediately holds. In particular, this is the case since the grounding $\operatorname{gr}(C, G)$ of a set of constraints $C$ w.r.t. a data graph $G$ grounds every constraint in $C$ with every node or every pair of nodes from $V(G)$.

Lemma 5.5. Let $G$ be a data graph and let $C$ be a set of constraints. Then I is a supported (or stable) model of C iff I is a supported (or stable) model of $\operatorname{gr}(C, G)$.

Independent constraints We now define the notion of independence of a set of ground constraints w.r.t. another set of ground constraints, which is similar to the same notion for ground ASP programs defined by Eiter et al. in [11] and to splitting sets defined by Lifschitz et al. in [17].

Definition 5.6. Given a set of ground constraints $C_{g r}$, the shape domain of $C_{g r}$, written as sdom $\left(C_{g r}\right)$, is the set of shape atoms $s(v)$ such that one of the following holds:

- $(\rho, v)$ is in $C_{g r}$ and $s$ occurs in the body or head of $\rho$, or
- $\left(\rho,\left(v, v^{\prime}\right)\right)$ is in $C_{g r}$ and $s$ is the head of $\rho$, or
- $\left(\rho,\left(v^{\prime}, v\right)\right)$ is in $C_{g r}$ and $s$ is in the body of $\rho$.

Let $C_{1}$ and $C_{2}$ be sets of ground constraints. We say that $C_{1}$ is independent of $C_{2}$ if it is not the case that there is some shape name $s$ that occurs in the head of some ground constraint $(\rho, v)$ or of the form $\left(\rho,\left(v, v^{\prime}\right)\right)$ in $C_{2}$ such that $s(v)$ occurs in $\operatorname{sdom}\left(C_{1}\right)$.

The above definition roughly states that $C_{1}$ is independent of $C_{2}$ if each 'ground' shape name that occurs in the head of a ground constraint in $C_{2}$ does not occur in any ground constraint (head or body) in $C_{1}$, hence the ground constraints in $C_{1}$ may have some effect on the ground constraints of $C_{2}$, but this is not the case for the other direction.

Theorem 5.7. Let $G$ be a data graph and let $C_{g r}=g r(C, G)$ be the grounding of $C$ w.r.t. G. Moreover, let $C_{1}$ and $C_{2}$ be such that $C_{g r}=C_{1} \cup C_{2}$ and $C_{1}$ is independent of $C_{2}$. Let $M=\operatorname{sdom}\left(C_{1}\right)$. Then, for every supported (or stable) model $I=G \cup A$ of $C_{g r}$ it is the case that $G \cup(A \cap M)$ is a supported (or stable) model of $C_{1}$.

Proof. We show the claim first for the supported model semantics and then for supported models with a level assignment. Let $I=G \cup A$ be a supported model of $C_{g r}$ and let $I_{M}=G \cup(A \cap M)$. We need to show that $I_{M}$ is a supported model of $C_{1}$, that is: 1) that $\llbracket \phi_{s} \rrbracket^{I_{M}}=\llbracket s \rrbracket^{I_{M}}$ for each shape name $s$ occurring in $C_{1}$, and 2) for each $v \in \llbracket s \rrbracket^{I_{M}}$ there exists in $C_{1}$ a ground constraint $(\rho, v)$ or $\left(\rho,\left(v, v^{\prime}\right)\right)$ with $s$ in the head of $\rho$. To see the latter, note that $s(v)$
is in $M$, so there is some constaint with $s$ in the head or the body and $v$ in the corresponding position, as in Definition 5.6. Note that it cannot occur in a body only, since there must be some constraint with $s$ in the head and grounded with $v$ in either of the $C_{i}$, but as $C_{1}$ is independent from $C_{2}$, it cannot be in $C_{2}$.

To show 1), we first note that since $C_{1}$ is independent from $C_{2}$, for each node $v \in \llbracket s \rrbracket^{I_{M}}$ it is the case that all the ground constraints $(\rho, v)$ or $\left(\rho,\left(v, v^{\prime}\right)\right)$ with $s$ in the head of $\rho$ that appear in $C_{g r}$ will also appear in $C_{1}$. Thus there is a unique $\phi_{s}$ that is the definiton of $s$ in both $C_{g r}$ and $C_{1}$, which is in fact its definiton in $C$. Now, let $s$ be an arbitrary shape name in $C_{1}$ and let $v$ be an arbitrary node in $V(G)$. It is left to show that $v \in \llbracket \phi_{s} \rrbracket^{I_{M}}$ if and only if $v \in \llbracket s \rrbracket^{I_{M}}$. For the $(\Rightarrow)$ direction, $v \in \llbracket \phi_{s} \rrbracket^{I_{M}}$ implies $v \in \llbracket \phi_{s} \rrbracket^{I}$. Since $I$ is a supported model of $C_{g r}$, it follows that $v \in \llbracket s \rrbracket^{I}$. That is, there is some ground constraint $(\rho, v)$ or $\left(\rho,\left(v, v^{\prime}\right)\right)$ with $s$ in the head of $\rho$ that appears in $C_{g r}$, and hence, it also appears in $C_{1}$. It follows that $s(v)$ is also in $M$, and hence also in $I_{M}$. For the $(\Leftarrow)$ direction, let $s(v) \in I_{M}$. This implies that $s(v) \in A$ and $s(v) \in M$. The first implies that $v \in \llbracket \phi_{s} \rrbracket^{I}$. Let $\phi$ be an arbitrary disjunct in $\phi_{s}$ such that $v \in \llbracket \phi \rrbracket^{I}$. It is left to show that $v \in \llbracket \phi \rrbracket^{I_{M}}$. The claim trivially holds for $\phi$ as in (NF1), (NF2) or (NF3). If $\phi$ is of the form $\neg s^{\prime}$, as in (NF4), then $s^{\prime}(v)$ must not be in $I$ and hence it is not in $A$, which implies that $v \in \llbracket \neg s^{\prime} \rrbracket^{I_{M}}$. If $\phi$ is a conjunction of shape names $s_{1}, \ldots s_{n}$, as in (NF4), then each $s_{i}(v)$ is in $A$. Since $\left(s \leftarrow s_{1} \wedge \cdots \wedge s_{n}, v\right)$ is in $C_{g r}$, then it must also be in $C_{1}$, which implies that each $s_{i}(v)$ is also in $M$, and hence also in $I_{M}$. Finally, for $\phi$ of the form $\geq_{n} E . s^{\prime}$, as in (NF6), reasoning as in the previous case, there must be at least $n$ nodes $v_{i}$ with an $E$-path from $v$ such that $s^{\prime}\left(v_{i}\right)$ in $A$. Since the grounding is with respect to every node in $V(G)$, for each such $v_{i}$ there is a ground constraint $\left(s \leftarrow \geq_{n} E . s^{\prime},\left(v, v_{i}\right)\right)$ in $C_{g r}$, and since $C_{1}$ is independent of $C_{2}$, each such ground constraint is also in $C_{1}$. Hence, by the definition of the shape domain of $C_{1}, s^{\prime}\left(v_{i}\right)$ is in $M$ for each such node $v_{i}$. From these observations follows that $s^{\prime}\left(v_{i}\right)$ is in $A \cap M$ for each $v_{i}$, which shows that $v$ satisfies $\geq_{n} E . s^{\prime}$ in $I_{M}$.

