Stratification in Approximation Fixpoint Theory and Its Application to Active Integrity Constraints

BART BOGAERTS, Vrije Universiteit Brussel (VUB), Belgium
LUÍS CRUZ-FILIPE, University of Southern Denmark, Denmark

Approximation fixpoint theory (AFT) is an algebraic study of fixpoints of lattice operators that unifies various knowledge representation formalisms. In AFT, stratification of operators has been studied, essentially resulting in a theory that specifies when certain types of fixpoints can be computed stratum per stratum. Recently, novel types of fixpoints related to *groundedness* have been introduced in AFT. In this paper, we study how those fixpoints behave under stratified operators.

One recent application domain of AFT is the field of *active integrity constraints* (AICs). We apply our extended stratification theory to AICs and find that existing notions of stratification in AICs are covered by this general algebraic definition of stratification. As a result, we obtain stratification results for a large variety of semantics for AICs.

CCS Concepts: • **Computing methodologies** → **Nonmonotonic, default reasoning and belief revision**; • **Theory of computation** → *Algebraic semantics*; • **Software and its engineering** → *Semantics*; • **Information systems** → *Integrity checking*;

ACM Reference Format:
Bart Bogaerts and Luís Cruz-Filipe. 2020. Stratification in Approximation Fixpoint Theory and Its Application to Active Integrity Constraints. *ACM Trans. Comput. Logic* 1, 1 (September 2020), 18 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Approximation fixpoint theory (AFT) is an abstract, lattice-theoretic formalism designed to study semantic principles occurring in a wide variety of knowledge representation languages. In AFT, fixpoints of lattice operators and so-called approximating operators are studied. Denecker, Marek and Truszczyński [2000] (from now on abbreviated as DMT) defined various types of fixpoints associated with such operators and called them supported, Kripke-Kleene, stable and well-founded fixpoints. For logic programming [van Emden and Kowalski 1976], they showed that Fitting's semantic operator is an approximator of the two-valued immediate consequence operator and that its four different types of fixpoints coincide exactly with the four equally named semantics of logic programs. They defined semantic operators and approximators [DMT 2003] for autoepistemic logic (AEL) [Moore 1985] and default logic (DL) [Reiter 1980]. The fixpoints of those operators correspond to existing semantics from those fields and induced some new semantics. Furthermore, they used the algebraic theory to characterize the relationship between AEL and DL [DMT 2003; 2011].

Authors' addresses: Bart Bogaerts, Vrije Universiteit Brussel (VUB), Department of Computer Science, Pleinlaan 2, 1050, Brussels, Belgium, bart.bogaerts@vub.be; Luís Cruz-Filipe, University of Southern Denmark, Department of Mathematics and Computer Science, Campusvej 55, 5230, Odense, Denmark, lcfilipe@gmail.com.

1529-3785/2020/9-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Many more application domains of AFT have been discovered, such as extensions of logic programming [Antic et al. 2013; Charalambidis et al. 2018; Pelov et al. 2007], (abstract) argumentation theory [Strass 2013; Strass and Wallner 2015], description logics [Liu et al. 2016], and active integrity constraints [Bogaerts and Cruz-Filipe 2018]. The algebraic theory has been extended with generalizations of notions such as strong equivalence [Truszczyński 2006], new types of induction processes [Bogaerts et al. 2016, 2018; Denecker and Vennekens 2007] and novel types of fixpoints such as grounded, strictly grounded and partial grounded fixpoints [Bogaerts et al. 2015a,b].

The importance of grounded fixpoints comes mainly from the fact that they provide a solid theoretic foundation for studying intuitions intuitions that have been present, either explicit or implicit, in many different domains of non-monotonic reasoning. For instance, in logic programming, they formalize the old intuitions that true atoms in "good" models of logic programs need to be grounded in the program, i.e., that there needs to be a non-self-supporting explanation why they hold. Similar intuitions showed up in other fields, phrased in different ways, e.g., that facts (or models) should not be *unfounded*, or that they should be supported by *cycle-free* arguments, or by arguments that contain no vicious circles, etc. In many cases, the occurrence of such ungrounded aspects of developed semantics has led to a significant research effort to resolve them. In several cases, great efforts were done to refine semantics which did allow ungrounded models or facts. For example, the fact that the completion semantics of logic programs [Clark 1978] allows ungrounded models, e.g., for the transitive closure program, led to the development of perfect [Przymusinski 1988], stable [Gelfond and Lifschitz 1988] and well-founded semantics [Van Gelder et al. 1991]. Also for auto-epistemic logic [Moore 1985], it was known that Moore's expansion semantics accepted ungrounded models, e.g., for the theory $\{KP \Rightarrow P\}$, which motivated several attempts to refine Moore's semantics [Halpern and Moses 1985; Konolige 1988]. Bogaerts et al. [2015a] argued that the intuitions in these fields (and others, including abstract argumentation) are captured by the formal notions of groundedness in AFT. This idea of groundedness also lived in the context of active integrity constraint. Its application to that domain has historically been the trigger for developing AFT-based semantics for AICs [Cruz-Filipe 2016].

One of the powers of AFT is that all theory is developed in an abstract fashion and is readily applicable to a wide variety of logics. This has been exploited by Vennekens et al. [2006], who studied stratification in AFT and subsequently applied the theory to logic programming, autoepistemic logic and default logic. They showed among others that if an operator and its approximator are *stratifiable*, then Kripke-Kleene, well-founded, stable and supported fixpoints can be computed stratum per stratum. When applied to logic programming, they recovered existing modularity results for logic programs under the stable semantics [Gelfond and Lifschitz 1988] from, e.g. Lifschitz and Turner [1994] and Eiter et al. [1997]. Furthermore, their theory also works for the well-founded [Van Gelder et al. 1991], Kripke-Kleene [Fitting 1985] and supported semantics [Clark 1978].

In this paper, we continue the work of Vennekens et al. [2006] on stratification. More precisely, we tackle the open research question of how the recently introduced (strictly/partial) grounded fixpoints behave under stratification. We show that for stratifiable operators, also these classes of fixpoints can be computed stratum per stratum. While these results can be considered unsurprising, they provide a necessary sanity check for these new classes of fixpoints in AFT. These algebraic results are directly applicable to all application domains of AFT. We apply our extension of the theory of Vennekens et al. [2006] to the field of active integrity constraints (AICs) [Flesca et al. 2004], a formalism to equip a database with a set of integrity constraints and a mechanism to fix the database in case one of the constraints is violated. Bogaerts and Cruz-Filipe [2018] applied AFT for the first time to AICs, showing how several existing semantics can be characterized as fixpoints of a suitably defined operator, and introducing some new semantics based on an approximator of this operator. In the current work, we show that this operator, and its approximator, are indeed stratifiable, and hence

that all semantics induced by AFT can be computed following the stratification. Practically, this means that we can repair the database iteratively, following the stratification, each time only taking a small set of AICs into account. Given the high complexity of computing repairs under various semantics, results of this kind are very valuable from a practical perspective. Furthermore, we study how exactly stratification in the algebraic sense relates to an existing notion of stratification in AICs [Cruz-Filipe 2014].

Bogaerts et al. [2015a] discuss applications of grounded fixpoints in the context of logic programming (with aggregates), autoepistemic logic, and abstract argumentation. While we apply our results only in the setting of active integrity constraints, they are directly applicable in those fields as well.

The main contributions of this paper can be summarized as follows:

- we show that a variety of recently defined classes of fixpoints behave well under stratification;
- we establish a precise correspondence between stratification of active integrity constraints and stratification in AFT;
- combining these two, we show that repairs under all AFT semantics of AICs can be computed stratum by stratum.

Summarized, in the current paper, we continue

(i) the research of Bogaerts, Vennekens, and Denecker (2015a; 2015b) by showing that (partial/strictly) grounded fixpoints behave
(ii) the research of Vennekens et al. [2006] by extending their stratification results to other types of fixpoints, and
(iii) the work of Bogaerts and Cruz-Filipe [2018] by studying properties of their semantics.

On top of that, Section 5 contains a novel type of results (Propositions 5.9 and 5.12) where we characterize precisely how stratification in the algebraic and the AIC setting relate.

The rest of this paper is structured as follows. In Section 2, we recall the basics of approximation fixpoint theory. In Section 3, we show that (strictly/partial) grounded fixpoints behave nicely with respect to stratification. Next, in Section 4, we revisit active integrity constraints. In Section 5, we apply our theory to AICs and show that a stratifiable set of AICs always generates a stratifiable operator. Section 6 concludes.

2 APPROXIMATION FIXPOINT THEORY

In this section, we recall the basics of approximation fixpoint theory [DMT 2012] and the stratification results of Vennekens et al. [2006]. Our presentation is based on the preliminaries of Bogaerts and Cruz-Filipe [2018].

2.1 Lattices, operators and approximators

A *partially ordered set (poset)* $\langle L, \leq \rangle$ is a set L equipped with a partial order $\leq$, i.e., a reflexive, antisymmetric, transitive relation. As usual, we write $x < y$ as abbreviation for $x \leq y \wedge x \neq y$. A partial order $\leq$ is *well-founded* if every non-empty subset $S \subseteq L$ has a minimal element. We call $\langle L, \leq \rangle$ a *complete lattice* if every subset S of L has a least upper bound and a greatest lower bound and denote them $\bigvee S$ and $\bigwedge S$ respectively. If $S = \{x, y\}$, we write $x \wedge y$ for $\bigwedge S$ and $x \vee y$ for $\bigvee S$. A complete lattice has a least element $\bot = \bigwedge L$ and a greatest element $\top = \bigvee L$.

A lattice L is *distributive* if $\wedge$ and $\vee$ distribute over each other, i.e., if $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$ and $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$ for all $x, y, z \in L$. A bounded lattice L is *complemented* if every element $x \in L$ has a complement: an element $\neg x \in L$ satisfying $x \wedge \neg x = \bot$ and $x \vee \neg x = \top$. A *Boolean lattice* is a distributive complemented lattice.

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 $\text{lfp}(O)$.

Bogaerts et al. [2015a] called a point $x \in L$ *grounded* for O if, for each $v \in L$ such that $O(v \wedge x) \leq v$, it holds that $x \leq v$. They called a point $x \in L$ *strictly grounded* if there does not exist a y such that $y < x$, and $O(y) \wedge x \leq y$. Intuitively, if the elements of the lattice are sets of some kind (for instance, sets of atoms), a point x is grounded for O if and only if whenever we jointly remove a subset of the elements of x, the operator O rederives at least one of the removed elements. For Boolean lattices, they showed that the notions of groundedness and strict groundedness coincide.

