Fixpoint Semantics for Recursive SHACL

Bart Bogaerts

Vrije Universiteit Brussel, Belgium

`bart.bogaerts@vub.be`

Maxime Jakubowski

Universiteit Hasselt, Belgium

Vrije Universiteit Brussel, Belgium

`maxime.jakubowski@uhasselt.be`

SHACL is a W3C-proposed language for expressing structural constraints on RDF graphs. The recommendation only specifies semantics for *non-recursive* SHACL; recently, some efforts have been made to allow *recursive* SHACL schemas. In this paper, we argue that for defining and studying semantics of recursive SHACL, lessons can be learned from years of research in non-monotonic reasoning. We show that from a SHACL schema, a three-valued semantic operator can directly be obtained. Building on *Approximation Fixpoint Theory* (AFT), this operator immediately induces a wide variety of semantics, including a supported, stable, and well-founded semantics, related in the expected ways. By building on AFT, a rich body of theoretical results becomes directly available for SHACL. As such, the main contribution of this short paper is providing theoretical foundations for the study of recursive SHACL, which can later enable an informed decision for an extension of the W3C recommendation.

1 Introduction

The *Semantic Web* [2] extends the World-Wide Web with machine-interpretable data. The de facto standard of this web is that data is stored and published in the Resource Description Framework [22], i.e., as a set of triples, referred to as a *graph*, often extended with semantic information expressed in OWL [20]. While in principle the Semantic Web is open and every agent can represent their data however they want, from the perspective of applications, or when consuming RDF data in some other way, it can often be useful to know which structural properties an RDF graph in question satisfies. In other words, there is a need for a declarative language for describing *integrity constraints* on RDF graphs.

Several proposals have emerged to fill this need, the most prominent of them being ShEx [9] and SHACL [23]. Both approaches start from the notion of a *shape*: a structural property that a node in an RDF graph can satisfy, e.g., the shape of "things that have a name" will be satisfied for those nodes that appear as the subject in a triple with predicate `foaf:name`. On top of a language for defining such shapes, the two proposals also have a mechanism for *targeting*: for specifying which nodes *should* satisfy which shapes, e.g., to declare that "All persons should satisfy the has-a-name shape".

In our paper, we will build on a formalization of SHACL [12], which has revealed a striking similarity between shapes and concepts descriptions, as known from description logics [3]; we recently deepened this connection further [7]. Following this line of research, we formalize a SHACL schema as a tuple (Def, T), where Def is a set of rules of the form $s \leftarrow \phi$ and T is a set of concept inclusions of the form $\phi \subseteq s$ with ϕ a shape (i.e., a concept description) and s a shape name. Def defines the shapes in terms of relations in the RDF graph and T contains the so-called *targeting* constraints.

The W3C recommendation for SHACL only specifies the semantics when Def is *non-recursive*, i.e., if no shape is defined in terms of itself, but recently, some efforts have been made to lift this restriction, i.e., to define a supported and a stable semantics for *recursive* SHACL [12, 1].

In this paper, we put forward another principled way to define semantics of recursive SHACL, building on *Approximation Fixpoint Theory* (AFT), an abstract lattice-theoretic framework originally designed to unify semantics of non-monotonic logics [14] with applications, among others, in (extensions of) logic programming, autoepistemic logic, default logic, abstract argumentation, and active integrity constraints [15, 21, 24, 19, 11, 5]. There are several advantages to defining semantics of SHACL in this way:

- It is *simple* and *straightforward*: the power of AFT, comes largely from the fact that all that is required to apply it, is to define a (three-valued) semantic operator (similar to Fitting's immediate consequence operator for logic programs [18]). In many domains (including SHACL), there is a natural choice for such an operator; AFT then immediately induces all major classes of semantics.
- It provides *confidence*: AFT guarantees that the developed semantics follow well-established principles in nonmonotonic reasoning. Even in case semantics are already defined, applying AFT can be a *sanity check*. A striking example of this is the fact that applications of AFT uncovered some issues in the semantics of (weighted and non-weighted) Abstract Dialectic Frameworks [24, 10, 4].
- It provides access to a *large body of theoretical results*, including theorems on *stratification* [27, 6], *predicate introduction* [28], and *strong equivalence* [25], thereby eliminating the need to "reinvent the wheel" by rediscovering these results in each of the separate domains.

