Grounded Fixpoints and Active Integrity Constraints*

Luís Cruz-Filipe

Dept. Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark
lcfilipe@gmail.com

Abstract

The formalism of active integrity constraints was introduced as a way to specify particular classes of integrity constraints over relational databases together with preferences on how to repair existing inconsistencies. The rule-based syntax of such integrity constraints also provides algorithms for finding such repairs that achieve the best asymptotic complexity.

However, the different semantics that have been proposed for these integrity constraints all exhibit some counter-intuitive examples. In this work, we look at active integrity constraints using ideas from algebraic fixpoint theory. We show how database repairs can be modeled as fixpoints of particular operators on databases, and study how the notion of grounded fixpoint induces a corresponding notion of grounded database repair that captures several natural intuitions, and in particular avoids the problems of previous alternative semantics.

In order to study grounded repairs in their full generality, we need to generalize the notion of grounded fixpoint to non-deterministic operators. We propose such a definition and illustrate its plausibility in the database context.

1998 ACM Subject Classification H.2.7 Database Administration, D.1.6 Logic Programming

Keywords and phrases grounded fixpoints, active integrity constraints

Digital Object Identifier 10.4230/OASIcs.ICLP.2016.11

1 Introduction

The classical definition of model of a logic theory requires models to be deductively closed. An alternative phrasing of this fact is saying that models are fixpoints of some entailment operator, and indeed the semantics of many modern logic frameworks can be described as (minimal) fixpoints of particular operators – in particular, those of logic programs, default logics, or knowledge representation formalisms based on argumentation.

Several of these formalisms focus on models that can be constructed "from the ground up" (such as the minimal model of a positive logic program). Grounded fixpoints of lattice operators, studied in [5], were proposed with the intent of capturing this notion in the formal setting of algebraic fixpoint theory, and were shown to abstract from many useful types of fixpoints in logic programming and knowledge representation.

In this work, we are interested in applying this intuition within the context of databases with integrity constraints – formulas that describe logical relations between data in a database, which should hold at all times. We focus on the particular formalism of active integrity constraints (AICs), which not only specify an integrity constraint, but also give indications on how inconsistent databases can be repaired. Although not all integrity constraints can be

* Supported by the Danish Council for Independent Research, Natural Sciences, grant DFF-1323-00247.

Technical Communications of the 32nd International Conference on Logic Programming (ICLP 2016).
Editors: Manuel Carro, Andy King, Neda Saeedloei, and Marina De Vos; Article No. 11; pp. 11:1–11:14

Open Access Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

expressed in this formalism, AICs capture the class of integrity constraints that can be written in denial clausal form, which includes many examples that are important in practice [14]. Using AICs, one can distinguish between different types of repairs that embody typical desirable properties – minimality of change [12, 24], the common sense law of inertia [20], or non-circular justification for repair actions [7]. These intuitions capture many aspects of the idea of "building a model from the ground up", present in grounded fixpoints. However, the semantics of both founded [6] and justified repairs [7] exhibit counter-intuitive behaviors, which led to the proposal of well-founded repairs [9]. These in turn are not modular with respect to stratification of repairs [8], rendering their computation problematic.

In this paper we show that repairs of inconsistent databases can be characterized as fixpoints of a particular operator, with minimality of change corresponding to being a minimal fixpoint, and that both founded and well-founded repairs can be described as fixpoints with additional properties. We then study grounded fixpoints of this operator, and show that they include all founded and well-founded repair, but not all justified repairs. In particular, grounded fixpoints avoid the circularity issues found in founded repairs, while including some intuitive non-justified repairs.

To study at AICs in their full generality, we need to consider non-deterministic operators. While there is currently no notion of grounded fixpoint of a non-deterministic operator, we show that we can define this concept in the context of AICs in a manner that naturally generalizes the deterministic definition. We then show how this in turn yields a plausible definition of grounded fixpoints of non-deterministic operators within the general framework of algebraic fixpoint theory.

Related work. Database consistency has long since been recognized as an important problem in knowledge management. Especially in relational databases, integrity constraints have been used for decades to formalize relationships between data in the database that are dictated by its semantics [2, 4].