Given a lattice L, approximation fixpoint theory (AFT) [DMT 2000] uses the 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, and use L^c to denote the set of consistent elements. Elements $(x, x) \in L^c$ are called *exact*. We sometimes abuse notation and use the tuple (x, y) and the interval $[x, y]$ interchangeably. The *precision ordering* 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 $[u, v] \subseteq [x, y]$. If L is a complete lattice, then so is $\langle L^2, \leq_p \rangle$.

AFT studies fixpoints of lattice operators $O : L \to L$ through operators approximating O. An operator $A : L^2 \to L^2$ is an *approximator* of O if it is $\leq_p$-monotone, and has the property that $A(x, x) = (O(x), O(x))$ for all x. Approximators are internal in L^c (i.e., map L^c into L^c). As usual, we often restrict our attention to *symmetric* approximators: approximators A such that, for all x and y, $A(x, y)_1 = A(y, x)_2$. DMT [2004] showed that the consistent fixpoints of interest of a symmetric approximator (see below for the definition of these fixpoints) are uniquely determined by an approximator's restriction to L^c and hence, that for practical purposes, it often suffices to define approximators on L^c.

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 $x = \text{lfp}(A(\cdot, y)_1)$ and $y = \text{lfp}(A(x, \cdot)_2)$, where $A(\cdot, y)_1$ denotes the operator $L \to L : z \mapsto A(z, y)_1$ and analogously $A(x, \cdot)_2$ denotes the operator $L \to L : z \mapsto A(x, z)_2$.
- The *A-well-founded fixpoint* is the least precise (i.e., the $\leq_p$-minimal) 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. This is equivalent to the condition that $x = \text{lfp}(A(\cdot, x)_1)$.
- A *partial A-grounded fixpoint* is a consistent pair $(x, y) \in L^c$ such that for each $v \in L$, whenever $A(x \wedge v, y \wedge v)_2 \leq v$, also $y \leq v$.

All (partial A-)stable fixpoints are (partial A-)grounded fixpoints and the A-well-founded fixpoint is the least precise partial A-grounded fixpoint [Bogaerts et al. 2015b].

2.2 Stratification in AFT

We now recall the stratification theory that has been worked out for AFT. All definitions and results in this section originate from Vennekens et al. [2006].

Definition 2.1. Let I be a set (referred to as the *index set*) and, for each $i \in I$, let $\langle S_i, \leq_i \rangle$ be a partially ordered set. The product set $S = \otimes_{i \in I} S_i$ is the set of functions

$$\bigotimes_{i \in I} S_i = \left\{ f : I \to \bigcup_{i \in I} S_i \mid \forall i \in I : f(i) \in S_i \right\}$$

and the product order $\leq_\otimes$ on $\otimes_{i \in I} S_i$ is defined by $x \leq_\otimes y$ iff $x(i) \leq_i y(i)$ for all $i \in I$.

In the rest of this paper, we assume that the index set I is itself endowed with a partial order $\prec$. Often, but not always, we require $\prec$ to be well-founded. Given a product lattice $L = \otimes_{i\in I} L_i$, we define $L|_{\preceq i}$ as the lattice $\otimes_{j\in I|j\prec i} L_j$ ordered by the appropriate restriction of $\leq$. Additionally, if $x \in L$, we write $x|_{\preceq i}$ as a shorthand for $x|_{\{j\in I|j\preceq i\}} \in L|_{\preceq i}$. We make similar conventions for $x|_{\prec i}$ and $L_{\prec i}$.

Definition 2.2. An operator O on a product lattice L is stratifiable if for each $i \in I$ and every $x, y \in L$ such that $x|_{\preceq i} = y|_{\preceq i}$, it holds that $O(x)|_{\preceq i} = O(y)|_{\preceq i}$.

In this case, for each $i \in I$ and $u \in L|_{\prec_i}$ there is a unique operator $O_i^u : L_i \to L_i$ satisfying the property that: if $x|_{\prec i} = u$, then $(O(x))(i) = O_i^u(x(i))$, for all $x \in L$. These operators are called the components of O.

The main result of Vennekens et al. [2006] is that all AFT-style fixpoints behave well with respect to stratification. Formally, it is summarized in the following theorem.

Theorem 2.3. *Let $L = \otimes_{i\in I} L_i$ be a product lattice, $O : L \to L$ an operator on L and $A : L^2 \to L^2$ an approximator of O. Assume $\prec$ is well-founded. If A is stratifiable, so is O. Furthermore, the following then holds for each pair $(x, y) \in L^2$:*

- *(x, y) is a fixpoint of A if and only if for each $i \in I$, $(x(i), y(i))$ is a fixpoint of $A_i^{(x,y)|_{\prec i}}$,*
- *(x, y) is the A-Kripke-Kleene fixpoint if and only if for each $i \in I$, $(x(i), y(i))$ is the $A_i^{(x,y)|_{\prec i}}$-Kripke-Kleene fixpoint,*
- *(x, y) is the A-well-founded fixpoint if and only if for each $i \in I$, $(x(i), y(i))$ is the $A_i^{(x,y)|_{\prec i}}$-well-founded fixpoint, and*
- *(x, y) is an A-stable fixpoint if and only if for each $i \in I$, $(x(i), y(i))$ is an $A_i^{(x,y)|_{\prec i}}$-stable fixpoint,*

Intuitively, this means that all AFT-style fixpoints can be computed "following the stratification", i.e., first computed on lower strata and than gradually extended to fixpoints of the original operator.

3 STRATIFICATION AND GROUNDED FIXPOINTS

Now, we continue the study of Vennekens et al. [2006] on how stratification can be exploited to compute various types of fixpoints of lattice operators. More specifically, we focus on grounded, strictly grounded and partial grounded fixpoints [Bogaerts et al. 2015a,b]. For grounded fixpoints, some stratification results are known in a general setting, not limited to product lattices [Bogaerts et al. 2016, Propositions 5.9 and 5.11]. In contrast to our setting, these results impose additional restrictions (monotonicity conditions) on the form of the operators.

Our main result is the following theorem. It states that all of these types of fixpoints behave nicely with respect to stratification, that is, they can be computed stratum by stratum in a sequential manner. Computing grounded, strictly grounded and partial fixpoints is computationally expensive: in several application domains, deciding whether a given operator has a grounded fixpoint is Σ_2^P-complete [Bogaerts and Cruz-Filipe 2018; Bogaerts et al. 2015a]. Being able to write particular instances of grounded fixpoints as compositions of grounded fixpoints of "smaller" operators can therefore be beneficial from a computational perspective. All in all, we split the task of finding grounded fixpoints into many smaller tasks, and the order $\prec$ provides us with a detailed execution plan by describing the dependencies between these tasks. Preliminary experiments in this direction [Cruz-Filipe et al. 2015], using different semantics than those studied here, indicate that this may lead to significant improvements in performance.

Theorem 3.1. *Let $L = \otimes_{i\in I} L_i$ be a product lattice with $\prec$ well-founded, $O : L \to L$ an operator on L and $A : L^2 \to L^2$ an approximator of O. If A is stratifiable (and hence, so is O), then the following hold:*

(i) *a point $x \in L$ is grounded for O if and only if $x(i)$ is grounded for $O_i^{x|_{\prec i}}$ for all $i \in I$,*

(ii) *a point $x \in L$ is strictly grounded for O if and only if $x(i)$ is strictly grounded for $O_i^{x|_{\prec i}}$ for all $i \in I$, and*

(iii) *a point $(x, y) \in L^c$ is A-grounded if and only if $(x(i), y(i))$ is $A_i^{(x,y)|_{\prec i}}$-grounded for all $i \in I$.*

Proof. The first point follows directly from Lemmas 3.2 and 3.4. The second point follows directly from Lemmas 3.5 and 3.6 below. The last point follows directly from Lemmas 3.7 and 3.8 below. □

We split the proof of this theorem in various lemmas to prove some stronger intermediate results. In particular Lemmas 3.4, 3.6 and 3.8 do not require $\prec$ to be well-founded.

Throughout this section, let $L = \otimes_{i \in I} L_i$ be a product lattice, O a stratifiable operator on L and A a stratifiable approximator of O.

Lemma 3.2. *Assume $\prec$ is well-founded and $x \in L$. If $x(i)$ is grounded for $O_i^{x|_{\prec i}}$ for all $i \in I$, then x is grounded for O.*

Proof. Follows immediately from the more general Lemma 3.7 since Bogaerts et al. [2015b] showed (in their Proposition 3.2) that x is grounded for O if and only if (x, x) is A-grounded. However, in order to strengthen the underlying intuitions, we also provide a direct proof.

The proof is by contraposition. Assume that $x \in L$ is not grounded for O, i.e. that there exists some $v \in L$ such that $O(x \wedge v) \leq v$ but $x \not\leq v$.

From $x \not\leq v$, it follows that $x(i) \not\leq v(i)$ for some $i \in I$, and since I is well-founded we can choose a minimal j with this property. In particular, for all $i < j$ we have that $x(i) \leq v(i)$, whence $(x \wedge v)|_{<j} = x|_{<j}$ and therefore

$$O(x \wedge v)(j) = O_j^{x|_{<j}}((x \wedge v)(j)) = O_j^{x|_{<j}}(x(j) \wedge v(j)).$$

From $O(x \wedge v) \leq v$ we conclude that in particular

$$O(x \wedge v)(j) \leq v(j).$$

Combining this with the previous yields that

$$O_j^{x|_{<j}}(x(j) \wedge v(j)) \leq v(j)$$

while $x(j) \not\leq v(j)$, and therefore $x(j)$ is not grounded for $O_j^{x|_{<j}}$, as we needed to show. □

The condition that $\prec$ is well-founded is a necessary condition of Lemma 3.2, as the following example illustrates.

Example 3.3. Consider $I = \mathbb{Z}$ with $\prec$ the standard order of integers. For each $i \in I$ consider a lattice $L_i = \{\bot_i, \top_i\}$. Consider the following operator O on $\oplus L_i$ that maps each $x \in L$ to $O(x)$ such that

- $(O(x))(i) = \top_i$ if $x(j) = \top_j$ for each $j \prec i$,
- $(O(x))(i) = \bot_i$ otherwise.