In a nutshell, our main contribution is establishing formal foundations for the study of recursive SHACL.

2   Formalization of SHACL

In *actual* SHACL, semantics is defined in terms of RDF graphs, but we recently showed how to reduce this to the logical setting [7]. As such, for the purpose of the current paper, we focus on the logical setting and take abstraction of RDF graphs. Throughout this paper, we (implicitly) fix a *vocabulary*, i.e., a set of *node names* (denoted N), *property names* (denoted P), and *shape names* (denoted S). A node name c is also referred to as a *constant*, a property name p as a *binary predicate symbol* and a shape name s as a *unary predicate symbol*. *Path expressions* E and *shapes* ϕ are defined as follows:

$$E ::= p \mid p^- \mid E \cup E \mid E \circ E \mid E^* \mid E?$$
$$\phi ::= \top \mid s \mid \{c\} \mid \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid \forall E.\phi \mid \geq_n E.\phi \mid eq(E,E) \mid disj(E,E) \mid closed(Q)$$

where p, c, and s represent property names, node names, and shape names respectively, and n stands for non-zero natural numbers. When developing our semantics, we will treat $\phi_1 \vee \phi_2$ as an abbreviation of $\neg(\neg\phi_1 \wedge \neg\phi_2)$ and $\forall E.\phi$ as an abbreviation for $\neg \geq_1 E.\neg\phi$. The reason for having them in the syntax is to enable the comparison with existing semantics in Section 5.

As usual, an interpretation I consists of **(i)** a set Δ^I, called the *domain* of I; **(ii)** for each constant c, an element $[\![c]\!]^I \in \Delta^I$; **(iii)** for each shape name s, a set $[\![s]\!]^I \subseteq \Delta^I$; and **(iv)** for each property name p, a binary relation $[\![p]\!]^I$ on Δ^I. When we say a *graph*-interpretation, we mean an interpretation that only consists of a domain and an interpretation for node and property names (not shape names). Intuitively, such an interpretation corresponds to an RDF graph. A path expression E evaluates in I to a binary relation $[\![E]\!]^I$ on Δ^I, and a shape ϕ to a subset $[\![\phi]\!]^I$ of Δ^I, as defined in Tables 1 and 2.

A SHACL *schema* is a tuple (Def, T) with

- *Def* a set of rules of the form $s \leftarrow \phi$ with s a shape name (referred to as the *head* of the rule) and ϕ a shape (the *body* of the rule), such that each $s \in S$ appears exactly once in the head of a rule, and
- *T* a set of (concept) inclusions of the form $\phi \subseteq s$, with ϕ a shape that does not mention any shape names[1], and s a shape name.

---

[1]The condition that ϕ does not mention any shape names will entail that the target query can be evaluated on the underlying

E	$[\![E]\!]^I$
p^-	$\{(a,b) \mid (b,a) \in [\![p]\!]^I\}$
$[\![E_1 \cup E_2]\!]^I$	$[\![E_1]\!]^I \cup [\![E_2]\!]^I$
$[\![E_1 \circ E_2]\!]^I$	$\{(a,b) \mid \exists c : (a,c) \in [\![E_1]\!]^I$
	$\wedge (c,b) \in [\![E_2]\!]^I\}$
$[\![E^*]\!]^I$	the reflexive-transitive
	closure of $[\![E]\!]^I$
$[\![E?]\!]^I$	$[\![E]\!]^I \cup \{(a,a) \mid a \in \Delta^I\}$

Table 1: Semantics of a path expression E in an interpretation I.