For the stable model semantics, it is clear that if $I$ has a level assignment, then $I_{M} \subseteq I$ also has a level assignment.

Independence of Magic Constraints We show that to compute the models of a set of constraints $C_{\text {magic }}$ resulting from the magic algorithm, it suffices to consider the models of the grounding of the module $C_{T}$ obtained by grounding every constraint with the nodes of the shapes atoms with 'magic' as prefix from $C_{\text {generate }}$.

Lemma 5.8. Let $G$ be a data graph, let $(C, T)$ be a SHACL shapes graph and let $G \cup M$ be the stable model of $C_{\text {generate }}$. For each shape name s such that magic_s appears in $M$ we let $N_{s}=\{v \mid$ magic_s $(v) \in M\}$. Let $C_{T, g r}$ be the set of ground constraints obtained by grounding every constraint $\rho$ in $C_{T}$ as follows:

- if $\rho$ is of the form (NF1) to (NF5), then $(s \leftarrow \phi, v) \in C_{T, g r}$ for each $v \in N_{s}$,
- if $\rho$ is of the form $s \leftarrow \geq_{n}$ E.s', that is (NF6), then ( $s \leftarrow$ $\left.\phi,\left(v, v^{\prime}\right)\right) \in C_{T, g r}$ for each $v, v^{\prime}$ with $v \in N_{s}$ and $v^{\prime} \in N_{s^{\prime}}$,
Then, the following hold under supported and stable model semantics:
(1) if I is a model of $C_{\text {magic }}$, then $I \backslash M$ is a model of $C_{T, g r}$, and
(2) if $I$ is a model of $C_{T, g r}$, then $I \cup M$ is a model of $C_{\text {magic }}$.

Proof. We use the term model instead of supported or stable model. For (1), let $I$ be a model of $C_{\text {magic }}$ and let $s$ be an arbitrary shape name occurring in $C_{\text {modified }}$. Then, it is the case that $\llbracket \phi_{s} \wedge$ magic_s $\rrbracket \rrbracket^{I}=\llbracket s \rrbracket^{I}$ and $\llbracket s \rrbracket^{I} \subseteq N_{s}$. Clearly, $s \leftarrow \phi_{s}$ is also the definition of $s$ in $C_{T}$ and by construction, for each $v \in N_{s}$ and for each constraint $s \leftarrow \phi$ in $C_{T}$, there is a ground constraint $(s \leftarrow \phi, v)$ or $\left(s \leftarrow \phi,\left(v, v^{\prime}\right)\right)$ in $C_{T, g r}$. By the definition of models of ground constraints, it follows that $I \backslash M$ is also a model of $C_{T, g r}$. For (2), let $I$ be a model of $C_{T, g r}$. Then, $\llbracket \phi_{s} \rrbracket^{I}=\llbracket s \rrbracket^{I}$ and $\llbracket s \rrbracket^{I} \subseteq N_{s}$. Since $N_{s}$ contains all the nodes appearing in atoms over magic_s in $M$, follows that $I \cup M$ is a model for $C_{\text {generate }}$ and for $C_{\text {modified }}$. For the stable model semantics, the existence of a level assignment for $I \backslash M$ in item (1) is trivial. For (2), it is left to show that if $I=G \cup A$ has a level assignment, then $I \cup M$ has a level assignment. Since $G \cup M$ is the stable model of $C_{\text {generate }}$, there is a level assignment for the atoms with magic as prefix such that level $\left(\phi_{\text {magic }}, v\right)<\operatorname{level}($ magic_s $s)$ for every magic_s $(v) \in M$, with magic_s $\leftarrow \phi_{\text {magic }}$ a constraint in $C_{\text {generate }}$. Let $L_{G}$ be such an assignment, where the atoms from $G$ are assigned integer 0 , and let $\ell$ be the integer of the highest ranked shape atom in $L_{G}$. Furthermore, let $L^{\prime}$ be a level assignment for $I$. We define $L_{M}$ which assigns to each shape atom $\alpha$ in $A$ an integer $i+\ell$, where $i$ is the level assigned to the atom $\alpha$ in $L^{\prime}$. Clearly, the level assignment composed of $L_{G}$ and $L_{M}$ is as desired.

The set of ground constraints $C_{T, g r}$ obtained from $C_{\text {magic }}$ is independent of the rest of the grounding.

Lemma 5.9. Let $G$ be a data graph and let $C_{g r}=g r(C, G)$ be the grounding of $C$ w.r.t. $G$. Then, $C_{T, g r}$ is independent of $C_{g r} \backslash C_{T, g r}$.

Proof. Towards a contradiction, let $s$ be a shape name that occurs in the head of a constraint $\rho$ with $(\rho, v)$ (or $\left(\rho,\left(v, v^{\prime}\right)\right)$ ) in $C_{g r} \backslash C_{T, g r}$ and assume $s(v) \in \operatorname{sdom}\left(C_{T, g r}\right)$. By Lemma 5.8 follows that magic_s $s(v)$ is in $M$, where $G \cup M$ is the stable model of $C_{\text {generate }}$, and there is some constraint with $s$ in the head or body that appears ground in $C_{T, g r}$ with $v$ in the expected position. It follows that the constraint $\rho$ should also appear in $C_{T}$ since $s$ will also be adorned by the algorithm and by Lemma 5.8 this constraint should also have been grounded in $C_{T, g r}$ with the node $v$, that is $(\rho, v)$ (or $\left.\left(\rho,\left(v, v^{\prime}\right)\right)\right)$ should be in $C_{T, g r}$, thus deriving a contradiction to the initial assumption.

Before we present the main lemma that puts everything together, we need to adapt the notion of dangerous constraints for ground constraints by considering shape atoms rather than shape names.