In this case, O has two fixpoints, namely $\top_L$ and $\bot_L$. It is clear that $\top_L$ is non-minimal and hence not grounded. However, for each i, $O_i^{\top_L|_{\prec i}}$ is a constant operator mapping to $\top_i$. Hence $\top_i = \top_L(i)$ is grounded for $O_i^{\top_L|_{\prec i}}$. This shows that the condition that $\prec$ is well-founded is essential in Lemma 3.2. ▲

Intuitively, the well-foundedness condition on $\prec$ guarantees that there is some base case where we start constructing our grounded fixpoint. This kind of intuition, that good fixpoints can be built from the ground up, lies at the heart of the definition of groundedness. In Example 3.3, there is no such base case for the fixpoint $\top_L$. For the converse of Lemma 3.2, the well-foundedness condition

on $\prec$ is not required: if the global point in L is grounded, then so are all its components, regardless of properties of the order.

Lemma 3.4. *Let $x \in L$. If x is grounded for O, then $x(i)$ is grounded for $O_i^{x|_{\prec i}}$ for all $i \in I$.*

Proof. Follows immediately from the more general Lemma 3.8 since Proposition 3.2 of Bogaerts et al. [2015b] shows that x is a grounded fixpoint of O if and only if (x, x) is an A-grounded fixpoint of A. In order to strengthen intuitions, we also provide a direct proof.

Assume x is grounded for O. Thus, for each $v \in L$ with $O(x \wedge v) \leq v$, it must hold that $x \leq v$.

Now, take any i, we need to show that $x(i)$ is grounded for $O_i^{x|_{\prec i}}$. To do that, take any $v_i \in L_i$ such that $O_i^{x|_{\prec i}}(x(i) \wedge v_i) \leq v_i$; we then need to show that $x(i) \leq v_i$. Now, define $v \in L$ as $v(j) = \top_j$ for all $j \neq i$ and $v(i) = v_i$. It then holds that $O(x \wedge v) \leq v$, hence also $x \leq v$ and in particular, $x(i) \leq v(i) = v_i$. □

Lemma 3.5. *Assume $\prec$ is well-founded and $x \in L$. If $x(i)$ is strictly grounded for $O_i^{x|_{\prec i}}$ for all $i \in I$, then x is strictly grounded for O.*

Proof. We proceed by contraposition. Assume that x is not strictly grounded for O, i.e. that there exists some $y \in L$ such that $y < x$ and $O(y) \wedge x \leq y$. From $y < x$ we conclude that $y(i) \leq x(i)$ for all $i \in I$, and furthermore that this inequality is strict in at least one instance. Since $\prec$ is well-founded, we can choose a minimal j for which $y(j) < x(j)$; then $y(i) = x(i)$ for all $i \prec j$, from which we conclude that $y|_{\prec j} = x|_{\prec j}$ and thus

$$O(y)(j) = O_j^{x|_{\prec j}}(y(j)).$$

However, from $O(y) \wedge x \leq y$, it follows that, in particular, $(O(y) \wedge x)(j) \leq y(j)$. Since

$$(O(y) \wedge x)(j) = O(y)(j) \wedge x(j) = O_j^{x|_{\prec j}}(y(j)) \wedge x(j),$$

we conclude that

$$O_j^{x|_{\prec j}}(y(j)) \wedge x(j) \leq y(j),$$

and thus $x(j)$ is not strictly grounded for $O_j^{x|_{\prec j}}$, as we needed to show. □

Since the lattices in Example 3.3 are Boolean lattices, the notions of groundedness and strict groundedness coincide in that example [Bogaerts 2015]. Hence, example 3.3 also illustrates that the condition that $\prec$ is well-founded is necessary for Lemma 3.5.

Lemma 3.6. *If $x \in L$ is strictly grounded for O, then for each i, $x(i)$ is strictly grounded for $O_i^{x|_{\prec i}}$.*

Proof. Assume $x \in L$ is strictly grounded for O, i.e., that there is no $y < x$ with $O(y) \wedge x \leq y$. We need to show that each $x(i)$ is strictly grounded for $O_i^{x|_{\prec i}}$.

Take any $i \in I$ and $y_i \in L_i$ with $y_i \leq x(i)$ and $O_i^{x|_{\prec i}}(y_i) \wedge x(i) \leq y_i$. We then show that $y_i = x(i)$. Define $y \in L$ as follows: $y(i) = y_i$ and for all $j \neq i$, $y(j) = x(j)$. Then, it is clear that $y \leq x$. For each $j \neq i$, it then holds that

$$\begin{aligned}(O(y) \wedge x)(j) &= (O(y))(j) \wedge x(j) \\ &\leq x(j) = y(j)\end{aligned}$$

Furthermore, $x|_{\prec i} = y|_{\prec i}$ and hence also

$$\begin{aligned}(O(y) \wedge x)(i) &= (O(y))(i) \wedge x(i) \\ &= O_i^{x|_{\prec i}}(y(i)) \wedge x(i) \\ &\leq y_i = y(i).\end{aligned}$$

Combining these two, we find that $O(y) \wedge x \leq y$ and thus, since x is strictly grounded for O and $y \leq x$, it must hold that $y = x$. In particular $y_i = x(i)$ as we needed to show. □

Lemma 3.7. *Assume $\prec$ is well-founded and $(x, y) \in L^c$. If $(x, y)(i)$ is $A_i^{(x,y)|_{\prec i}}$-grounded for all $i \in I$, then (x, y) is A-grounded.*

Proof. The proof is again by contraposition. Assume that $(x, y) \in L^c$ is not A-grounded, i.e. that there exists some $v \in L$ such that $A(x \wedge v, y \wedge v)_2 \leq v$ but $y \not\leq v$.

From $y \not\leq v$, it follows that $y(i) \not\leq v(i)$ for some $i \in I$, and since $\prec$ is well-founded we can choose a minimal j with this property. In particular, for all $i \prec j$ we have that $x(i) \leq y(i) \leq v(i)$, whence $(x \wedge v)|_{\prec j} = x|_{\prec j}$ and $(y \wedge v)|_{\prec j} = y|_{\prec j}$ and therefore

$$\begin{aligned} A(x \wedge v, y \wedge v)(j) &= A_j^{(x,y)|_{\prec j}}((x \wedge v, y \wedge v)(j)) \\ &= A_j^{(x,y)|_{\prec j}}(x(j) \wedge v(j), y(j) \wedge v(j)). \end{aligned}$$

From $A(x \wedge v, y \wedge v)_2 \leq v$ we conclude that in particular

$$A(x \wedge v, y \wedge v)_2(j) \leq v(j).$$

Combining this with the previous yields that

$$A_j^{(x,y)|_{\prec j}}(x(j) \wedge v(j), y(j) \wedge v(j))_2 \leq v(j)$$

while $y(j) \not\leq v(j)$, and therefore $(x, y)(j)$ is not $A_j^{(x,y)|_{\prec j}}$-grounded, as we needed to show. □

Lemma 3.8. *If $(x, y) \in L^c$ is A-grounded, then $(x, y)(i)$ is a $A_i^{(x,y)|_{\prec i}}$-grounded for all $i \in I$.*

Proof. Assume $(x, y) \in L^c$ is A-grounded.

Take any $i \in I$, we need to show that $(x(i), y(i))$ is $A_i^{(x,y)|_{\prec i}}$-grounded. Therefore, take any $v_i \in L_i$ with $A_i^{(x,y)|_{\prec i}}(x(i) \wedge v_i, y(i) \wedge v_i)_2 \leq v_i$. We should show that $y(i) \leq v_i$. Define $v \in L$ as follows: $v(j) = \top_j$ for all $j \neq i$ and $v(i) = v_i$. Since $v_j = \top_j$ for all $j \neq i$, it is clear that $A(x \wedge v, y \wedge v)_2(j) \leq v(j)$ for all such j. It is also clear that for $j \prec i$, $(x, y)(j) = (x \wedge v, y \wedge v)(j)$ and thus that $(x, y)|_{\prec i} = (x \wedge v, y \wedge v)|_{\prec i}$. Hence,

$$\begin{aligned} A(x \wedge v, y \wedge v)_2(i) &= A_i^{(x,y)|_{\prec i}}(x(i) \wedge v(i), y(i) \wedge v(i))_2 \\ &\leq v(i) \end{aligned}$$

Now, for all $j \in I$, we found that $A(x \wedge v, y \wedge v)_2(j) \leq v(j)$ and thus $A(x \wedge v, y \wedge v)_2 \leq v$. Since (x, y) is A-grounded, it must be the case that $y \leq v$ and in particular $y(i) \leq v(i) = v_i$, which we needed to show. □

All in all, these lemmas provide a complete picture of how groundedness (and variants thereof) with respect to an operator relates to groundedness with respect to its components.

4 ACTIVE INTEGRITY CONSTRAINTS

In modern-day databases, it is essential to specify semantic relationships between the data that is being stored. Such relationships are typically described by means of logical formulas, called *integrity constraints*. One of the most important tasks in database maintenance is to guarantee that integrity constraints remain satisfied after changes of the database.

The problem of restoring consistency of a database relative to a set of integrity constraints is known as the problem of database repair, and has been an important topic of research for several decades [Abiteboul 1988]. This problem includes two different aspects: *finding* possible repairs and *choosing* which one to apply. Typically, there are several possible ways to repair an inconsistent

database. Possible criteria to choose the "best" one that are widely accepted include minimality of change [Winslett 1990] – change as little as possible – and the common-sense law of inertia, discussed, e.g., by Przymusinski and Turner [1997] – do not change anything unless there is a reason to.

Active integrity constraints (AICs) [Flesca et al. 2004] were introduced as a formalism that captures one of the popular approaches in real-world database systems: rule-based updates, known as event-condition-action (ECA) rules [Teniente and Olivé 1995; Widom and Ceri 1996], which specify actions to be performed when a particular trigger occurs and specific conditions hold. Although they are simple to implement, the semantics of ECA rules is complex, and in particular their interaction is hard to analyze [Caroprese et al. 2006; May and Ludäscher 2002; Paton and Díaz 1999]. This is addressed by AICs: these are logic programming-style rules whose body expresses (the negation of) an integrity constraint that must be satisfied, and whose head includes update actions that can be applied to restore consistency. The declarative semantics of AICs includes several progressively more restricted classes of repairs, which can be used as criteria to select a preferred repair [Bogaerts and Cruz-Filipe 2018; Caroprese and Truszczyński 2011]. Algorithms for computing these repairs [Cruz-Filipe et al. 2013] have been implemented as a prototype [Cruz-Filipe et al. 2015].