ϕ	$[\![\phi]\!]^I$
$\top$	Δ^I
$\{c\}$	$\{c^I\}$
$\phi_1 \wedge \phi_2$	$[\![\phi_1]\!]^I \cap [\![\phi_2]\!]^I$
$\neg\phi_1$	$\Delta^I \setminus [\![\phi_1]\!]^I$
$\geq_n E.\phi_1$	$\{a \in \Delta^I \mid \sharp([\![\phi]\!]^I_1 \cap [\![E]\!]^I(a)) \geq n\}$
$eq(E_1, E_2)$	$\{a \in \Delta^I \mid [\![E_1]\!]^I(a) = [\![E_2]\!]^I(a)\}$
$disj(E_1, E_2)$	$\{a \in \Delta^I \mid [\![E_1]\!]^I(a) \cap [\![E_2]\!]^I(a) = \emptyset\}$
$closed(Q)$	$\{a \mid [\![p]\!]^I(a) = \emptyset$ for every $p \in P \setminus Q\}$

Table 2: Semantics of a shape ϕ in an interpretation I. We use $\sharp X$ to denote the cardinality of X. For a binary relation R and an element a, we use $R(a)$ to denote $\{b \mid (a,b) \in R\}$.

A shape name s_1 *depends* on shape name s_2 in *Def* if there is a rule $s_1 \leftarrow \phi$ in *Def*, and s_2 or some shape name that depends on s_2 occurs in ϕ. A schema is *non-recursive* if no shape name depends on itself.

If (Def, T) is a *non-recursive* SHACL schema, and I a graph-interpretation, then I can be uniquely extended to an interpretation I' such that for each $s \in S$, $[\![s]\!]^{I'} = [\![\phi]\!]^{I'}$ if $s \leftarrow \phi \in Def$. In that case, we say that I *validates* with respect to (Def, T) if $[\![\phi]\!]^{I'} \subseteq [\![s]\!]^{I'}$ for each inclusion $\phi \subseteq s$ in T. For the *recursive* case, the situation is less obvious; in Section 4, we use approximation fixpoint theory to study the different semantic options that arise, but before doing so, we recall preliminaries on AFT.

3 Approximation Fixpoint Theory

A *complete lattice* $\langle L, \leq \rangle$ is a set L equipped with a partial order $\leq$, such that every set $S \subseteq L$ has a least upper bound and a greatest lower bound. A complete lattice has a least element $\bot$ and a greatest element $\top$. An operator $O : L \to L$ is *monotone* if $x \leq y$ implies that $O(x) \leq O(y)$. An element $x \in L$ is a *fixpoint* of O if $O(x) = x$. Every monotone operator O in a complete lattice has a least fixpoint, denoted lfp(O).

Given a lattice L, AFT uses a bilattice L^2. We define *projections* for pairs as usual: $(x,y)_1 = x$ and $(x,y)_2 = y$. Pairs $(x,y) \in L^2$ are used to approximate elements in the interval $[x,y] = \{z \mid x \leq z \wedge z \leq y\}$. We call $(x,y) \in L^2$ *consistent* if $x \leq y$, that is, if $[x,y]$ is non-empty. We use L^c to denote the set of consistent elements. The *precision order* on L^2 is defined as $(x,y) \leq_p (u,v)$ if $x \leq u$ and $v \leq y$. If (u,v) is consistent, this means that (x,y) approximates all elements approximated by (u,v).

In its original form, AFT makes use of *approximators*, which are operators on L^2, but [16] showed that all the consistent fixpoints studied in AFT are uniquely determined by an approximator's restriction to L^c and developed a theory of *consistent approximators*. An operator $A : L^c \to L^c$ is a *consistent approximator* of O if it is $\leq_p$-monotone and *coincides with O on L* (meaning $A(x,x) = (O(x), O(x))$ for all $x \in L$). AFT studies fixpoints of O using fixpoints of A.

- The *A-Kripke-Kleene fixpoint* is the $\leq_p$-least fixpoint of A; it approximates all fixpoints of O.
- A *partial A-stable fixpoint* is a pair (x,y) such that $x = $ lfp($A(\cdot, y)_1$) and $y = $ lfp($A(x, \cdot)_2$), where $A(\cdot, y)_1 : L \to L$ maps z to $A(z,y)_1$ and similarly for $A(x, \cdot)_2$.
- The *A-well-founded fixpoint* is the least precise ($\leq_p$-least) partial A-stable fixpoint.
- An *A-stable fixpoint* of O is a fixpoint x of O such that (x,x) is a partial A-stable fixpoint.

---

graph, without knowledge of the shape definitions. In actual SHACL, the condition is even more limited; there only very specific queries are allowed as targets.

These definitions allow reconstructing all major logic programming semantics by taking for O Van Emden and Kowalski's immediate consequence operator T_P [26] and for A Fitting's three- (or four-) valued extension Ψ_P [18].