Whenever an integrity constraint is violated, it is necessary to change the database in order to regain consistency. This process of bringing the database back to consistency is known as *database repair*, and the problem of database repair is to determine whether such a transformation is possible. Typically, there are several possible ways of repairing an inconsistent database, and several criteria have been proposed to evaluate them. *Minimality of change* [12, 24] demands that the database be changed as little as possible, while the *common-sense law of inertia* [20] states that every change should have an underlying reason. While these criteria narrow down the possible database repairs, human interaction is ultimately required to choose the "best" possible repair [22].

Database management systems typically implement integrity constraints as a variant of event-condition-action rules (ECAs, [22, 23]), for which rule processing algorithms have been proposed and a procedural semantics has been defined. However, their lack of declarative semantics makes it difficult to understand the behavior of multiple ECAs acting together and to evaluate rule-processing algorithms in a principled way. Active integrity constraints (AICs) [14] are inspired by the same principle, encoding an integrity constraint together with preferred update actions to repair it. The update actions are limited to addition and removal of tuples from the database, as this suffices to implement the three main operations identified in the seminal work of Abiteboul [1]. AICs follow the tradition of expressing database dependencies through logic programming, which is common namely in the setting of deductive databases [17, 19, 20].

The declarative semantics for AICs [6, 7] is based on the concept of founded and justified repairs, motivated by different interpretations of the common-sense law of inertia, and the

operational semantics for AICs [9] allows their direct computation by means of intuitive tree algorithms, which have been implemented over SQL databases [10]. However, neither founded nor justified repairs are completely satisfactory, as counter-intuitive examples have been produced exhibiting limitations of both types of repairs. Similar flaws have been exposed for the alternative notion of well-founded repairs [9].

Deciding whether a database can be repaired is typically a computationally hard problem. In the framework of AICs, the complexity of this problem depends on the type of repairs allowed, varying between NP-complete and Σ_p^2. Because of this intrinsic complexity, techniques to split a problem in several smaller ones are important in practice. A first step in this direction was taken in [18], but that work explicitly forbids cyclic dependencies. A more general study, in the context of AICs, was undertaken in [8], which studies conditions under which a set of constraints can be split into smaller sets, whose repairs may then be computed separately.

In the more general setting of knowledge bases with more powerful reasoning abilities, the problem of computing repairs is much more involved than in databases, as it amounts to solving an abduction problem [15]. In those frameworks, AICs can help greatly with finding repairs, and we are currently investigating how this formalism can be applied outside the database world [11].

The operational semantics for AICs proposed in [9] was inspired by Antoniou's survey on semantics of default logic [3]. The realization that Reiter's original semantics for default logic [21] defines extensions by means of what is essentially a fixpoint definition naturally leads to the question of whether we can characterize repairs of inconsistent databases in a similar way. Indeed, some connections between the semantics for AICs and logic programming have been discussed in [7], and fixpoints play a crucial role in defining several semantics for logic programs [13]. These include the standard construction of minimal models of positive logic programs and the notion of answer sets (via the Gelfond–Lifschitz transform). Fixpoints also abound in other domains of logic; many of these occurrences of fixpoints are summarized in [5], and showing that several of them can be seen as instances of the same abstract notion constitutes one of those authors' motivation for studying grounded fixpoints.

2 Preliminaries

In this section we review the concepts and results that are directly relevant for the remainder of the presentation: grounded fixpoints of lattice operators [5], the formalism of active integrity constraints [14], founded [6], justified [7] and well-founded [9] (weak) repairs, and parallelization results for these.

Grounded fixpoints. A partial order is a binary relation that is reflexive, antisymmetric and transitive. A set L equipped with a partial order $\leq$ is called a *poset* (for partially ordered set), and it is customary to write $x < y$ if $x, y \in L$ are such that $x \leq y$ and $x \neq y$. Given $S \subseteq L$, an *upper bound* of S is an element x such that $s \leq x$ for all $s \in S$, and x is a *least upper bound* (lub) or *join* of S if $x \leq y$ for all upper bounds y of S, and we write $x = \bigvee S$. The notion of *(greatest) lower bound*, or *meet*, is dually defined, and written $\bigwedge S$. Meets and joins, if they exist, are necessarily unique. For binary sets, it is standard practice to write $x \wedge y$ and $x \vee y$ instead of $\bigwedge\{x, y\}$ and $\bigvee\{x, y\}$.