In the rest of this section, we recall some background on AICs. We assume a fixed set of propositional atoms At. As usual, a literal is either an atom $a \in At$ or its negation $\neg a$. Given a literal l, we write $|l|$ for its underlying atom, i.e. $|a| = a$ and $|\neg a| = a$. The literals a and $\neg a$ are called *dual literals*, and we write l^D to denote the dual of a literal l. A *database* is a set of atoms, which we identify with the propositional interpretation satisfying exactly the atoms in that set. Formally, if DB is a database, then $DB \models a$ iff $a \in DB$, and $DB \models \neg a$ iff $a \notin DB$.

Databases may be updated by means of *update actions* of the form $+a$ or $-a$, with $a \in At$. Intuitively, $+a$ denotes the action "add a to the database (if it is not there already)" and $-a$ denotes the action "remove a from the database (if it is there)". The actions $+a$ and $-a$ are *dual actions*; if α is an action, we write α^D for its dual. A set of update actions $\mathcal{U}$ is *consistent* if it does not contain any pair of dual actions, i.e. $\{+a, -a\} \not\subseteq \mathcal{U}$ for all $a \in At$. If $\mathcal{U}$ is consistent, we define the result of updating DB by $\mathcal{U}$ as

$$\mathcal{U}(DB) = (DB \cup \{a \mid +a \in \mathcal{U}\}) \setminus \{a \mid -a \in \mathcal{U}\}\,.$$

Literals and update actions are connected by means of the natural operators ua and lit: $\mathsf{ua}(a) = +a$, $\mathsf{ua}(\neg a) = -a$, $\mathsf{lit}(+a) = a$ and $\mathsf{lit}(-a) = \neg a$, for $a \in At$.

An *active integrity constraint* (AIC) is a rule r of the form[1] $l_1 \wedge \ldots \wedge l_k \supset \alpha$ where $\mathsf{lit}(\alpha) \in \{l_1^D, \ldots, l_k^D\}$. The *body* of r is $l_1 \wedge \ldots \wedge l_k$, and the *head* of r is α. We say that r is *applicable* in a database DB if $DB \models \mathsf{body}(r)$, i.e. if DB satisfies all literals in $\mathsf{body}(r)$, and that r is *satisfied* by DB otherwise. The constraint $\mathsf{lit}(\alpha) \in \{l_1^D, \ldots, l_k^D\}$ ensures that r is always satisfied in $\alpha(DB)$. The set of *non-updateable literals of* r is defined as $\mathsf{nup}(r) = \{l_1, \ldots, l_k\} \setminus \mathsf{lit}(\alpha^D)$.

Flesca et al. [2004], Caroprese and Truszczyński [2011] and Bogaerts and Cruz-Filipe [2018] have defined several different semantics for AICs. In this work, we focus on semantics based on approximation fixpoint theory.

AFT-style semantics for AICs. Let $\mathcal{U}$ and $\mathcal{V}$ be sets of update actions. We define

$$\mathcal{U} \uplus \mathcal{V} = (\mathcal{U} \cup \mathcal{V}) \setminus \{\alpha \mid \alpha, \alpha^D \in \mathcal{U} \cup \mathcal{V}\}.$$

If all actions in $\mathcal{U}$ change DB and all actions in $\mathcal{V}$ change $\mathcal{U}(DB)$, then $(\mathcal{U} \uplus \mathcal{V})(DB) = \mathcal{V}(\mathcal{U}(DB))$.

[1] In the terminology used in previous works, this is a *normal* AIC. In this work we restrict ourselves to normal AICs, following Bogaerts and Cruz-Filipe [2018], and omit the general definition.

We now assume DB to be a fixed database and denote by $\mathcal{A}$ the set of update actions that change DB, i.e.

$$\mathcal{A} = \{-a \mid a \in DB\} \cup \{+a \mid a \in At \setminus DB\}\,.$$

Since DB only contains atoms, any subset of $\mathcal{A}$ is necessarily a consistent set of update actions.

Definition 4.1 ([Cruz-Filipe 2016])**.** Let η be a set of AICs over At. The operator $\mathcal{T}_\eta^{DB} : 2^{\mathcal{A}} \to 2^{\mathcal{A}}$ is defined as follows:

$$\mathcal{T}_\eta^{DB}(\mathcal{U}) = \mathcal{U} \uplus \{\mathsf{head}(r) \mid r \in \eta \wedge \mathcal{U}(DB) \models \mathsf{body}(r)\}$$

A set of update actions $\mathcal{U} \subseteq \mathcal{A}$ is a *grounded* repair for $\langle DB, \eta\rangle$ if it is a grounded fixpoint of $\mathcal{T}_\eta^{DB}$.

Observe that $\mathcal{T}_\eta^{DB}(\mathcal{U})$ is well defined: if $\mathcal{U} \subseteq \mathcal{A}$ and $\mathcal{U}(DB) \models \mathsf{body}(r)$, then necessarily either $\mathsf{head}(r)^D \in \mathcal{U}$ and is removed in $\mathcal{T}_\eta^{DB}(\mathcal{U})$, or $\mathsf{head}(r)^D \notin \mathcal{U}$ and therefore $\mathsf{head}(r) \in \mathcal{A}$ (since $\mathsf{lit}(\mathsf{head}(r))^D \in DB$). In either case, $\mathcal{T}_\eta^{DB}(\mathcal{U}) \subseteq \mathcal{A}$.

Example 4.2. Consider the following set of AICs η

$$a \wedge \neg b \supset -a$$
$$\neg a \wedge b \supset -b$$
$$a \wedge \neg c \supset +c$$
$$b \wedge \neg c \supset +c.$$

The first two rules here express that a and b should be equivalent, and that if this is not the case, they should both become false). The last two rules express that c should be true whenever a or b is true. Consider the database $DB = \{a, b\}$. There are two repairs for $\langle DB, \eta\rangle$: $\mathcal{U}_1 = \{-a, -b\}$ and $\mathcal{U}_2 = \{+c\}$. Intuitively, since a and b are already equivalent, the first two rules should not be triggered. What should happen is that c is made true. I.e., the repair $\mathcal{U}_2$ is the intended one.

The notion of groundedness formalizes this: $\mathcal{U}_1$ is not a grounded repair, while $\mathcal{U}_2$ is. ▲

By defining an approximator for $\mathcal{T}_\eta^{DB}$, we obtain more classes of semantics for AICs.

Definition 4.3 ([Bogaerts and Cruz-Filipe 2018])**.** A *partial action set* is a tuple $\mathfrak{U} = (\mathfrak{U}_c, \mathfrak{U}_p)$ where $\mathfrak{U}_c, \mathfrak{U}_p \subseteq \mathcal{A}$.

Intuitively, a partial action set approximates a set of update actions: the actions in $\mathfrak{U}_c$ are those that are *certainly* applied, while the actions in $\mathfrak{U}_p$ are *possibly* applied.

If $\alpha \in \mathcal{A}$, the *value* $\mathfrak{U}(\alpha)$ of α in $\mathfrak{U}$ is *true* (**t**) if $\alpha \in \mathfrak{U}_c$ and $\alpha \in \mathfrak{U}_p$, *false* (**f**) if $\alpha \notin \mathfrak{U}_c$ and $\alpha \notin \mathfrak{U}_p$, *unknown* (**u**) if $\alpha \notin \mathfrak{U}_c$ and $\alpha \in \mathfrak{U}_p$, or *inconsistent* (**i**) if $\alpha \in \mathfrak{U}_c$ and $\alpha \notin \mathfrak{U}_p$. We say that $\mathfrak{U}$ is *consistent* if $\mathfrak{U}_c \subseteq \mathfrak{U}_p$, in which case $\mathfrak{U}(\alpha) \neq \mathbf{i}$ for all $\alpha \in \mathcal{A}$. The set of all consistent partial action sets is denoted P.

A (consistent) partial database is a mapping $\mathfrak{DB} : At \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$. The intended reading is: $\mathfrak{DB}(a) = \mathbf{t}$ if a is certainly in the database, $\mathfrak{DB}(a) = \mathbf{f}$ if a is certainly not in the database, and $\mathfrak{DB}(a) = \mathbf{u}$ otherwise.

For $\mathfrak{U} \in P$, we define $\mathfrak{U}(DB)$ to be the partial database such that

$$\mathfrak{U}(DB) : a \mapsto \begin{cases} DB(a) & \text{if } \mathfrak{U}(*a) = \mathbf{f} \\ DB(a)^{-1} & \text{if } \mathfrak{U}(*a) = \mathbf{t} \\ \mathbf{u} & \text{otherwise,} \end{cases}$$

where $DB(a) = \mathbf{t}$ if $a \in DB$ and $DB(a) = \mathbf{f}$ otherwise, and $*a$ represents $+a$ if $a \notin DB$, and $-^a$ otherwise.

Definition 4.4 ([Bogaerts and Cruz-Filipe 2018])**.** Given a partial database $\mathfrak{DB}$, a set of AICs η and an update action α, we define the *support* of α with respect to $\langle \mathfrak{DB}, \eta \rangle$ as

$$\mathsf{supp}_{\mathfrak{DB},\eta}(\alpha) = \max_{\leq_t} \{\mathsf{nup}(r)^{\mathfrak{DB}} \mid r \in \eta \wedge \mathsf{head}(r) = \alpha\},$$

where $\mathsf{nup}(r)^{\mathfrak{DB}}$ refers to the standard three-valued truth evaluation of the conjunction of all literals in $\mathsf{nup}(r)$ in the partial interpretation $\mathfrak{DB}$ based on Kleene's truth tables [Kleene 1938].

Intuitively, this means that the support of an action α is the highest truth value of the (non-updateable part of the) body of a rule in η with α in the head.