4   Fixpoint Semantics for Recursive SHACL

For the rest of this paper, we fix a SHACL schema (Def, T) and a graph-interpretation I. We already mentioned that if Def is non-recursive, it uniquely induces a complete interpretation I' in which all constraints in T are to be verified. When Def is recursive, however, the situation becomes more complex. On the one hand, there is a range of possible semantics dealing with recursion. On the other hand, some of the semantics yield not a single interpretation I', but either a set of them, or a *three-valued* interpretation. This will give us a choice between *brave* and *cautious* validation; the focus of this paper is on the treatment of negation, but we briefly discuss brave and cautious validation below.

To apply AFT, the first step to take is to determine a suitable lattice. In our case, the obvious candidate is the lattice L_I (from now on, denoted L) of all interpretations I' with domain Δ^I that agree with I on $N \cup P$, or in other words, the set of interpretations that expand I. This set is equipped with the standard truth order, $I_1 \leq_t I_2$ if $[\![s]\!]^{I_1} \subseteq [\![s]\!]^{I_2}$ for all $s \in S$. The role of the semantic operator is to update the value of the interpretation of the shapes. In analogy with logic programming, its definition is straightforward: it maps the interpretation I' to $T_{Def}(I')$ such that for each shape name s with defining rule $s \leftarrow \phi$, we define $T_{Def}(I')(s) = [\![\phi]\!]^{I'}$.

With the lattice $\langle L, \leq_t \rangle$, elements of L^c are pairs $\mathscr{I} = (\mathscr{I}_1, \mathscr{I}_2)$ of two interpretations with $\mathscr{I}_1 \leq_t \mathscr{I}_2$; such pairs correspond one-to-one to *three-valued interpretations* that assign each $s \in S$ a function $\Delta \rightarrow \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$, mapping a to $\mathbf{t}$ if a in $[\![s]\!]^{\mathscr{I}_1}$, to $\mathbf{f}$ if $a \notin [\![s]\!]^{\mathscr{I}_2}$ and to $\mathbf{u}$ otherwise (in other words, $\mathscr{I}_1$ represents what is *certainly true* and $\mathscr{I}_2$ what is *possibly true*). From now on, we simply refer to elements of L^c as three-valued interpretations.

We can evaluate a shape ϕ in a three-valued interpretation $\mathscr{I}$ with a straightforward extension of Kleene's truth tables, as also used in previous studies of recursive SHACL [12, 1]; for completeness, this is included in Table 3. This table makes use of the truth order $\leq_t$ on truth values defined as $\mathbf{f} \leq_t \mathbf{u} \leq_t \mathbf{t}$, and the negation on truth values defined as usual: $\neg\mathbf{t} = \mathbf{f}; \neg\mathbf{f} = \mathbf{t}; \neg\mathbf{u} = \mathbf{u}$.

ϕ	$[\![\phi]\!]^{\mathscr{I}}(a)$
$\top$	$\mathbf{t}$
$\{c\}$	$\mathbf{t}$ if $a = c$; $\mathbf{f}$ otherwise
s	$[\![s]\!]^{\mathscr{I}}(a)$
$\neg\phi_1$	$\neg[\![\phi_1]\!]^{\mathscr{I}}(a)$
$\phi_1 \wedge \phi_2$	$\min_{\leq_t}([\![\phi_1]\!]^{\mathscr{I}}(a), [\![\phi_2]\!]^{\mathscr{I}}(a))$
$\geq_n E.\phi_1$	$\begin{cases} \mathbf{t} & \text{if } \sharp\{b \in [\![E]\!]^{\mathscr{I}}(a) \mid [\![\phi_1]\!]^{\mathscr{I}}(b) = \mathbf{t}\} \geq n, \\ \mathbf{f} & \text{if } \sharp\{b \in [\![E]\!]^{\mathscr{I}}(a) \mid [\![\phi_1]\!]^{\mathscr{I}}(b) \leq_t \mathbf{u}\} < n, \\ \mathbf{u} & \text{otherwise} \end{cases}$
$eq(E_1, E_2)$	$\mathbf{t}$ if $a \in [\![eq(E_1, E_2)]\!]^{\mathscr{I}}$; $\mathbf{f}$ otherwise
$disj(E_1, E_2)$	$\mathbf{t}$ if $a \in [\![disj(E_1, E_2)]\!]^{\mathscr{I}}$; $\mathbf{f}$ otherwise
$closed(Q)$	$\mathbf{t}$ if $[\![p]\!]^{\mathscr{I}}(a) = \emptyset$ for all $p \in P \setminus Q$; $\mathbf{f}$ otherwise