Definition 5.10. Assume a set of ground constraints $C_{g r}$. The ground shape dependency graph of $C_{g r}$ is a marked directed graph $G D G_{C_{g r}}$, where there is an edge from node $s_{1}(v)$ to node $s_{2}(v)$ if there is a ground constraint $\left(s_{2} \leftarrow \phi, v\right)$ in $C$ such that $s_{1}$ occurs in $\phi$, an edge from node $s_{1}\left(v^{\prime}\right)$ to node $s_{2}(v)$ if there is a ground constraint ( $s_{2} \leftarrow \phi,\left(v, v^{\prime}\right)$ ) in $C_{g r}$ such that $\phi$ is of the form $\geq E_{n} \cdot s_{1}$, and the edge is marked if $\phi=\neg s_{2}$. Cycles and odd cycles of $G D G_{C_{g r}}$ are defined exactly as for $D G_{C}$. Let $C_{g r}$ be a set of ground constraints. A shape atom $s(v)$ is dangerous if (1) $s(v)$ occurs in an odd cycle of $G D G_{C_{g r}}$, or (2) $s$ occurs in $\phi$ for some constraint $\left(s^{\prime} \leftarrow \phi, v\right) \in C_{g r}$ or $\left(s^{\prime} \leftarrow \phi,\left(v^{\prime}, v\right)\right)$ such that $s^{\prime}$ is dangerous. A ground constraint $\left(s^{\prime} \leftarrow \phi, v\right)$ or $\left(s^{\prime} \leftarrow \phi,\left(v, v^{\prime}\right)\right.$ is dangerous if $s^{\prime}(v)$ is dangerous.

We observe that if a ground constraint $(\rho, v)$ or $\left(\rho,\left(v, v^{\prime}\right)\right)$ in $g r(C, G)$ is dangerous, then $\rho$ is also dangerous in $C$.

Lemma 5.11. Let $G$ be a data graph and let $(C, T)$ a SHACL document. Assume $G \cup M$ is the stable model of $C_{\text {generate }}$ and $M^{\prime}=\{s(v) \mid$ magic_s $(v) \in M\}$. Then, the following hold under supported- and stable model semantics:
(1) if $I=G \cup A$ is a model of $C$, then $I^{\prime}=G \cup M \cup A^{\prime}$ is a model of $C_{\text {magic }}$, where $A^{\prime}=A \cap M^{\prime}$, and
(2) if $C$ is consistent with $G$ and $I^{\prime}=G \cup M \cup A^{\prime}$ is a model of $C_{\text {magic }}$, then there exists an $A$ such that $I=G \cup A$ is a model of $C$ and $A^{\prime}=A \cap M^{\prime}$.

Proof. For item (1), we show the claim for supported models and then for models with a level assignment. By Lemma 5.5 it follows that $I$ is a model of $C_{g r}$. By Lemma 5.9 follows that $C_{T, g r}$ is independent of $C_{g r} \backslash C_{T, g r}$, where $C_{T}$ is the module of $C$ w.r.t. $T$ and $C_{T, g r}$ is the grounding of $C_{T}$ w.r.t. the stable model $M$ as defined in Lemma 5.11. The latter and Theorem 5.7, item (1), imply that $G \cup\left(A \cap M^{\prime}\right)$ is a model of $C_{T, g r}$, which together with Lemma 5.8, item (2), imply that $I^{\prime}=G \cup\left(A \cap M^{\prime}\right) \cup M$ is a model of $C_{\text {magic }}$. For the stable model semantics, it is left to show that $I^{\prime}$ has a level assignment if $I$ has a level assignment. This is done analogously to the proof of Lemma 5.8 item (2), arguing that we can combine the level assignments for $G \cup M$ and $I$ into a level assingment for $I^{\prime}$.

For item (2), similarly as above, we first show the claim for supported models and argue about the exsitence of a level assignment. Let $I^{\prime}=G \cup M \cup A^{\prime}$ be a model of $C_{\text {magic }}$. By Lemma 5.8, item (1), follows that $I^{\prime} \backslash M$ is a model of $C_{T, g r}$. Let $C_{A^{\prime}}$ be the set of ground constraints $((s \leftarrow v), v)$ for each $v$ such that $s(v) \in A^{\prime}$. Since $C_{T, g r}$ is independent from $\left(C_{g r} \backslash C_{T, g r}\right)$, then we know that we can replace $C_{T, g r}$ by each of its models, and the models of $C_{g r}$ are preserved. That is, if there exists some $A$ such that $I=G \cup A$ is a model of $C_{g r}$ with $A^{\prime}=A \cap M^{\prime}$, then $I$ would also a model of $\left(C_{g r} \backslash C_{T, g r}\right) \cup C_{A^{\prime}}$. Clearly, $A^{\prime}$ would be contained in $I$. So, it is only left to show that such an $A$ exists, which we argue similarly as in the proof of Theorem 3.4 in [13]: in a nutshell, we show that there is no constraint participating in an odd cycle in $C^{\prime}=C_{g r} \backslash C_{T, g r}$ that will 'kill' the assignment $A^{\prime}$ of $C_{T, g r}$.

To this aim, let $C_{o d d}$ denote the set of all the ground constraints in $C^{\prime}$ that are dangerous, that is $C_{o d d}$ contains all the ground constraints from $C^{\prime}$ with a dangerous head that is in an odd cycle, reachable from an odd cycle, or reaches an odd cycle. It follows that the shape domain sdom $\left(C_{o d d}\right) \cap \operatorname{sdom}\left(C^{\prime} \backslash C_{o d d}\right)=\emptyset$. Moreover, we show that $\operatorname{sdom}\left(C_{o d d}\right) \cap \operatorname{sdom}\left(C_{T, g r}\right)=\emptyset$, which implies that $\operatorname{sdom}\left(C_{o d d}\right) \cap \operatorname{sdom}\left(C_{A^{\prime}}\right)=\emptyset$. By Definition 5.6 , there is no ground constraint $(\rho, v)$ or $\left(\rho,\left(v, v^{\prime}\right)\right)$ in $C_{o d d}$ with $s$ in the head of $\rho$ such that $s(v) \in \operatorname{sdom}\left(C_{T, g r}\right)$ since $C_{T, g r}$ is independent of $C^{\prime}$ and $C_{o d d}$ is contained in $C^{\prime}$. Assume towards a contradiction that there is a ground constraint $(\rho, v)$ or $\left(\rho,\left(v^{\prime}, v\right)\right)$ in $C_{o d d}$ with $s$ in the body of $\rho$ such that $s(v) \in \operatorname{sdom}\left(C_{T, g r}\right)$. First, we observe that since such a ground constraint in $C_{g r}$ is dangerous, then $\rho$ must also also be dangerous in $C$, and by definition, the shape name $s$ occurring in the body of $\rho$ is also dangerous. Since $s(v) \in \operatorname{sdom}\left(C_{T, g r}\right)$, then by Definition 5.6, there is a constraint $\rho^{\prime}$ such that at least one of the following holds:

- $\left(\rho^{\prime}, v\right)$ is in $C_{T, g r}$ and $s$ occurs in the body or head of $\rho^{\prime}$, or
- $\left(\rho^{\prime},\left(v, v^{\prime}\right)\right)$ is in $C_{T, g r}$ and $s$ is the head of $\rho^{\prime}$, or
- $\left(\rho^{\prime},\left(v^{\prime}, v\right)\right)$ is in $C_{T, g r}$ and $s$ is in the body of $\rho^{\prime}$.