Definition 4.5 ([Bogaerts and Cruz-Filipe 2018])**.** Given a set of AICs η, we define an operator $\mathfrak{T}_{\langle DB,\eta\rangle} : P \longrightarrow P$, such that for each $\mathfrak{U} \in P$ and each $\alpha \in \mathcal{A}$:

- If $\mathfrak{U}(\alpha) = \mathbf{f}$, then $\mathfrak{T}_{\langle DB,\eta\rangle}(\mathfrak{U})(a) = \mathsf{supp}_{\mathfrak{U}(DB),\eta}(\alpha)$.
- If $\mathfrak{U}(\alpha) = \mathbf{t}$, then $\mathfrak{T}_{\langle DB,\eta\rangle}(\mathfrak{U})(\alpha) = \mathsf{supp}_{\mathfrak{U}(DB),\eta}(\alpha^D)^{-1}$.
- Otherwise (i.e., if $\mathfrak{U}(\alpha) = \mathbf{u}$):
  - if $\mathsf{supp}_{\mathfrak{U}(DB),\eta}(\alpha) = \mathbf{t}$ and $\mathsf{supp}_{\mathfrak{U}(DB),\eta}(\alpha^D) = \mathbf{f}$, then $\mathfrak{T}_{\langle DB,\eta\rangle}(\mathfrak{U})(\alpha) = \mathbf{t}$;
  - if $\mathsf{supp}_{\mathfrak{U}(DB),\eta}(\alpha^D) = \mathbf{t}$ and $\mathsf{supp}_{\mathfrak{U}(DB),\eta}(\alpha) = \mathbf{f}$, then $\mathfrak{T}_{\langle DB,\eta\rangle}(\mathfrak{U})(\alpha) = \mathbf{f}$;
  - otherwise, $\mathfrak{T}_{\langle DB,\eta\rangle}(\mathfrak{U})(\alpha) = \mathbf{u}$.

The operator $\mathfrak{T}_{\langle DB,\eta\rangle}$ induces a number of semantics for AICs: (partial) stable repairs, AFT-well-founded repairs, Kripke-Kleene repairs and partial grounded repairs, defined as the corresponding fixpoint of $\mathfrak{T}_{\langle DB,\eta\rangle}$ (see [Bogaerts and Cruz-Filipe 2018] for details).

Example 4.6. Consider the following set η of AICs:

$$\begin{aligned} \neg a \wedge \neg b &\supset +a \\ \neg a \wedge \neg b &\supset +b \\ a \wedge \neg c &\supset +c \\ \neg d &\supset +d \\ \neg d \wedge \neg e &\supset +e \end{aligned}$$

with $DB = \emptyset$. Intuitively, in this example, there are two good repairs: every good repair should contain $+d$ and none should contain $+e$. The first two constraints express a choice to fix the lack of an a or a b: one of the two should be added. The third rule expresses that if a is added, so should c. As such, the two acceptable repairs are $\{+a, +c, +d\}$ and $\{+b, +d\}$. These are exactly the two stable repairs.

We know that the Kripke-Kleene repair approximates these two repairs, but it does so in quite a coarse way. In this example, the Kripke-Kleene repair will assign $+d$ to be true and all other actions to be unknown. The well-founded repair provides a more fine-grained approximation of the good repairs by assigning $+d$ to be true, and $+e$ to be false (and the rest to be unknown). ▲

5 STRATIFICATION OF AICS

Cruz-Filipe [2014] showed that all the declarative semantics for AICs defined by Caroprese and Truszczyński [2011] respect stratification in the sense that whenever the rules in η_1 do not depend on literals occurring in the head of rules in η_2, a set of update actions U is a repair (in a certain class of repairs, determined by the semantics at hand) if and only if, it is composed of a repair for η_1 and one for η_2 (see Theorem 5.3 for such a result for the class of grounded repairs). While such results might not seem surprising, they are far from trivial. In fact, not all semantics of AICs satisfy a similar property. For instance so-called well-founded repairs [Cruz-Filipe et al. 2013] do

not respect stratification. In this section, we extend the results of Cruz-Filipe [2014] in two ways: first of all, we extend them to stratifications of more than two (possibly infinitely many) strata. Secondly, we extend them to cover all AFT-style semantics of AICs.

As before, let I be an index set equipped with a partial order $\prec$. In this section, we assume $\prec$ to be well-founded. For the rest of this section, let $(\mathcal{A}_i)_{i\in I}$ be a partition of $\mathcal{A}$. We say that η is *stratified*[2] if for each rule $r \in \eta$ with $\mathsf{head}(r) = \alpha$ and $l \in \mathsf{body}(r)$ such that $\{\alpha, \alpha^D\} \cap \mathcal{A}_i \neq \emptyset$ and $\{\mathsf{ua}(l), \mathsf{ua}(l)^D\} \cap \mathcal{A}_j \neq \emptyset$, it must hold that $j \preceq i$. In words, this means that for all rules $r \in \eta$, all literals in the body should come from a lower stratum than the head action. For each $i \in I$, we define

$$\eta_i = \left\{r \in \eta \mid \left\{\mathsf{head}(r), \mathsf{head}(r)^D\right\} \cap \mathcal{A}_i \neq \emptyset\right\}.$$

We will also write $\eta_{\prec i} = \bigcup_{j\prec i} \eta_j$, and similarly for $\eta_{\preceq i}$, $\eta_{\not\prec i}$ and $\eta_{\not\preceq i}$.

We now study the relationship between stratifiability of $\mathcal{T}_\eta$ over I and stratification of the set η as given above. First, we show that $\mathcal{T}_\eta$ is stratifiable if η is stratified. Furthermore, the stratification of the operator coincides exactly with stratification of AICs, in the sense that the components of $\mathcal{T}_\eta$ are precisely the operators described by Cruz-Filipe [2014].

We start by remarking on the well-known fact that the lattice $2^{\mathcal{A}}$ is isomorphic to the product lattice $\bigotimes 2^{\mathcal{A}_i}$, where the isomorphism maps a set $U \in 2^{\mathcal{A}}$ to $f \in \bigotimes 2^{\mathcal{A}_i}$ with $f(i) = U \cap \mathcal{A}_i$.

Lemma 5.1. *If η is stratified, then the operator $\mathcal{T}_\eta : 2^{\mathcal{A}} \to 2^{\mathcal{A}}$ is stratifiable.*

Proof. We need to show that, for each $i \in I$ and for every $\mathcal{U}, \mathcal{V} \in 2^{\mathcal{A}}$ such that $\mathcal{U}|_{\preceq i} = \mathcal{V}|_{\preceq i}$, we have $\mathcal{T}_\eta(\mathcal{U})|_{\preceq i} = \mathcal{T}_\eta(\mathcal{V})|_{\preceq i}$. Let $\mathcal{U}$ and $\mathcal{V}$ be sets of update actions such that $\mathcal{U}|_{\preceq i} = \mathcal{V}|_{\preceq i}$.

For every rule $r \notin \eta_{\preceq i}$, the actions $\mathsf{head}(r)$ and head^D do not occur in $\mathcal{A}_{\preceq i}$. Therefore, $\mathcal{T}_\eta(\mathcal{U})|_{\preceq i} = \mathcal{T}_{\eta_i}(\mathcal{U})|_{\preceq_i}$ and $\mathcal{T}_\eta(\mathcal{V})|_{\preceq i} = \mathcal{T}_{\eta_i}(\mathcal{V})|_{\preceq_i}$.

Furthermore, for each rule $r \in \eta_{\preceq i}$, $\mathcal{U}(DB) \models \mathsf{body}(r)$ iff $\mathcal{U}|_{\preceq i}(DB) \models \mathsf{body}(r)$, since the atoms underlying actions in $\mathcal{U}|_{\not\preceq i}$ cannot occur in $\mathsf{body}(r)$. Therefore, $\mathcal{U}(DB) \models \mathsf{body}(r)$ iff $\mathcal{V}(DB) \models \mathsf{body}(r)$. From this, we find that

$$\{\mathsf{head}(r) \mid r \in \eta_i \wedge \mathcal{U}(DB) \models \mathsf{body}(r)\} = \{\mathsf{head}(r) \mid r \in \eta_i \wedge \mathcal{V}(DB) \models \mathsf{body}(r)\}$$

and thus that

$$\mathcal{T}_\eta(\mathcal{U})|_{\preceq_i} = \mathcal{T}_{\eta_i}(\mathcal{U})|_{\preceq_i} = \mathcal{T}_{\eta_i}(\mathcal{V})|_{\preceq_i} = \mathcal{T}_\eta(\mathcal{V})|_{\preceq_i}. \qquad \square$$

Lemma 5.2. *If η is stratified, then the components of $\mathcal{T}_\eta$ are given by $(\mathcal{T}_\eta)_i^{\mathcal{U}} = \mathcal{T}_{\eta_i}^{\mathcal{U}(DB)}$.*

Proof. Given that components are unique, it suffices to show that the operator defined in the statement of the lemma satisfies the component property, i.e. that if $\mathcal{V}|_{\prec i} = \mathcal{U}$, then $\mathcal{T}_\eta^{DB}(\mathcal{V})|_i = \mathcal{T}_{\eta_i}^{\mathcal{U}(DB)}(\mathcal{V}|_i)$.

First observe that, for any $\mathcal{V}$,

$$\begin{aligned}\mathcal{T}_\eta^{DB}(\mathcal{V}) &= \mathcal{V} \uplus \{\mathsf{head}(r) \mid r \in \eta \wedge \mathcal{V}(DB) \models \mathsf{body}(r)\} \\ &= \mathcal{V} \uplus \{\mathsf{head}(r) \mid r \in \eta_{\prec i} \wedge \mathcal{V}(DB) \models \mathsf{body}(r)\} \\ &\quad \uplus \{\mathsf{head}(r) \mid r \in \eta_{\not\prec i} \wedge \mathcal{V}(DB) \models \mathsf{body}(r)\}\end{aligned}$$

since the actions in the heads of rules in $\eta_{\prec i}$ and $\eta_{\not\prec i}$ are disjoint. Furthermore, $\{\mathsf{head}(r) \mid r \in \eta_{\prec i} \wedge \mathcal{V}(DB) \models \mathsf{body}(r)\} \subseteq \mathcal{A}_{\prec i}$, while $r \in \eta_{\not\preceq i}$ can only contribute with elements in $\mathcal{A}_{\not\preceq i}$. Therefore,

$$\mathcal{T}_\eta^{DB}(\mathcal{V})|_i = \mathcal{V}|_i \uplus \{\mathsf{head}(r) \mid r \in \eta_i \wedge \mathcal{V}(DB) \models \mathsf{body}(r)\}$$

[2]The original definition of stratification of AICs given by Cruz-Filipe [2014] slightly differs from ours and is based on a precedence relation that, intuitively, captures when a certain rule can safely be applied before another. Despite this difference, the two definitions are equivalent for the case of two strata (which is the case considered in [Cruz-Filipe 2014]); we used slightly different notation since this allows us to generalize to more than two strata more uniformly.