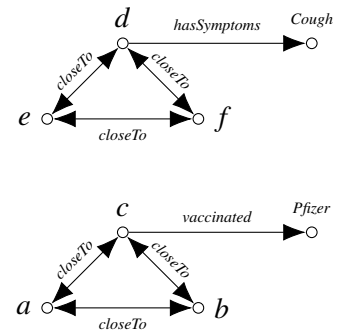Table 3: Three-valued semantics of shapes.

Figure 1: Visual representation of the example interpretation.

Once a three-valued evaluation of shapes is defined, an approximator is obtained directly: like the

operator, the approximator updates the value of each shape symbol according to its defining rule: it maps $\mathscr{I}$ to $\Psi_{Def}(\mathscr{I})$ where for each shape $s \in S$ defined by the rule $s \leftarrow \phi$, $\Psi_{Def}(\mathscr{I})(s) = [\![\phi]\!]^{\mathscr{I}}$.

Theorem 4.1. Ψ_{Def} *is a consistent approximator of* T_{Def}.

At this point, AFT dictates what the supported models (fixpoints of T_{Def}), (partial) stable models (Ψ_{Def}-stable fixpoints), well-founded model (Ψ_{Def}-well-founded fixpoint), and Kripke-Kleene model (Ψ_{Def}-KK fixpoint) of *Def* are. It is worth stressing that to arrive to this point, we made two choices. The first was our choice of order on the lattice. We opted here for the truth order, but its inverse would also have been a possible choice. Several of the semantics induced by AFT aim to minimize models in the chosen order for reasons of *groundedness* [8], e.g., if *s* has $s \leftarrow s$ as defining rule, in stable and well-founded semantics, our chosen order would result in no nodes satisfying *s*. The second choice we made is *which three-valued truth evaluation to use*; we opted for the most obvious choice: a direct extension of Kleene's three-valued truth tables, which was used in other studies of recursive SHACL as well [12, 1]. Given these choices, models of the different types (stable, well-founded, ...) are defined by AFT, and hence semantics for brave and cautious validation under of each semantics are established.

Definition 4.2. Let $\sigma \in \{KK, WF\}$ and let $\mathscr{I}$ be the σ-model. We say that *I cautiously (resp. bravely)* σ-*validates* with respect to (Def, T) if $[\![\phi \wedge \neg s]\!]^{\mathscr{I}} = \mathbf{f}$ (resp. $[\![\phi \wedge \neg s]\!]^{\mathscr{I}} \leq_t \mathbf{u}$) for every $\phi \subseteq s$ in T.
Let $\sigma \in \{St, Sup\}$ and let M be the set of σ-models. We say that *I cautiously (resp. bravely)* σ-*validates* with respect to (Def, T) if $[\![\phi \wedge \neg s]\!]^{I'} = \mathbf{f}$ for all (resp. for some) $I' \in M$ for every $\phi \subseteq s$ in T.

Let us illustrate the differences between the various types of models on a small example.

Example 4.3. Consider binary predicates *closeTo*, *hasSymptoms*, and *vaccinated* and an interpretation *I* with domain $\{a, b, c, d, e, f, Pfizer, Cough\}$, where $a, \ldots, f$ represent people (divided in two cliques of three "close" friends); one person (*c*) is vaccinated and one person (*d*) shows Covid symptoms. This interpretation is visually depicted in Fig. 1. We define two shapes: the shape of people at risk (those who **(i)** are not vaccinated and **(ii)** have symptoms or are close to someone at risk) and the shape of people who can go to office (those who are not at risk), as formalized below:

$$atRisk \leftarrow \neg \geq_1 vaccinated.\top \wedge (\geq_1 hasSymptoms.\top \vee \exists closeTo.atRisk)$$
$$canWork \leftarrow \neg atRisk$$