A *complete lattice* is a poset in which every set has a join and a meet. In particular, complete lattices have a greatest element $\top$ and a smallest element $\bot$. The *powerset lattice* of a set S is $\langle \wp(S), \subseteq \rangle$, whose elements are the subsets of S ordered by inclusion. The powerset

lattice is a complete lattice with joins given by union and meets given by intersection. Its greatest element is S, and its smallest element is $\emptyset$.

A lattice operator is a function $\mathcal{O} : L \to L$. A *fixpoint* of $\mathcal{O}$ is an element $x \in L$ for which $\mathcal{O}(x) = x$. If $x \leq y$ for all fixpoints y of $\mathcal{O}$, then x is said to be the *least* (or *minimal fixpoint* of $\mathcal{O}$. Lattice operators do not need to have fixpoints, but *monotone* operators (i.e. those for which $x \leq y$ implies $\mathcal{O}(x) \leq \mathcal{O}(y)$) always have a minimal fixpoint.

We will be interested in two particular kinds of fixpoints, introduced in [5]. We summarize the definitions and Propositions 3.3, 3.5 and 3.8 from that work.

▶ **Definition 1.** Let $\mathcal{O}$ be an operator over a lattice $\langle L, \leq \rangle$. An element $x \in L$ is:
- *grounded* for $\mathcal{O}$ if $\mathcal{O}(x \wedge v) \leq v$ implies $x \leq v$, for all $v \in L$;
- *strictly grounded* for $\mathcal{O}$ if there is no $y \in L$ such that $y < x$ and $(\mathcal{O}(y) \wedge x) \leq y$.

▶ **Lemma 2.** *Let $\mathcal{O}$ be an operator over a lattice $\langle L, \leq \rangle$.*
1. *All strictly grounded fixpoints of $\mathcal{O}$ are grounded.*
2. *If $\langle L, \leq \rangle$ is a powerset lattice, then all grounded fixpoints of $\mathcal{O}$ are strictly grounded.*
3. *All grounded fixpoints of $\mathcal{O}$ are minimal.*

We will be working mostly in a powerset lattice, so throughout this paper we will treat the notions of strictly grounded and grounded as equivalent.

Active integrity constraints (AICs). The formalism of AICs was originally introduced in [14], but later simplified in view of the results in [6]. We follow the latter's definition, with a more friendly and simplified notation.

We assume a fixed set At of *atoms* (typically, closed atomic formulas of a first-order theory); subsets of At are *databases*. A *literal* is either an atom (a) or its negation ($\neg a$), and a database DB satisfies a literal ℓ, denoted $DB \models \ell$, if: ℓ is an atom $a \in DB$, or ℓ is $\neg a$ and $a \notin DB$. An *update action* α has the form $+a$ or $-a$, where $a \in At$; $+a$ and $-a$ are *dual* actions, and we represent the dual of α by α^D. Update actions are intended to change the database: $+a$ adds a to the database (formally: it transforms DB into $DB \cup \{a\}$), while $-a$ removes it (formally: it transforms DB into $DB \setminus \{a\}$). A set of update actions $\mathcal{U}$ is *consistent* if it does not contain an action and its dual. A consistent set of update actions $\mathcal{U}$ acts on a database DB by updating DB by means of all its actions simultaneously; we denote the result of this operation by $\mathcal{U}(DB)$.