Suppose now that $\mathcal{V}_{<i} = \mathcal{U}$. Then $\mathcal{V}(DB) = \mathcal{V}|_{\not<i}(\mathcal{U}(DB))$, since $\mathcal{U} = \mathcal{V}_{<i}$ and $\mathcal{V}_{\not<i}$ change disjoint parts of DB. We then conclude that

$$\begin{aligned}\mathcal{T}_\eta^{DB}(\mathcal{V})|_i &= \mathcal{V}_i \uplus \{\mathsf{head}(r) \mid r \in \eta_i \wedge \mathcal{V}_i(\mathcal{U}(DB)) \models \mathsf{body}(r)\} \\ &= \mathcal{T}_{\eta_i}^{\mathcal{U}(DB)}(\mathcal{V}_i)\end{aligned}$$

observing as above that $\mathcal{T}_{\eta_i}^{\mathcal{U}(DB)}$ maps $2^{\mathcal{A}_i}$ to $2^{\mathcal{A}_i}$. □

Theorem 5.3. *If η is stratified, then a set of update actions $\mathcal{U}$ is a grounded repair of $\langle DB, \eta\rangle$ if and only if, for every i, $\mathcal{U}_i$ is a grounded repair of $\langle \mathcal{U}|_{<i}(DB), \eta_i\rangle$.*

Proof. Follows immediately by combining Lemmas 5.1 and 5.2 with Theorem 3.1. □

The particular case of $I = \{1, 2\}$ with $1 \prec 2$ was proven directly, as Lemma 29 by Cruz-Filipe [2016]. We showed that it can be obtained as a special case of stratification in approximation fixpoint theory. For the other semantics of AICs induced by AFT, no stratification results exist yet. We show that similar results hold: indeed, the approximator $\mathfrak{T}_{\langle DB,\eta\rangle}$ is stratifiable, similar to $\mathcal{T}_\eta$.

Lemma 5.4. *If η is stratified, then the approximator $\mathfrak{T}_{\langle DB,\eta\rangle} : 2^{\mathcal{A}} \to 2^{\mathcal{A}}$ is stratifiable.*

Proof. Let $\mathfrak{U}$ and $\mathfrak{V}$ be consistent partial action sets such that $\mathfrak{U}|_{\leq i} = \mathfrak{V}|_{\leq i}$, and let $\alpha \in \mathcal{A}_{\leq i}$. If $r \in \eta$ is such that $\mathsf{head}(r) = \alpha$ or $\mathsf{head}(r) = \alpha^D$, then by definition $r \in \eta_{\leq i}$. Therefore, all literals in $\mathsf{body}(r)$ are changeable only by actions in $\eta_{\leq i}$, and thus the truth value of all literals in $\mathsf{body}(r)$ in $\mathfrak{U}(DB)$ and $\mathfrak{V}(DB)$ coincide. Hence $\mathsf{supp}_{\mathfrak{U}(DB),\eta_{\leq i}}(\alpha) = \mathsf{supp}_{\mathfrak{V}(DB),\eta_{\leq i}}(\alpha)$ and $\mathsf{supp}_{\mathfrak{U}(DB),\eta_{\leq i}}(\alpha^D) = \mathsf{supp}_{\mathfrak{V}(DB),\eta_{\leq i}}(\alpha^D)$, whence $\mathfrak{T}_\eta(\mathfrak{U})(\alpha) = \mathfrak{T}_\eta(\mathfrak{V})(\alpha)$. Since this holds for every $\alpha \in \mathcal{A}_{\leq i}$, it follows that $\mathfrak{T}_{\langle DB,\eta\rangle}(\mathfrak{U})|_{\leq i} = \mathfrak{T}_{\langle DB,\eta\rangle}(\mathfrak{V})|_{\leq i}$. □

As observed by Vennekens et al. [2006], the components of the approximator $\mathfrak{T}_\eta$ are guaranteed to be approximators of the components of $\mathcal{T}_\eta$. However, operators can have multiple approximators, it is not clear a priori how these approximators of the components can be obtained. Here, we prove a stronger connection, similar to Lemma 5.2: the components of $\mathfrak{T}_\eta$ are the approximators defined from the sets of AICs η_i as in Definition 4.5.

Lemma 5.5. *If η is stratified, then the components of $\mathfrak{T}_\eta$ are given by $(\mathfrak{T}_{\langle DB,\eta\rangle})_i^{\mathfrak{U}} = \mathfrak{T}_{\langle \mathfrak{U}(DB),\eta_i\rangle}$.*

Proof. Again it suffices to show that the operator defined in the statement of the lemma satisfies the component property.

Suppose that $\mathfrak{V}|_{<i} = \mathfrak{U}$, and let $\alpha \in \eta_i$. If $\alpha \in \mathsf{head}(r)$ for some $r \in \eta$, then necessarily $r \in \eta_i$. Furthermore, $\mathfrak{V}|_{\not<i}(\mathfrak{U}(DB)) = \mathfrak{V}(DB)$, and since actions in $\mathfrak{V}|_{\not\leq i}$ do not change the truth value of atoms in the body of r we conclude that $\mathsf{supp}_{\mathfrak{V}(DB),\eta}(\alpha) = \mathsf{supp}_{\mathfrak{V}|_i(\mathfrak{U}(DB)),\eta_i}(\alpha)$, and likewise for α^D, whence $((\mathfrak{T}_{\langle DB,\eta\rangle})^{\mathfrak{U}}(\mathfrak{V}))|_i = \mathfrak{T}_{\langle \mathfrak{U}(DB),\eta_i\rangle}(\mathfrak{V}|_i)$. □

By combining our results with those of Vennekens et al. [2006], we obtain that all AFT-style semantics for AICs can be computed in a stratified manner.

Theorem 5.6. *If η is stratified, the following statements hold:*

- *a partial set of update actions $(\mathcal{U}_c, \mathcal{U}_p)$ is a (partial) stable repair of $\langle DB, \eta\rangle$ if and only if $(\mathcal{U}_c \cap \mathcal{A}_i, \mathcal{U}_p \cap \mathcal{A}_i)$ is a (partial) stable repair of $\langle \mathcal{U}|_{<i}(DB), \eta_i\rangle$ for every i;*
- *a partial set of update actions $(\mathcal{U}_c, \mathcal{U}_p)$ is a AFT-well-founded repair of $\langle DB, \eta\rangle$ if and only if $(\mathcal{U}_c \cap \mathcal{A}_i, \mathcal{U}_p \cap \mathcal{A}_i)$ is a AFT-well-founded repair of $\langle \mathcal{U}|_{<i}(DB), \eta_i\rangle$ for every i;*
- *a partial set of update actions $(\mathcal{U}_c, \mathcal{U}_p)$ is a Kripke-Kleene repair of $\langle DB, \eta\rangle$ if and only if $(\mathcal{U}_c \cap \mathcal{A}_i, \mathcal{U}_p \cap \mathcal{A}_i)$ is a Kripke-Kleene repair of $\langle \mathcal{U}|_{<i}(DB), \eta_i\rangle$ for every i;*

- *a partial set of update actions $(\mathcal{U}_c, \mathcal{U}_p)$ is a (partial) grounded repair of $\langle DB, \eta\rangle$ if and only if $(\mathcal{U}_c \cap \mathcal{A}_i, \mathcal{U}_p \cap \mathcal{A}_i)$ is a (partial) grounded repair of $\langle \mathcal{U}|_{\prec i}(DB), \eta_i\rangle$ for every i.*

Theorem 5.6 proves that all the AFT-style repairs can be computed following the stratification.

To end this section, we dig a bit deeper into the relation between stratification in AICs and stratification in AFT. We already showed that whenever a set of AICs is stratified, so are the associated operator and approximator. However, one might wonder whether the converse also holds, i.e., if an operator obtained from a set of AICs is stratifiable, is the set of AICs stratifiable as well? As it turns out, this is not the case in general, as the following example illustrates.

Example 5.7. Consider the following set of AICs η.

$$p \wedge \neg q \supset +q \qquad\qquad q \wedge \neg p \supset +p$$
$$\neg p \wedge \neg q \supset +q \qquad\qquad \neg q \wedge \neg p \supset +p$$

Since all rules' bodies contain literals that use both p and q, η can only be stratified in the trivial way (with I a singleton). However, η induces the same operator as the set $\eta' = \{\neg q \supset +q, \neg p \supset +p\}$, which can be stratified with $\mathcal{A}_1 = \{+p\}$ and $\mathcal{A}_2 = \{+q\}$.

It is important to note there that we use a closed-world assumption: in each given database, p is either true or false. Under an open-world assumption, η and η' might have a different interpretation, where the rules in η should only be applied in case there is explicit knowledge about p. ▲

The problem with the set η in the previous example is that the rules on the left collectively imply that the decision to add q does not depend on the particular logical value of p, i.e., in order to see that q must be added no matter what, we need to reason by case splitting on p (if p holds, the first rule derives q; otherwise, the second). In a certain sense, one could say that these rules contain redundant information. After all, why would one enforce the constraint that q always holds by $p \wedge \neg q \supset +q$ and $\neg p \wedge \neg q \supset +q$ instead of simply stating $\neg q \supset +q$? We now formally define what we mean by redundancy and show that for sets of AICs without redundancy, stratification of AICs corresponds *exactly* to stratification in AFT.

Definition 5.8. Two databases DB and DB' are said to be *equivalent modulo* p if $DB \setminus \{p\} = DB' \setminus \{p\}$.

An atom p is *redundant* with respect to an action α in a set of AICs η if: for every pair of databases DB and DB' equivalent modulo p, there is a rule $r \in \eta$ with $\mathrm{head}(r) = \alpha$ such that $DB \models \mathrm{body}(r)$ iff there is a rule $r' \in \eta$ with $\mathrm{head}(r) = \alpha$ such that $DB' \models \mathrm{body}(r')$.

In particular, if p and $\neg p$ do not occur in the body of *any* rule whose head is α, then p is redundant with respect to α. However, this notion also captures situations as in Example 5.7, where e.g. p is redundant with respect to $+q$. For sets of AICs that do not have redundant atoms in rule bodies, the converse of Lemma 5.1 does hold.