For this set of shape definitions, the unique stable model equals the well-founded model and states that *d*, *e*, and *f* are at risk, while *a*, *b*, and *c* can work. In the Kripke-Kleene model, *d*, *e*, and *f* are again at risk, *c* is not at risk (and hence can work), but for *a* and *b* it is unknown whether they are at risk. There are two supported models: the stable model and one in which everyone except for *c* is at risk. ▲

5 Comparison with Existing Semantics

Corman et al. [12] already defined a supported semantics and Andreşel et al. [1] a stable semantics for SHACL. For clarity, we refer to the existing semantics as CRS-supported and ACORSS-stable semantics, and to the semantics induced by AFT, e.g., as AFT-stable. Both of them focus on brave validation, but Andreşel et al. also mention the possibility of cautious validation. The main results on correspondence between the semantics are summarized here.

In fact, Corman et al. [12, Definition 5] already defined the three-valued immediate consequence operator Ψ_{Def} (there denoted $\mathbf{T}$). While the focus of that work was on supported semantics, we now showed that in fact, by defining the approximator Ψ_{Def}, they had everything at hand to define the full. Since Corman et al. also characterized supported models as fixpoints of Ψ_{Def} (in their Definition 17), our semantics and theirs coincide.

Theorem 5.1. *I CRS-validates* (Def, T) *if and only if I bravely Sup-validates* (Def, T).

The situation is somewhat different for stable semantics, which Andreşel et al. [1] defined in terms of level mappings.

Definition 5.2 ([1]). Let I' be an interpretation. A *level assignment* for I' is a function *level* that maps tuples in $\{(\phi, a) \mid \llbracket \phi \rrbracket^{I'}(a) = \mathbf{t}\}$ to integers and satisfies **(i)** $level(\phi_1 \wedge \phi_2, a) = \max\{level(\phi_1, a), level(\phi_2, a)\}$, **(ii)** $level(\phi_1 \vee \phi_2, a) = \min\{level(\phi_i, a) \mid i \in \{1, 2\} \wedge \llbracket \phi_i \rrbracket^{I'}(a) = \mathbf{t}\}$, **(iii)** $level(\geq_n E.\phi)$ is the smallest $k \geq 0$ for which there are n elements $b_1, \ldots b_n \in \Delta^{I'}$ such that $level(\phi, b_i) \leq k$, $(a, b_i) \in \llbracket E \rrbracket^{I'}$, and $\llbracket \phi \rrbracket^{I'}(b_i) = \mathbf{t}$, and **(iv)** $level(\forall E.\phi, a) = \max(\{level(\phi, b) \mid (a, b) \in \llbracket E \rrbracket^{I'} \wedge \llbracket \phi \rrbracket^{I'}(b) = \mathbf{t}\})$.

A supported model I' is an ARCOSS-stable model if there exists a level assignment for I' such that $level(s, a) > level(\phi_s, a)$ for each rule $s \leftarrow \phi_s$ in *Def* and each a with $\llbracket s \rrbracket^{I'}(a) = \mathbf{t}$.

For the correspondence in case of stable semantics, we recall a normal form of Andreşel et al. [1]:

Definition 5.3. *Def* is in *shape normal form* if all rules in *Def* have one of the following forms:

$$s \leftarrow \top \qquad s \leftarrow \{c\} \qquad s \leftarrow \neg s' \qquad s \leftarrow s' \wedge s'' \qquad s \leftarrow s' \vee s''$$
$$s \leftarrow \geq_n E.s' \qquad s \leftarrow \forall E.s' \qquad s \leftarrow eq(E, E') \qquad s \leftarrow disj(E, E')$$

Theorem 5.4. *If I' is an AFT-stable model of Def, then it is also an ACORSS-stable model. If Def is in shape normal form, the converse also holds.*

The difference between our stable semantics and the ACORSS-stable semantics is a *semantic* (in terms of the standard three-valued truth evaluation) versus a *syntactic* (the level mappings are defined in terms of the syntactic structure of the shapes) treatment of negation and is illustrated in the next example.