Thus $\rho^{\prime}$ contains a dangerous shape name $s$ in the body or in the head. Hence, by Definition 4.4, $\rho$ should also be present in the module $C_{T}$ and by Definition 5.2 , the ground constraint $(\rho, v)$ or $\left(\rho,\left(v^{\prime}, v\right)\right)$ should also be contained in $C_{T, g r}$ and not in $C_{o d d}$, deriving a contradiction. From the above observations it follows that the shape domain of $C_{o d d}$ has no intersection with the shape domain of the rest of the constraints in $C_{g r}$. Since $C_{g r}$ is consistent, it follows that there exists a model of $C_{o d d}$. Let $G \cup A_{o d d}$ be such a model. Note that the rest of the set of ground constraints, that is $\left(C_{g r} \backslash\left(C_{T, g r} \cup C_{o d d}\right)\right) \cup C_{A^{\prime}}$ does not contain any odd cycles. It is known from answer set programming that programs that do not contain any odd cycle are consistent [10], that is they admit a model. This result can be lifted to SHACL by results from Andresel et al. in [4], which showed that a SHACL shapes graph can be translated into a logic program with negation that preserves validation under the stable and supported model semantics. These transformation does not change the parity of negation in cyclic dependencies of shape names. That is, a shape name $s$ appears in a cycle with an odd number of negative edges if and only if the corresponding unary atom over $s$ in targeted program appears in a cycle with an odd number of negative edges. So, as $\left(C_{g r} \backslash\left(C_{T, g r} \cup C_{o d d}\right)\right) \cup C_{A^{\prime}}$ has no odd cycles, we know it has a model. Let $G \cup A^{\prime} \cup A_{\text {rest }}$ be such a model. It then follows that $G \cup A^{\prime} \cup A_{\text {rest }} \cup A_{o d d}$ is a model of $C_{g r}$. By Lemma 5.5 and since $T \subseteq I^{\prime}$, it follows that the desired model $I$ of $C$ is $I=G \cup A$ where $A=A^{\prime} \cup A_{\text {rest }} \cup A_{\text {odd }}$. Either $A_{\text {odd }}$ nor $A_{\text {res }}$ have any atoms from $M^{\prime}$, so $A^{\prime}=A \cap M^{\prime}$. This finishes the proof for the supported model semantics.

For the stable model semantics, it remains to argue that there is a level assignment for $I$, which is not hard to do, since we know there are level assingment exists for each of $G \cup A_{\text {rest }}, G \cup A_{\text {odd }}$ and $G \cup A$. As $A_{\text {rest }}$ and $A_{\text {odd }}$ are disjoint, and they are both disjoint from $A^{\prime}$, we can combine them into a level assignment for $I$.

Intuitively, if we ignore the 'magic' prefix and the shape atoms with the 'magic' prefix, the above lemma states that, for consistent inputs, the set of models of $G$ and $C$ restricted to the shape atoms resulting from $C_{\text {generate }}$ is equivalent to the set of the models of $G$ and $C_{\text {magic }}$. For inconsistent inputs, there may exist a model of $G$ and $C_{\text {magic }}$ even if there is no assignment that satisfies all the constraints in $C$. This gives us the desired correctness result:

Theorem 5.12. Let $G$ be a data graph and let $(C, T)$ be a shapes graph. The following hold under supported and stable model semantics:
(1) If $G$ bravely validates $(C, T)$, then $G$ bravely validates $\left(C_{\text {magic }}, T\right)$.
(2) If $G$ cautiously validates $\left(C_{\text {magic }}, T\right)$, then $G$ cautiously validates $(C, T)$.
(3) If $C$ is consistent with $G$, then $G$ validates $\left(C_{\text {magic }}, T\right)$ iff $G$ validates $(C, T)$, under both brave and cautious validation.

Proof. We show the proof for stable models; it is identical for supported models. For (1), suppose $I=G \cup A$ is a stable model of $C$ such that $T \subseteq A$. By Lemma 5.11, item 1, it follows that there is an $I^{\prime}$ of the form $G \cup M \cup A^{\prime}$ that is a model of $C_{\text {magic }}$, where $G \cup M$ is the stable model of $C_{\text {generate }}$ and $A^{\prime}=A \cap M^{\prime}$. It is left to show that $T \subseteq A^{\prime}$. This follows by construction of $C_{\text {generate }}$ which contains a
magic seed for each shape atom in the target, and thus, for every $s(v) \in T$, magic $\_s(v)$ is in $M$, and hence, $s(v) \in M^{\prime} \cap A$.

For (2), the claim trivially holds for the case where $C$ is not consistent with $G$. Otherwise, assume $C$ is consistent with $G$ and suppose $G$ cautiously validates $C_{\text {magic }}$. Then, either (a) $C_{\text {magic }}$ is not consistent with $G$, which by Lemma 5.11 item (1), it follows that $C$ is not consistent with $G$, and thus the claim follows, or (b) $C_{\text {magic }}$ is consistent with $G$ and $T \subseteq I^{\prime}$ for each stable model $I^{\prime}$ of $C_{\text {magic }}$. For (b), let $I^{\prime}=G \cup M \cup A^{\prime}$ be an arbitrary stable model of $C_{\text {magic }}$ such that $T \subseteq A^{\prime}$. By Lemma 5.11 item (2), there is a stable model $I=G \cup A$ of $C$ that contains $A^{\prime}$, which implies $T \subseteq A$.

For (3), that $C$ is consistent with $G$ implies $C_{\text {magic }}$ is consistent with $G$. It is left to show (i) if $G$ bravely validates ( $C_{\text {magic }}, T$ ), then $G$ bravely validates ( $C, T$ ), and (ii) if $G$ cautiously validates $(C, T)$, then $G$ cautiously validates $\left(C_{\text {magic }}, T\right)$. Since there is a one to one correspondence between stable models for $C$ and for $C_{\text {magic }}$, arguing as above using Lemma 5.11, item 1 and 2, the claim easily follows for both brave and cautious validation.

## 6 INCONSISTENCY-TOLERANT SEMANTICS

As has been observed in previous sections, the Magic Shapes algorithm may output a shapes graph $\left(C_{\text {magic }}, T\right)$ which is consistent with the data graph although the input shapes graph $(C, T)$ is not. This suggests that the constraints and the part of the data graph causing the inconsistency are not relevant for validating the target. Such situations in the context of SHACL may arise very often due to the large RDF triplestores, which may inevitably, have faulty facts. The 2 -valued semantics that we have considered until now, which requires the existence of a global assignment, may be too restrictive in this setting. Corman et al. [8] address this drawback of the 2 -valued supported model semantics by proposing the notion of faithful assignments, which may leave some shape atoms as undefined. We propose a similar inconsistency-tolerant semantics for the stable- and supported model semantics, based on the models of our magic transformation. To this aim, we first define a new notion of assignments which may contain negated shape atoms.