Proposition 5.9. *Suppose that that $\mathcal{T}_\eta$ is stratifiable. If there is no rule $r \in \eta_i$ whose body contains a literal underlying an action in $\mathcal{A}_j$ with $i \prec j$ that is redundant with respect to* $\mathrm{head}(r)$*, then η is stratified.*

Proof. Assume, towards contradiction, that η is not stratified. Then, there is a rule $r \in \eta_i$ such that $\mathrm{ua}\left(\mathrm{body}(r) \cup \mathrm{body}(r)^D\right) \cap \mathcal{A}_j \neq \emptyset$. Let α be an action in this intersection and $p = |\mathrm{lit}(\alpha)|$ be the atom underlying α. By hypothesis, p is not redundant with respect to $\mathrm{head}(r)$ in η, hence there exist two databases[3] $\mathcal{U}(DB)$ and $\mathcal{V}(DB)$ equivalent modulo p such that $\mathcal{U}(DB) \models \mathrm{body}(r_{\mathcal{U}})$ for some rule $r_{\mathcal{U}}$ with $\mathrm{head}(r_{\mathcal{U}}) = \alpha$ and $\mathcal{V}(DB) \not\models \mathrm{body}(r'')$ for any r'' with $\mathrm{head}(r'') = \alpha$. By construction, $\mathcal{U}|_{\preceq i} = \mathcal{V}|_{\preceq i}$ (since $\mathcal{U}$ and $\mathcal{V}$ only differ on p), but $\mathcal{T}_\eta(\mathcal{U})|_{\preceq i} \neq \mathcal{T}_\eta(\mathcal{V})|_{\preceq i}$ (since $\alpha \in \mathcal{T}_\eta(\mathcal{U})$ but $\alpha \notin \mathcal{T}_\eta(\mathcal{V})$) contradicting the hypothesis that $\mathcal{T}_\eta$ is stratifiable. □

[3]Note that every database can be written in this form

In practice, redundancy in AICs is undesirable: it obscures the nature of the constraints expressed, introduces extra (unnecessary constraints) and possibly makes finding good repairs harder. However, on the other hand, there might be good reasons to have redundancy, for instance in case different sets of integrity constraints are maintained by different developers independently, some redundancy can occur, where even if a constraint of one of the developers is redundant, it is still in place for consistency in case the other maintainer changes their constraints.

What we showed now is that redundancy can even obscure the stratified nature of a set of AICs; it is redundancy that is responsible for the difference in stratification between AICs and AFT.

In a certain sense, one might say that stratification of the operator is *semantic* stratification, while stratification, as defined for AICs is *syntactic* stratification. Since the approximator captures more syntactic elements than the operator, one might wonder what the relation there is. Of course, since whenever $\mathfrak{T}_\eta$ is stratifiable, so is $\mathcal{T}_\eta$ [Vennekens et al. 2006], it follows immediately from Proposition 5.9 that if $\mathfrak{T}_\eta$ is stratifiable and the the hypothesis of the lemma holds, then η is stratified. However, one might still expect that more refined results exist. Indeed, for Example 5.7, the approximator $\mathfrak{T}_\eta$ is not stratified either. The following example shows that $\mathfrak{T}_\eta$ can be stratified without η being stratified.

Example 5.10. Let η_1 and η_2 be the following sets of AICs:

$$\eta_1 = \{p \wedge q \supset -q, \quad q \supset -q\},$$
$$\eta_2 = \{q \wedge p \supset -p\}$$

and let η be $\eta_1 \cup \eta_2$. In this case, η is not stratified, since there is an occurrence of p in the first rule of η_1. However, the first rule is subsumed by the second, in the sense that, whenever $-q$ can be derived from the first rule, the second rule could be used to derive it as well. Therefore, the approximator $\mathfrak{T}_\eta$ is stratifiable. ▲

We now formalize this notion of subsumption (a reformulation of the well known notion of subsumption of clauses in terms of AICs) and show that only subsumed rules can be responsible for a discrepancy in stratification in the AIC-sense and stratification of the approximator.

Definition 5.11. Let r and r' be two AICs. We say that r *subsumes* r' if $\mathsf{head}(r) = \mathsf{head}(r')$ and $\mathsf{body}(r) \subseteq \mathsf{body}(r')$.

A set of AICs η is *subsumption-free* if it contains no two rules such that one subsumes the other.

Proposition 5.12. *Assume that $\mathfrak{T}_\eta$ is stratifiable. If η is subsumption-free, then η is stratifiable.*

Proof. Without loss of generality, assume that $DB = \emptyset$. (This can be assumed using Corollary 6.10 of Bogaerts and Cruz-Filipe [2018], which states that all AFT-induced semantics of AICs are invariant under shifting [Caroprese and Truszczyński 2011; Marek and Truszczynski 1998].)

Assume towards a contradiction that $\mathfrak{T}_\eta$ is stratifiable while there is some p with $+p \in \mathcal{A}_j$ such that either p or $\neg p$ occurs in the body of a rule $r \in \eta_i$. Assume it is p that occurs in r (a symmetrical argument holds for the case where it is $\neg p$). Consider the (unique) partial set of update actions $\mathfrak{U}$ such that

- $p^{\mathfrak{U}}(DB) = \mathbf{u}$,
- $l^{\mathfrak{U}}(DB) = \mathbf{f}$ for all $l \in \mathsf{body}(r) \setminus \{p\}$, and
- $l^{\mathfrak{U}}(DB) = \mathbf{u}$ for all literals not in $\mathsf{body}(r) \cup \neg\mathsf{body}(r)$.

Also consider the (unique) partial set of update actions $\mathfrak{U}'$, equal to $\mathfrak{U}$ except for $p^{\mathfrak{U}}(DB) = \mathbf{f}$.

It is clear that $\mathfrak{U}$ and $\mathfrak{U}'$ agree on $\mathcal{A}|_{\leq i}$, hence, since $\mathfrak{T}_\eta$ is stratifiable, it must hold that

$$\mathfrak{T}_\eta(\mathfrak{U})|_{\leq i} = \mathfrak{T}_\eta(\mathfrak{U}')|_{\leq i}\,.$$

Now, since $\mathsf{nup}(r)^{\mathfrak{T}_\eta(\mathfrak{U})} = \mathbf{u}$, it holds that $\mathsf{head}(r)^{\mathfrak{T}_\eta(\mathfrak{U})} \geq_t \mathbf{u}$. Hence, there must also be a rule r' with $\mathsf{head}(r) = \mathsf{head}(r')$ and $\mathsf{nup}(r')^{\mathfrak{T}_\eta(\mathfrak{U}')} \geq_t \mathbf{u}$. Due to the definition of $\mathfrak{U}'$, $\mathsf{nup}(r')$ can then consist only of literals in $\mathsf{body}(r) \setminus \{p\}$, which means that r' subsumes r, leading to a contradiction. □

It is easy to see that adding/removing a subsumed rule to/from a set of AICs η does not change $\mathfrak{T}_\eta$. Furthermore, it is easy to check (syntactically) which rules are subsumed by others. As such, Proposition 5.12 provides with a syntactic criterion to verify if the approximator of a set of AICs is stratified: we first remove all subsumed AICs and then check for stratifiability in the AIC sense [Cruz-Filipe 2014].

To summarize, in this section we first proved our main result related to AICs, namely Theorem 5.6, which states that all AFT semantics induced by AICs behave well with respect to stratification. After that, we investigated in great detail what the relationship is between stratifiability in AFT and in active integrity constraint. Using two auxiliary notions (redundancy and subsumption) we established a precise connection.

6 CONCLUSION

In this paper, we studied how grounded fixpoints, and variants thereof, behave when the operator and approximator involved are stratified. We showed that these types of fixpoints can be computed stratum per stratum. We applied our theory and the theory of Vennekens et al. [2006] to the field of active integrity constraints and found that notions of stratification defined there are instantiations of the more general theory. In particular, if a set of AICs is stratifiable, so are the associated operator and approximator and hence, all AFT-induced semantics behave nicely with respect to this stratification. We furthermore studied under which conditions the inverse implication holds, i.e., when stratifiability of the operator/approximator implies stratifiability of the set of AICs; this study yielded, among others, a syntactic check to verify if the approximator is stratified.

As such, on the theoretical side, we progressed the understanding of grounded, strictly grounded and partial grounded fixpoints in approximation fixpoint theory. On the practical side, we showed how, for a stratified set of AICs, various types of repairs can be computed following the stratification. Given the high complexity of these computations, the fact that such results can significantly reduce the size of the sets of AICs that have to be considered at each stage, can have great influence on computation times. Furthermore, our theoretic results lay the basis for distributed computations of repairs under various semantics of active integrity constraints. Developing efficient implementations that exploit stratification. Another line of future work is to investigate whether a strong correspondence, such as found in Propositions 5.9 and 5.12 can also be established in other domains.

ACKNOWLEDGMENTS

Luís Cruz-Filipe was partially supported by the Danish Council for Independent Research, Natural Sciences under grant DFF-7014-00041.

REFERENCES

Serge Abiteboul. 1988. Updates, A New Frontier. In *ICDT'88, 2nd International Conference on Database Theory, Bruges, Belgium, August 31 - September 2, 1988, Proceedings (Lecture Notes in Computer Science)*, Marc Gyssens, Jan Paredaens, and Dirk Van Gucht (Eds.), Vol. 326. Springer, 1–18. https://doi.org/10.1007/3-540-50171-1_1

Christian Antic, Thomas Eiter, and Michael Fink. 2013. Hex Semantics via Approximation Fixpoint Theory. In *Logic Programming and Nonmonotonic Reasoning, 12th International Conference, LPNMR 2013, Corunna, Spain, September 15-19, 2013. Proceedings (LNCS)*, Pedro Cabalar and Tran Cao Son (Eds.), Vol. 8148. Springer, 102–115. https://doi.org/10.1007/978-3-642-40564-8_11

Bart Bogaerts. 2015. *Groundedness in logics with a fixpoint semantics.* Ph.D. Dissertation. Department of Computer Science, KU Leuven. https://lirias.kuleuven.be/handle/123456789/496543 Denecker, Marc (supervisor), Vennekens, Joost and Van den Bussche, Jan (cosupervisors).

Bart Bogaerts and Luís Cruz-Filipe. 2018. Fixpoint Semantics for Active Integrity Constraints. *Artif. Intell.* 255 (2018), 43–70. https://doi.org/10.1016/j.artint.2017.11.003

Bart Bogaerts, Joost Vennekens, and Marc Denecker. 2015a. Grounded fixpoints and their applications in knowledge representation. *Artif. Intell.* 224 (2015), 51–71. https://doi.org/10.1016/j.artint.2015.03.006

Bart Bogaerts, Joost Vennekens, and Marc Denecker. 2015b. Partial Grounded Fixpoints. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, Qiang Yang and Michael Wooldridge (Eds.). AAAI Press, 2784–2790. http://ijcai.org/papers15/Abstracts/IJCAI15-394.html

Bart Bogaerts, Joost Vennekens, and Marc Denecker. 2016. On Well-Founded Set-Inductions and Locally Monotone Operators. *ACM Trans. Comput. Logic* 17, 4, Article 27 (Sept. 2016), 32 pages. https://doi.org/10.1145/2963096

Bart Bogaerts, Joost Vennekens, and Marc Denecker. 2018. Safe inductions and their applications in knowledge representation. *Artificial Intelligence* 259 (2018), 167 – 185. https://doi.org/10.1016/j.artint.2018.03.008

Luciano Caroprese, Sergio Greco, Cristina Sirangelo, and Ester Zumpano. 2006. Declarative Semantics of Production Rules for Integrity Maintenance. In *ICLP (LNCS)*, Sandro Etalle and Mirosław Truszczyński (Eds.), Vol. 4079. Springer, 26–40.