Example 5.5 (Example 4.3 continued). Suppose that in our same interpretation, we wish to define a shape that identifies possible superspreaders. To do this, we say that a person is "safe" if they are vaccinated, or in contact with at most 1 non-safe person. This can be formalized as:

$$Safe \leftarrow \exists vaccinated.\top \vee \leq_1 closeTo.\neg Safe,$$

Where $\leq_1$ is an abbreviation for $\neg \geq_2$. With the interpretation of Fig. 1, there is a single AFT-stable model in which a, b, and c are safe, but d, e, and f are not. However, there are two ACROSS-stable models: the one mentioned above, and one in which everyone is safe, including the three-clique of non-vaccinated people.   ▲

6 Conclusion

When Corman et al. [12] defined the supported model semantics for SHACL, they showed how to translate *actual SHACL expressions* (as specified by the W3C recommendation) into logical expressions in a language akin to description logics. For studying recursive SHACL expressions, they even already defined the operator Ψ_{Def}. Hence, the only work left to obtain a rich family of semantics, was observing that this operator is indeed an approximators and applying AFT. As such, we believe this paper establishes strong and formal foundations for the study of recursive SHACL. Indeed, AFT does not just dictate how the semantics are to be defined, but immediately provides guarantees such as stratification and predicate introduction results that can be instrumental when developing concrete validators for recursive SHACL: we immediately obtain results about which transformations can safely be applied to our theories.

Our results have been presented without taking a stance on the choice of semantics. It is important to realize though, that if one wants to view *Def* as an (inductive) *definition* of the shapes, it has been

argued repeatedly that the well-founded semantics correctly formalizes this [17]. Under the well-founded semantics, SHACL integrates first-order constraints (of a restricted form) with inductive definitions and aggregates, and hence can be seen as a fragment of the language FO(ID, Agg), the formal foundation of the IDP language [13]. It is a topic for future work to investigate whether this can be exploited for either extending SHACL, or for developing alternative validation mechanisms, building on IDP.

References

[1] M. Andreşel, J. Corman, M. Ortiz, J.L. Reutter, O. Savkovic & M. Simkus (2020): *Stable model semantics for recursive SHACL*. In: *Proceedings of WWW*, pp. 1570–1580, doi:10.1145/3366423.3380229.

[2] Grigoris Antoniou, Paul Groth, Frank van Harmelen & Rinke Hoekstra (2012): *A Semantic Web Primer, 3rd Edition*. MIT Press.

[3] F. Baader, I. Horrocks, C. Lutz & U. Sattler (2017): *An Introduction to Description Logic*. Cambridge University Press, doi:10.1017/9781139025355.

[4] Bart Bogaerts (2019): *Weighted Abstract Dialectical Frameworks through the Lens of Approximation Fixpoint Theory*. In: *Proceedings of AAAI*, pp. 2686–2693, doi:10.1609/aaai.v33i01.33012686.

[5] Bart Bogaerts & Luís Cruz-Filipe (2018): *Fixpoint Semantics for Active Integrity Constraints*. AIJ 255, pp. 43–70, doi:10.1016/j.artint.2017.11.003.

[6] Bart Bogaerts & Luís Cruz-Filipe (2021): *Stratification in Approximation Fixpoint Theory and Its Application to Active Integrity Constraints*. ACM Trans. Comput. Log. 22(1), pp. 6:1–6:19, doi:10.1145/3430750.

[7] Bart Bogaerts, Maxime Jakubowski & Jan Van den Bussche (2021): *SHACL: A Description Logic in Disguise*. CoRR abs/2108.06096.

[8] Bart Bogaerts, Joost Vennekens & Marc Denecker (2015): *Grounded fixpoints and their applications in knowledge representation*. AIJ 224, pp. 51–71, doi:10.1016/j.artint.2015.03.006.

[9] I. Boneva, J.E. Labro Gayo & E.G. Prud'hommeaux (2017): *Semantics and validation of shape schemas for RDF*. In: *Proceedings of ISWC*, pp. 104–120, doi:10.1007/978-3-319-68288-4_7.

[10] Gerhard Brewka, Hannes Strass, Stefan Ellmauthaler, Johannes Peter Wallner & Stefan Woltran (2013): *Abstract Dialectical Frameworks Revisited*. In: *Proceedings of IJCAI*, pp. 803–809, doi:10.5555/2540128.2540245.