Definition 6.1. A literal assignment $A$ for a graph $G$ is a set of shape atoms of the form $s(v)$ or $\neg s(v)$ where $s \in \mathrm{~S}$ and $v \in V(G)$.

If for a shape name $s$ appearing in $C$ and a node $v \in V(G)$ we have $\{s(v), \neg s(v)\} \cap A=\emptyset$, then we say that $s(v)$ is undefined in $A$.

Definition 6.2. Assume a set of shape constraints $C$ and a literal assignment $A$ for $G$. Then, $I=G \cup A$ is called a 3-valued supported model of $C$ if for each $v \in V(G)$ and for each constraint $s \leftarrow \phi$ in $C$ :

- if $s(v) \in I$, then $v \in \llbracket \phi \rrbracket^{I}$, and
- if $\neg s(v) \in I$, then $v \notin \llbracket \phi \rrbracket^{I}$.

If there is a level assignment for $I$, then $I$ is 3 -valued stable model of $C$. A graph $G$ validates a shapes graph $(C, T)$ under the 3-valued supported model semantics (or 3-valued stable model semantics) if there exists a 3 -valued supported model (or 3-valued stable model) $I$ of $C$ such that $T \subseteq A$.

Note that each supported (or stable) model $G \cup A$ of a set of constraints $C$ can be seen as a 3 -valued supported (or stable) model $G \cup A^{\prime}$ where no shape atom remains undefined: simply let $A^{\prime}$ extend $A$ with the negated version of each shape atom $s(v) \notin A$ with $s$ a shape name appearing in $C$ and $v$ a node in $V(G)$.

While 3-valued validation is highly desirable, to our knowledge no algorithm that allows to decide the existence of a 3-valued stable model for a given $(C, T)$ that may contain unrestricted recursion and negation had been proposed until now, neither under the supported nor under the stable semantics. Our magic transformation, however, is an effective way to do 3 -valued validation. Indeed, we can see the models of the magic as 3 -valued models where only the relevant atoms identified by $C_{\text {generate }}$ are defined to be true or false, and where all the remaining atoms, which play no role in the validation of the target, are left undefined.

Lemma 6.3. Let $G$ be a data graph, let $C$ be a shapes graph, and let $I^{\prime}=G \cup M \cup A^{\prime}$ be a supported (stable) model of $C_{\text {magic }}$, where $G \cup M$ is the stable model of $C_{\text {generate }}$. Let $M^{\prime}=\{s(v) \mid$ magic_ $s(v) \in M\}$. Then, $I=G \cup A$ is a 3-valued supported (stable) model of $C$, where

$$
A=A^{\prime} \cup\left\{\neg s(v) \mid s(v) \in M^{\prime} \backslash A^{\prime}\right\}
$$

Proof. In order to show that $I=G \cup A$ is a 3-valued supported model we need to show that (i) if $s(v) \in I$, then $v \in \llbracket \phi \rrbracket^{I}$, and (ii) if $\neg s(v) \in I$, then $v \notin \llbracket \phi \rrbracket^{I}$. Let $I^{\prime}=G \cup M \cup A^{\prime}$ be a supported model of $C_{\text {magic }}$, it follows that $\llbracket s \rrbracket^{I^{\prime}}=\llbracket$ magic $\_s \rrbracket^{I^{\prime}} \cap \llbracket \phi \rrbracket^{I^{\prime}}$ for each constraint $s \leftarrow$ magic_s $\wedge \phi \in C_{\text {magic }}$. Note that the evaluation of shape expressions is not affected by additional negated shape atoms (see Table 1), hence the equality $\llbracket \phi \rrbracket^{I}=\llbracket \phi \rrbracket^{I^{\prime}}$ holds. For positive shape atoms $s(v) \in I$, we know that $s(v) \in A^{\prime}$, which implies that $v \in \llbracket \phi \rrbracket^{I}$. For the negated shape atoms of the form $\neg s(v) \in A$, we know that $s(v)$ is in $M^{\prime}$ but not in $A^{\prime}$, which implies that $v \notin \llbracket \phi \rrbracket^{I}$.

To prove it for the 3 -valued stable model semantics it is left to show that $I$ has a level assignment such that for all $s(v) \in I$ it holds that level $\left(\phi_{s}, a\right)<\operatorname{level}(s, a)$. Let $I^{\prime}$ be a stable model of $C_{\text {magic }}$, then it has a level assignment where level $\left(\right.$ magic_s $\left.\wedge \phi_{s}\right)=$ $\max \left(\left\{l e v e l(\right.\right.$ magic_s $)$, level $\left.\left.\left(\phi_{s}\right)\right\}\right)<$ level $(s, v)$ for all $s(v) \in I^{\prime}$. From this observation follows that $\operatorname{level}\left(\phi_{s}\right)<\operatorname{level}(s, v)$ for all $s(v) \in I^{\prime}$. By construction of $I$ every $s(v)$ in $I$ is also in $I^{\prime}$, therefore for all $s(v) \in I$ it holds that level $\left(\phi_{s}\right)<\operatorname{level}(s, v)$.

Theorem 6.4. Consider a data graph $G$ and a shapes graph ( $C, T$ ). If $G$ validates ( $C_{\text {magic }}, T$ ) under the supported (stable) model semantics, then $G$ validates $(C, T)$ under the 3-valued supported (stable) model semantics.

Proof. Let $I=G \cup M \cup A^{\prime}$ be a stable model of $C_{\text {magic }}$ and $T \subseteq A$. Then $G$ is valid against a shapes graph $(C, T)$ under the 3 -valued supported model semantics (or stable model semantics) if (i) $I$ is a 3 -valued supported- or stable model and (ii) $T \subseteq A$. By Theorem 6.3 (i) holds, since $G \cup A$ is a 3 -valued supported model (or stable model) of $C$, where $A=A^{\prime} \cup\left\{\neg s(v) \mid s(v) \in M^{\prime} \backslash A^{\prime}\right\}$ and $M^{\prime}=\{s(v) \mid$ magic_s $(v) \in M\}$, and (ii) holds by assumption.

The converse does not hold, as we show in the following example. Our magic technique provides a stronger inconsistency-tolerant validation than arbitrary 3 -valued models.