Luciano Caroprese and Mirosław Truszczyński. 2011. Active integrity constraints and revision programming. *TPLP* 11, 6 (2011), 905–952. https://doi.org/10.1017/S1471068410000475

Angelos Charalambidis, Panos Rondogiannis, and Ioanna Symeonidou. 2018. Approximation Fixpoint Theory and the Well-Founded Semantics of Higher-Order Logic Programs. *CoRR* abs/1804.08335 (2018). arXiv:1804.08335 http://arxiv.org/abs/1804.08335 to appear in TPLP.

Keith L. Clark. 1978. Negation as Failure. In *Logic and Data Bases*. Plenum Press, 293–322.

Luís Cruz-Filipe. 2014. Optimizing Computation of Repairs from Active Integrity Constraints. In *Foundations of Information and Knowledge Systems - 8th International Symposium, FoIKS 2014, Bordeaux, France, March 3-7, 2014. Proceedings (Lecture Notes in Computer Science)*, Christoph Beierle and Carlo Meghini (Eds.), Vol. 8367. Springer, 361–380. https://doi.org/10.1007/978-3-319-04939-7_18

Luís Cruz-Filipe. 2016. Grounded Fixpoints and Active Integrity Constraints. In *Technical Communications of the 32nd International Conference on Logic Programming, ICLP 2016 TCs, October 16-21, 2016, New York City, USA (OASIcs)*, Manuel Carro, Andy King, Marina De Vos, and Neda Saeedloei (Eds.), Vol. 52. Schloss Dagstuhl, 11:1–11:14. https://doi.org/10.4230/OASIcs.ICLP.2016.11

Luís Cruz-Filipe, Michael Franz, Artavazd Hakhverdyan, Marta Ludovico, Isabel Nunes, and Peter Schneider-Kamp. 2015. repAIrC: A Tool for Ensuring Data Consistency. In *KMIS 2015 - Proceedings of the International Conference on Knowledge Management and Information Sharing, part of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2015), Volume 3, Lisbon, Portugal, November 12-14, 2015*, Ana L. N. Fred, Jan L. G. Dietz, David Aveiro, Kecheng Liu, and Joaquim Filipe (Eds.). SciTePress, 17–26. https://doi.org/10.5220/0005586400170026

Luís Cruz-Filipe, Graça Gaspar, Patrícia Engrácia, and Isabel Nunes. 2013. Computing Repairs from Active Integrity Constraints. In *Seventh International Symposium on Theoretical Aspects of Software Engineering, TASE 2013, 1-3 July 2013, Birmingham, UK*. IEEE Computer Society, 183–190. https://doi.org/10.1109/TASE.2013.32

Marc Denecker, Maurice Bruynooghe, and Joost Vennekens. 2012. Approximation fixpoint theory and the semantics of logic and answers set programs. In *Correct Reasoning*, Esra Erdem, Joohyung Lee, Yuliya Lierler, and David Pearce (Eds.). LNCS, Vol. 7265. Springer, 178–194. https://doi.org/10.1007/978-3-642-30743-0_{}13

Marc Denecker, Victor Marek, and Mirosław Truszczyński. 2000. Approximations, Stable Operators, Well-Founded Fixpoints and Applications in Nonmonotonic Reasoning. In *Logic-Based Artificial Intelligence*, Jack Minker (Ed.). The Springer International Series in Engineering and Computer Science, Vol. 597. Springer US, 127–144. https://doi.org/10.1007/978-1-4615-1567-8_6

Marc Denecker, Victor Marek, and Mirosław Truszczyński. 2003. Uniform semantic treatment of default and autoepistemic logics. *Artif. Intell.* 143, 1 (2003), 79–122. http://dx.doi.org/10.1016/S0004-3702(02)00293-X

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

Marc Denecker, Victor Marek, and Mirosław Truszczyński. 2011. Reiter's Default Logic Is a Logic of Autoepistemic Reasoning And a Good One, Too. In *Nonmonotonic Reasoning – Essays Celebrating Its 30th Anniversary*, Gerd Brewka, Victor Marek, and Mirosław Truszczyński (Eds.). College Publications, 111–144. http://arxiv.org/abs/1108.3278

Marc Denecker and Joost Vennekens. 2007. Well-Founded Semantics and the Algebraic Theory of Non-monotone Inductive Definitions. In *LPNMR (Lecture Notes in Computer Science)*, Chitta Baral, Gerhard Brewka, and John S. Schlipf (Eds.), Vol. 4483. Springer, 84–96. https://doi.org/10.1007/978-3-540-72200-7_9

Thomas Eiter, Georg Gottlob, and Heikki Mannila. 1997. Disjunctive Datalog. *ACM Trans. Database Syst.* 22, 3 (1997), 364–418. https://doi.org/10.1145/261124.261126

Melvin Fitting. 1985. A Kripke-Kleene semantics for logic programs. *The Journal of Logic Programming* 2, 4 (1985), 295 – 312. https://doi.org/10.1016/S0743-1066(85)80005-4

Sergio Flesca, Sergio Greco, and Ester Zumpano. 2004. Active integrity constraints. In *Proceedings of the 6th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, 24-26 August 2004, Verona, Italy*, Eugenio Moggi and David Scott Warren (Eds.). ACM, 98–107. https://doi.org/10.1145/1013963.1013977

Michael Gelfond and Vladimir Lifschitz. 1988. The Stable Model Semantics for Logic Programming. In *ICLP/SLP*, Robert A. Kowalski and Kenneth A. Bowen (Eds.). MIT Press, 1070–1080. http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.24.6050

Joseph Y. Halpern and Yoram Moses. 1985. Towards a Theory of Knowledge and Ignorance: Preliminary Report. In *Logics and Models of Concurrent Systems (NATO ASI Series)*, Krzysztof R. Apt (Ed.), Vol. 13. Springer Berlin Heidelberg, 459–476. https://doi.org/10.1007/978-3-642-82453-1_16

Stephen Cole Kleene. 1938. On Notation for Ordinal Numbers. *The Journal of Symbolic Logic* 3, 4 (1938), 150–155. http://www.jstor.org/stable/2267778

Kurt Konolige. 1988. On the relation between default and autoepistemic logic. *Artif. Intell.* 35, 3 (1988), 343–382. https://doi.org/10.1016/0004-3702(88)90021-5

Vladimir Lifschitz and Hudson Turner. 1994. Splitting a Logic Program. In *Logic Programming, Proceedings of the Eleventh International Conference on Logic Programming, Santa Marherita Ligure, Italy, June 13-18, 1994*, Pascal Van Hentenryck (Ed.). MIT Press, 23–37.

Fangfang Liu, Yi Bi, Md. Solimul Chowdhury, Jia-Huai You, and Zhiyong Feng. 2016. Flexible Approximators for Approximating Fixpoint Theory. In *Advances in Artificial Intelligence - 29th Canadian Conference on Artificial Intelligence, Canadian AI 2016, Victoria, BC, Canada, May 31 - June 3, 2016. Proceedings (Lecture Notes in Computer Science)*, Richard Khoury and Christopher Drummond (Eds.), Vol. 9673. Springer, 224–236. https://doi.org/10.1007/978-3-319-34111-8_28

V. Wiktor Marek and Miroslaw Truszczynski. 1998. Revision Programming. *Theor. Comput. Sci.* 190, 2 (1998), 241–277. https://doi.org/10.1016/S0304-3975(97)00092-3

Wolfgang May and Bertram Ludäscher. 2002. Understanding the global semantics of referential actions using logic rules. *ACM Trans. Database Syst.* 27, 4 (2002), 343–397.

Robert C. Moore. 1985. Semantical considerations on nonmonotonic logic. *Artif. Intell.* 25, 1 (1985), 75–94. https://doi.org/10.1016/0004-3702(85)90042-6

Norman W. Paton and Oscar Díaz. 1999. Active Database Systems. *ACM Comput. Surv.* 31, 1 (1999), 63–103.

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

Teodor C. Przymusinski. 1988. On the Declarative Semantics of Deductive Databases and Logic Programs. In *Foundations of Deductive Databases and Logic Programming.*, Jack Minker (Ed.). Morgan Kaufmann, 193–216.

Teodor C. Przymusinski and Hudson Turner. 1997. Update by Means of Inference Rules. *J. Log. Program.* 30, 2 (1997), 125–143. https://doi.org/10.1016/S0743-1066(96)00091-X

Raymond Reiter. 1980. A Logic for Default Reasoning. *Artif. Intell.* 13, 1-2 (1980), 81–132. https://doi.org/10.1016/0004-3702(80)90014-4

Hannes Strass. 2013. Approximating operators and semantics for abstract dialectical frameworks. *Artif. Intell.* 205 (2013), 39–70. https://doi.org/10.1016/j.artint.2013.09.004

Hannes Strass and Johannes Peter Wallner. 2015. Analyzing the computational complexity of abstract dialectical frameworks via approximation fixpoint theory. *Artif. Intell.* 226 (2015), 34–74. https://doi.org/10.1016/j.artint.2015.05.003

Ernest Teniente and Antoni Olivé. 1995. Updating Knowledge Bases While Maintaining Their Consistency. *VLDB J.* 4, 2 (1995), 193–241. http://www.vldb.org/journal/VLDBJ4/P193.pdf

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

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

Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf. 1991. The Well-Founded Semantics for General Logic Programs. *J. ACM* 38, 3 (1991), 620–650. https://doi.org/10.1145/116825.116838

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

Jennifer Widom and Stefano Ceri (Eds.). 1996. *Active Database Systems: Triggers and Rules For Advanced Database Processing.* Morgan Kaufmann.

Marianne Winslett. 1990. *Updating logical databases.* Cambridge tracts in theoretical computer science, Vol. 9. Cambridge University Press.