[11] Angelos Charalambidis, Panos Rondogiannis & Ioanna Symeonidou (2018): *Approximation Fixpoint Theory and the Well-Founded Semantics of Higher-Order Logic Programs*. TPLP 18(3-4), pp. 421–437, doi:10.1017/S1471068418000108.

[12] J. Corman, J.L. Reutter & O. Savkovic (2018): *Semantics and validation of recursive SHACL*. In: *Proceedings of ISWC*, pp. 318–336. Extended version, technical report KRDB18-01.

[13] Broes De Cat, Bart Bogaerts, Maurice Bruynooghe, Gerda Janssens & Marc Denecker (2016): *Predicate Logic as a Modelling Language: The IDP System*. CoRR abs/1401.6312v2, doi:10.1145/3191315.3191321.

[14] Marc Denecker, Victor Marek & Mirosław Truszczyński (2000): *Approximations, Stable Operators, Well-Founded Fixpoints and Applications in Nonmonotonic Reasoning*. In: *Logic-Based Artificial Intelligence*, pp. 127–144, doi:10.1007/978-1-4615-1567-8_6.

[15] Marc Denecker, Victor Marek & Mirosław Truszczyński (2003): *Uniform semantic treatment of default and autoepistemic logics*. AIJ 143(1), pp. 79–122, doi:10.1016/S0004-3702(02)00293-X.

[16] Marc Denecker, Victor Marek & Mirosław Truszczyński (2004): *Ultimate approximation and its application in nonmonotonic knowledge representation systems*. Information and Computation 192(1), pp. 84–121, doi:10.1016/j.ic.2004.02.004.

[17] Marc Denecker & Joost Vennekens (2014): *The Well-Founded Semantics Is the Principle of Inductive Definition, Revisited*. In: *Proceedings of KR*, pp. 22–31, doi:10.5555/3031929.3031933.

[18] Melvin Fitting (2002): *Fixpoint semantics for logic programming — A survey*. Theoretical Computer Science 278(1-2), pp. 25–51, doi:10.1016/S0304-3975(00)00330-3.

[19] Fangfang Liu, Yi Bi, Md. Solimul Chowdhury, Jia-Huai You & Zhiyong Feng (2016): *Flexible Approximators for Approximating Fixpoint Theory*. In: *Proceedings of Canadian AI*, pp. 224–236, doi:10.1007/978-3-319-34111-8_28.

[20] (2012): *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax*. W3C Recommendation.

[21] Nikolay Pelov, Marc Denecker & Maurice Bruynooghe (2007): *Well-founded and Stable Semantics of Logic Programs with Aggregates*. TPLP 7(3), pp. 301–353, doi:10.1017/S1471068406002973.

[22] (2014): *RDF 1.1 Primer*. W3C Working Group Note. Available at https://www.w3.org/TR/rdf11-primer/.

[23] (2017): *Shapes Constraint Language (SHACL)*. W3C Recommendation. Available at https://www.w3.org/TR/shacl/.

[24] Hannes Strass (2013): *Approximating operators and semantics for abstract dialectical frameworks*. AIJ 205, pp. 39–70, doi:10.1016/j.artint.2013.09.004.

[25] Mirosław Truszczyński (2006): *Strong and uniform equivalence of nonmonotonic theories - an algebraic approach*. Ann. Math. Artif. Intell. 48(3-4), pp. 245–265, doi:10.1007/s10472-007-9049-2.

[26] Maarten H. van Emden & Robert A. Kowalski (1976): *The Semantics of Predicate Logic as a Programming Language*. J. ACM 23(4), pp. 733–742, doi:10.1145/321978.321991.

[27] Joost Vennekens, David Gilis & Marc Denecker (2006): *Splitting an operator: Algebraic modularity results for logics with fixpoint semantics*. ACM Trans. Comput. Log. 7(4), pp. 765–797, doi:10.1145/1182613.1189735.

[28] Joost Vennekens, Maarten Mariën, Johan Wittocx & Marc Denecker (2007): *Predicate Introduction for Logics With a Fixpoint Semantics. Parts I and II*. Fundamenta Informaticae 79(1-2), pp. 187–227, doi:10.5555/2366527.2366536.