Example 6.5. Consider $(C, T)$ and $G$, where $C=\left\{s \leftarrow s^{\prime} \wedge\right.$ $\left.\neg s, s^{\prime} \leftarrow \exists r\right\}, T=\left\{s^{\prime}(a)\right\}$ and $G=\{r(a, a)\}$. The set $G \cup\left\{s^{\prime}(a)\right\}$ is a 3 -valued stable and supported model that validates the target, but it is not a 2 -valued stable or supported model of $C_{\text {magic }}$. Since $s$ occurs in an odd cycle, the constraint $\rho=s \leftarrow s^{\prime} \wedge \neg s$ is dangerous, and since $s^{\prime}$ occurs in its body, by Definition 4.4, $\rho$ also occurs in

Table 2: Constraints used in our experiments. We use $=_{n} r$ to abbreviate $\geq_{n} r \wedge \leq_{n} r$, and $\exists r$ to abbreviate $\exists r$.T. $C_{1}=\left\{\quad\right.$ Actor $\leftarrow$ Person $\wedge\left(\exists\right.$ starring ${ }^{-}$.Movie $\vee \exists$ occupation.actor $) \quad$ Famous $\leftarrow \exists$ knownFor

Director $\leftarrow$ Person $\wedge \exists$ director ${ }^{-}$. Movie
Location $\leftarrow \exists$ country
Movie $\leftarrow={ }_{1}$ imdbId $\wedge \exists$ starring.Actor
TranslatedMovie $\leftarrow$ Movie $\wedge \geq_{2}$ language.LivingLanguage
Person $\leftarrow={ }_{1}$ birthPlace. Location
WorkingPerson $\leftarrow$ Person $\wedge$ Employee
WorkingClass $\leftarrow$ Person $\wedge \exists$ child $^{-}$.WorkingParent
$C_{2}=\left(C_{1} \backslash\{\right.$ Employee $\leftarrow$ ヨemployer $\left.\}\right) \cup\{$ Employee $\leftarrow$ Eemployer $\neg$ Employee $\}$

Table 3: Results of experiments with the shacl-asp validator for $C_{1}$ and $C_{2}$, and targets $T$ over single nodes or classes with a number of nodes specified in the second column. For $C_{1}$, the column ' valid $_{2,3}$ ' indicates the number of valid nodes under both 2 - and 3 -valued semantics (the same before and after magic). For $C_{2}$, the 2 -valued semantics always fails, and 'valid ${ }_{3}$ ' shows the number of targeted nodes that are validated under 3-valued semantics using magic. Evaluation times before magic $t$ and after magic $t_{m}$ are measured in seconds. For each $i \in\{1,2\},\left|C_{i m}\right|$ is the number of constraints after magic and $N_{i m}$ is the number of graph nodes appearing in the magic shape names in the model of $C_{i \text { generate }}$.

| Targets |  | $C_{1}$ |  |  |  |  | $C_{2}$ |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $T$ | nodes | valid $_{2,3}$ | $t$ | $t_{\text {m }}$ | $\left\|C_{1 m}\right\|$ | $N_{1 \mathrm{~m}}$ | valid ${ }_{3}$ | $t$ | $t_{\mathrm{m}}$ | $\left\|C_{2 \mathrm{~m}}\right\|$ | $N_{2 \mathrm{~m}}$ |
| Musician( mozart) | 1 | 0 | 783 | 243 | 6 | 1 | 0 | 860 | 231 | 6 | 1 |
| Musician(Actor) | 4.872 | 0 | 958 | 253 | 6 | 6908 | 0 | 957 | 240 | 6 | 6908 |
| Musician(Person) | 2267445 | 5248 | 894 | 410 | 6 | 2396852 | 5248 | 936 | 407 | 6 | 2396852 |
| Actor( cameron) ${ }^{(1)}$ | 1 | 1 | 768 | 384 | 8 | 3 | 1 | 894 | 404 | 8 | 3 |
| Actor(Actor) | 4872 | 614 | 905 | 450 | 8 | 253606 | 614 | 929 | 441 | 8 | 253606 |
| Actor(Person) | 2267445 | 16645 | 975 | 620 | 8 | 2567841 | 16645 | 933 | 642 | 8 | 2567841 |
| Movie(Film) | 143121 | 2449 | 882 | 474 | 8 | 302030 | 2449 | 999 | 448 | 8 | 302030 |
| TranslatedMovie( Film ) | 143121 | 5 | 849 | 491 | 12 | 303021 | 5 | 905 | 508 | 12 | 303021 |
| Employee(bill) | 1 | 1 | 868 | 42 | 2 | 1 | nm | 936 | 45 | 4 | 3 |
| Employee (mark) | 1 | 0 | 785 | 39 | 2 | 1 | 0 | 831 | 43 | 4 | 1 |
| Employee(Actor) | 4872 | 6 | 825 | 42 | 2 | 4872 | 6 | 977 | 44 | 4 | 4879 |
| Employee(Person) | 2267445 | 9290 | 868 | 86 | 2 | 2267445 | nm | 880 | 145 | 4 | 2273847 |
| WorkingPerson( bill) | 1 | 1 | 846 | 241 | 8 | 2 | nm | 981 | 259 | 10 | 2 |
| WorkingPerson(mark) | 1 | 0 | 795 | 234 | 8 | 2 | 0 | 879 | 249 | 10 | 2 |
| WorkingPerson(Actor) | 4872 | 2 | 864 | 250 | 8 | 6908 | 2 | 997 | 294 | 10 | 6915 |
| WorkingPerson(Person) | 2267445 | 2959 | 906 | 410 | 8 | 2396852 | nm | 970 | 488 | 10 | 2403194 |
| WorkingClass(bill) | 1 | 0 | 734 | 258 | 14 | 2 | 0 | 890 | 283 | 16 | 2 |
| WorkingClass( mark) ${ }^{(2)}$ | 1 | 1 | 822 | 291 | 14 | 7 | 1 | 942 | 285 | 16 | 7 |
| WorkingClass(Actor) | 4872 | 0 | 906 | 278s | 14 | 7055 | 0 | 979 | 313 | 16 | 7056 |
| WorkingClass(Person) | 2267445 | 40 | 887 | 445 | 14 | 2396883 | 40 | 985 | 588 | 16 | 2397188 |

$C_{T}$, and hence, the constraint $s \leftarrow$ magic_s $\wedge s^{\prime} \wedge \neg s$ will be in the output of the magic algorithm. Since there is no consistent way to assign or not assign node $a$ to shape name $s$, there is no 2 -valued stable or supported model of $C_{\text {magic }}$.

## 7 IMPLEMENTATION AND EXPERIMENTS

We implemented a prototype of the Magic Shapes algorithm and analysed its effect on the performance of Shacl-Asp, the only existing validator for full SHACL. The source code of the prototype and all files used for the experiments are available online. ${ }^{2}$

Setting. The experiments were performed on a Linux server with a 24 core Intel Xeon CPU running at 2.20 GHz and 264 GB of RAM.

[^2]We used the Shacl-Asp validator with an Apache Jena TDB ${ }^{3}$ as an RDF triple store to access the data graph.
Data Graph. The data for the experiments was obtained from DBPedia (latest-core, version 2021-12) ${ }^{4}$. More precisely, we used the datasets 'PersonData', 'Instance Types', 'Labels', 'Mappingbased Literals' and 'Mappingbased Objects'. The datasets are in Turtle syntax and contain about 119 million triples ( 15.4 GB ). The ShaclAsp validator extracts from this graph all triples that mention the data predicates in the shapes graphs, and transforms them into a set $G_{\text {asp }}$ of ASP facts. For our test cases, $G_{\text {asp }}$ has almost 7 million facts and 4 million constants (graph nodes). To assess our inconsistency-tolerant validation, we added a few facts to $G_{\text {asp. }}$. Specifically, we added employer (bill, bob), employer (bob, jim), and

[^3]employer(jim, bill) to create an employer cycle over three individuals, we also added a birthPlace for each of them.

Shapes Graph. We created two sets of shape constraints describing persons and their professions, in the spirit of the test cases in [7]. We show them-in abstract syntax-in Table 2. Note that $C_{2}$ can result in inconsistency if the data graph has cycles over the relation employer. $C_{1}$ and $C_{2}$ were combined with different targets to obtain 40 different shapes graphs. We are using class targets: the target $s(C)$ is a shortcut for the set of targets containing $s(v)$ for each node $v$ that has type $C$ in $G$. The number of nodes in the graph that are targeted in our shapes graphs ranges from one to over two million, see the second column of Table 3.

Results. Table 3 summarizes the results of our experiments. The average time (in seconds) of running the validator five times on the original shapes graph and five times on its magic variant for each test case is shown in the $t$ and $t_{\mathrm{m}}$ columns. The Magic Shapes algorithm improves the performance of the DLV validator significantly, by at least $30 \%$ in our test cases.

The constraints in $C_{1}$ are recursive: there is a cycle between shape names Actor and Movie. As $C_{1}$ is always consistent-the mild form of negation in $C_{1}$ does not break consistency-the number of targeted nodes that are validated under the 2 - and 3 -valued semantics coincides, independently of the magic. Their total can be seen in the column 'valid ${ }_{2,3}$ '.

In contrast, the additional constraint with negation in $C_{2}$ is dangerous, as it creates a marked self-loop to the dangerous shape name Employee in the dependency graph $D G_{C_{2}}$ of $C_{2}$. In fact, $C_{2}$ is inconsistent with $G_{\text {asp }}$ independently of any targets $T$ : due to the mentioned employer cycle, there is no way to neither assign nor omit Employee for each of bill, bob and jim, and there is no stableor supported model. As a consequence, none of the targeted nodes can be validated with the 2 -valued semantics. However, using magic, we can validate under the 3 -valued semantics all expected nodes for the 16 targets that are not related to these individuals being or not employees, see column 'valid 3 '. The magic versions of $C_{2}$ over the shape names Musician, Actor, Movie, or TranslatedMovie coincide with the corresponding versions of $C_{1}$, and they are positive. This is reflected in the results in Table 3, which are identical, and only $t$ and $t_{m}$ may be slightly higher for $C_{2}$ as it has an additional constraint. For the rest of the targets, the dangerous constraint participates in $C_{2 \mathrm{~m}}$ (hence there are two more constraints than in $C_{1 \mathrm{~m}}$, one for $C_{2 \text { generate }}$ and one for $C_{2 \text { modified }}$. We remark that the constraint for Employee will not participate in the magic version in $C_{1 \text { generate }}$, but the new version will be in $C_{2 \text { generate }}$. This explains why $N_{2 m}$ may be slightly larger than $N_{1 m}$. However, the dangerous constraint will only cause inconsistency if bill, bob, or jim is adorned with magic_Employee by $C_{2 \text { generate }}$. In our tests, this is the case for Employee( bill), Employee(Person), WorkingPerson(bill), and WorkingPerson(Person). For the rest of the targets in Table 3, the dangerous constraint will behave as the positive constraint for Employee in $C_{1}$, validating the same number of nodes.

Finally, we observe that in some cases recursion may eliminated by the magic algorithm (e.g., for $C_{1}$ with a Musician target, since the shape names responsible for recursion are not reachable), or negation (e.g., for $C_{2}$ with Musician, Actor, Movie, and TranslatedMovie

Table 4: Experiments on reduced data graphs. The times in the first colum are for validating the original $C_{i}$ over $G_{\text {asp }}$, and in the second column for the magic version $C_{i_{m}}$ over $G_{i \mathrm{~m}}$.

|  | $C, G_{\text {asp }}$ | $C_{\mathrm{m}}, G_{i \mathrm{~m}}$ | $N_{i \mathrm{~m}}$ | $\left\|G_{i \mathrm{~m}}\right\|$ |
| :---: | :---: | :---: | :---: | :---: |
| $C_{1},(1)$ | 768 s | 0.13 s | 3 | 461 |
| $C_{1},(2)$ | 822 s | 1.87 s | 7 | 26069 |
| $C_{2},(2)$ | 942 s | 2.31 s | 7 | 26069 |

targets). In principle, this could enable the use other validators for SHACL fragments, like Shacl2Sparql and Trav-Shacl. ${ }^{5}$

Unfortunately, the current shacl-AsP does not easily allow us to fully leverage the fact that the subgraph identified as relevant may be several order of magnitudes smaller, and it loads the full $G_{\text {asp }}$ with its millions of nodes even when only a handful of them are relevant in the validation. To explore the potential for further reducing the data graph, we hand-picked three shape graphs with single nodes as targets, and dropped from $G_{\text {asp }}$ all facts not mentioning nodes in $N_{i m}$. Even though the resulting $G_{i m}$ may still contain thousands of facts (since for example, some locations may occur in thousands of birth place facts), the execution times dropped by a couple of orders of magnitude. This observation suggests that it would be promising to optimize sHACL-ASP to work with a better approximation of $G_{i m}$.

## 8 CONCLUSIONS AND OUTLOOK

Adapting the Magic Set technique, we have proposed a magic transformation for unrestricted SHACL validation. It takes as input a shapes graph $(C, T)$ and produces a new ( $\left.C_{\text {magic }}, T\right)$. The new $C_{\text {magic }}$ not only discards shape constraints that are irrelevant for validating the target, but also identifies the relevant parts of the data graph, thus enabling a target-oriented validation. The new shapes graph can be equivalently used for validation on every graph that is consistent with $C$. In case of inconsistency, $C_{\text {magic }}$ may still admit partial, target-oriented validation. To our knowledge, no practical approaches for such inconsistency tolerant validation were available to date. We implemented our magic transformation and ran experiments with Shacl-Asp, the only SHACL validator that can support unrestricted recursion and negation, and showed that the simpler shapes graphs produced by the Magic Shapes algorithm are evaluated significantly more efficiently than the original ones.

We are currently exploring possible ways to exploit the Magic Shapes technique in a more efficient and robust SHACL validator. Specifically, we want to develop an improved SHACL validator which does not require to load huge graphs as sets of facts, but instead uses a SPARQL end-point (in the style of Shacl2Spargl and Trav-Shacl) to access only the potentially much smaller data graph that the magic algorithm identifies as relevant.

## ACKNOWLEDGMENTS

This work was funded in whole or in part by the Faculty of Informatics of TU Wien, the Vienna Business Agency, the Austrian Science Fund (FWF) projects P30360 and P30873, and the FWF and netidee SCIENCE project T 1349-N.

[^4]
## REFERENCES

1] Weronika T. Adrian, Mario Alviano, Francesco Calimeri, Bernardo Cuteri, Carmine Dodaro, Wolfgang Faber, Davide Fuscà, Nicola Leone, Marco Manna, Simona Perri, Francesco Ricca, Pierfrancesco Veltri, and Jessica Zangari. 2018 The ASP System DLV: Advancements and Applications. Künstliche Intell. 32, 2-3 (2018), 177-179. https://doi.org/10.1007/s13218-018-0533-0
[2] Mario Alviano and Wolfgang Faber. 2011. Dynamic magic sets and super-coherent answer set programs. AI Communications 24, 2 (2011), 125-145.
[3] Mario Alviano, Wolfgang Faber, Gianluigi Greco, and Nicola Leone. 2012. Magic sets for disjunctive datalog programs. Artificial Intelligence 187 (2012), 156-192
[4] Medina Andresel, Julien Corman, Magdalena Ortiz, Juan L. Reutter, Ognjen Savkovic, and Mantas Šimkus. 2020. Stable Model Semantics for Recursive SHACL. In Proc. of The Web Conference 2020. ACM, 1570-1580. https://doi.org/ 10.1145/3366423.3380229
[5] Franz Baader, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. 2017. An Introduction to Description Logic. Cambridge University Press. http: //www.cambridge.org/de/academic/subjects/computer-science/knowledge-management-databases-and-data-mining/introduction-descriptionlogic?format=PB\#17zVGeWD2TZUeu6s. 97
[6] F. Bancilhon, D. Maier, Y. Sagiv, and J. Ullman. 1985. Magic sets and other strange ways to implement logic programs (extended abstract). In PODS '86.
[7] Julien Corman, Fernando Florenzano, Juan L. Reutter, and Ognjen Savkovic. 2019. Validating Shacl Constraints over a Sparql Endpoint. In ISWC. Springer. https://doi.org/10.1007/978-3-030-30793-6_9
[8] Julien Corman, Juan L. Reutter, and Ognjen Savkovic. 2018. Semantics and Validation of Recursive SHACL. In Proc. of ISWC'18. Springer. https://doi.org/10. 1007/978-3-030-00671-6_19
[9] Chiara Cumbo, Wolfgang Faber, Gianluigi Greco, and Nicola Leone. 2004. Enhancing the Magic-Set Method for Disjunctive Datalog Programs. In Logic Programming, 20th International Conference, ICLP (Lecture Notes in Computer Science), Bart Demoen and Vladimir Lifschitz (Eds.). Springer. https://doi.org/10.1007/978-3-540-27775-0_26
[10] Phan Minh Düng. 1992. On the Relations between Stable and Well-Founded Semantics of Logic Programs. Theor. Comput. Sci. 105, 1 (1992), 7-25. https:
//doi.org/10.1016/0304-3975(92)90285-N
[11] Thomas Eiter, Georg Gottlob, and Heikki Mannila. 1997. Disjunctive Datalog. 22, 3 (1997), 364-418. https://doi.org/10.1145/261124.261126
[12] Thomas Eiter, Giovambattista Ianni, and Thomas Krennwallner. 2009. Answer Set Programming: A Primer. In Reasoning Web. Semantic Technologies for Information Systems, 5th International Summer School 2009, Brixen-Bressanone, Italy, August 30 - September 4, 2009, Tutorial Lectures (Lecture Notes in Computer Science), Vol. 5689. Springer, 40-110. https://doi.org/10.1007/978-3-642-03754-2_2
[13] Wolfgang Faber, Gianluigi Greco, and Nicola Leone. 2007. Magic Sets and their application to data integration. F. Comput. Syst. Sci. (2007), 584-609. https: //doi.org/10.1016/j.jcss.2006.10.012
[14] Mónica Figuera, Philipp D. Rohde, and Maria-Esther Vidal. 2021. Trav-SHACL: Efficiently Validating Networks of SHACL Constraints. In WWW. ACM / IW3C2, 3337-3348. https://doi.org/10.1145/3442381.3449877
[15] José Emilio Labra Gayo, Eric Prud'hommeaux, Iovka Boneva, and Dimitris Kontokostas. 2017. Validating RDF Data. Morgan \& Claypool Publishers. https://doi.org/10.2200/S00786ED1V01Y201707WBE016
[16] Martin Leinberger, Philipp Seifer, Tjitze Rienstra, Ralf Lämmel, and Steffen Staab. 2020. Deciding SHACL Shape Containment Through Description Logics Reasoning. In Proc. of ISWC 2020 (Lecture Notes in Computer Science), Vol. 12506. Springer, 366-383. https://doi.org/10.1007/978-3-030-62419-4_21
[17] Vladimir Lifschitz and Hudson Turner. 1999. Splitting a Logic Program. (01 1999).
[18] Inderpal Singh Mumick, Sheldon J. Finkelstein, Hamid Pirahesh, and Raghu Ramakrishnan. 1990. Magic is Relevant. In Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, Hector Garcia-Molina and H. V. Jagadish (Eds.). ACM Press, 247-258. https://doi.org/10.1145/93597.98734
[19] Christos H. Papadimitriou and Mihalis Yannakakis. 1997. Tie-Breaking Semantics and Structural Totality. 7. Comput. Syst. Sci. 54 (1997), 48-60.
[20] Paolo Pareti, George Konstantinidis, Fabio Mogavero, and Timothy J. Norman. 2020. SHACL Satisfiability and Containment. In Proc. of ISWC 2020. Springer. https://doi.org/10.1007/978-3-030-62419-4_27
[21] Kenneth A. Ross. 1994. Modular Stratification and Magic Sets for Datalog Programs with Negation. 7. ACM 41, 6 (1994). https://doi.org/10.1145/195613.195646


[^0]:    This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit https://creativecommons.org/licenses/by-nc-nd/4.0/ to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.
    Proceedings of the VLDB Endowment, Vol. 15, No. 10 ISSN 2150-8097.
    doi:10.14778/3547305.3547329

[^1]:    ${ }^{1}$ https://www.w3.org/TR/shacl/\#targets

[^2]:    ${ }^{2}$ https://github.com/biziloehnert/magicSHACL.

[^3]:    ${ }^{3}$ https://jena.apache.org/documentation/tdb/
    ${ }^{4}$ https://www.dbpedia.org/resources/latest-core/

[^4]:    ${ }^{5}$ We could not run the output of magic on these prototypes since their parsers appear to be incomplete and do not support constraints like $s \leftarrow s^{\prime}$.