Literals and update actions are related by natural mappings lit and ua, where $\mathsf{lit}(+a) = a$, $\mathsf{lit}(-a) = \neg a$, $\mathsf{ua}(a) = +a$ and $\mathsf{ua}(\neg a) = -a$. An AIC is a rule r of the form

$$\ell_1, \ldots, \ell_n \supset \alpha_1 \mid \ldots \mid \alpha_k \tag{1}$$

where $n, k \geq 1$ and $\{\mathsf{lit}(\alpha_1^D), \ldots, \mathsf{lit}(\alpha_k^D)\} \subseteq \{\ell_1, \ldots, \ell_n\}$. The intuition behind this notation is as follows: the *body* of the rule, $\mathsf{body}(r) = \ell_1, \ldots, \ell_n$ describes an inconsistent state of the database. If $DB \models \ell_1 \wedge \ldots \wedge \ell_n$, which we write as $DB \models \mathsf{body}(r)$, then r is *applicable*, and we should fix this inconsistency by applying one of the actions in the *head* of r, $\mathsf{head}(r) = \alpha_1 \mid \ldots \mid \alpha_k$. The syntactic restriction was motivated by the observation [6] that actions that do not satisfy this condition may be removed from $\mathsf{head}(r)$ without changing the semantics of AICs, which we now describe.

Generic integrity constraints were previously written as first-order clauses with empty head (see [14]), and we can see AICs as a generalization of this concept: an integrity constraint $\ell_1 \wedge \ldots \wedge \ell_n \to \bot$ expresses no preferences regarding repairs, and thus corresponds to the (closed instances of the) AIC $\ell_1, \ldots, \ell_n \supset \mathsf{ua}(\ell_1)^D \mid \ldots \mid \mathsf{ua}(\ell_n)^D$. Our presentation essentially

treats At as a set of propositional symbols, following [7]; for the purposes of this paper, the distinction is immaterial (we can identify an AIC including variables with the set of its closed instances), but our choice makes the presentation much simpler.

A set of update actions $\mathcal{U}$ is a *weak repair* for DB and a set η of AICs (shortly, for $\langle DB, \eta\rangle$) if: (i) every action in $\mathcal{U}$ changes DB and (ii) $\mathcal{U}(DB) \not\models \mathsf{body}(r)$ for all $r \in \eta$. Furthermore, if $\mathcal{U}$ is minimal wrt set inclusion, then $\mathcal{U}$ is said to be a *repair*; repairs are also minimal among all sets satisfying only condition (ii), embodying the principle of *minimality of change* [24] explained earlier.

▶ **Definition 3.** A set of update actions $\mathcal{U}$ is *founded* wrt $\langle DB, \eta\rangle$ if, for every $\alpha \in \mathcal{U}$, there exists $r \in \mathcal{U}$ such that $\alpha \in \mathsf{head}(r)$ and $\mathcal{U}(DB) \models \mathsf{body}(r) \setminus \{\mathsf{lit}(\alpha^D)\}$. A *founded (weak) repair* is a (weak) repair that is founded.

The intuition is as follows: in a founded weak repair, every action has *support* in the form of a rule that "requires" its inclusion in $\mathcal{U}$. We will use the (equivalent) characterization of founded sets: $\mathcal{U}$ is founded iff, for every $\alpha \in \mathcal{U}$, there is a rule r such that $\alpha \in \mathsf{head}(r)$ and $(\mathcal{U} \setminus \{\alpha\})(DB) \models \mathsf{body}(r)$.

However, Caroprese *et al.* [7] discovered that there can be founded repairs exhibiting *circularity of support* (see Example 17 below), and they proposed the stricter notion of justified repair.

▶ **Definition 4.** Let $\mathcal{U}$ be a set of update actions and DB be a database.

- The *no-effect actions* wrt DB and $\mathcal{U}$ are the actions that do not affect either DB or $\mathcal{U}(DB)$: $\mathsf{neff}_{DB}(\mathcal{U}) = \{+a \mid a \in DB \cap \mathcal{U}(DB)\} \cup \{-a \mid a \notin DB \cup \mathcal{U}(DB)\}$.
- The set of *non-updateable literals* of an AIC r is $\mathsf{body}(r) \setminus \mathsf{lit}\left(\mathsf{head}(r)^D\right)$, where the functions lit and $\cdot^D$ are extended to sets in the natural way.
- $\mathcal{U}$ is *closed under* η if, for each $r \in \eta$, $\mathsf{ua}(\mathsf{nup}(r)) \subseteq \mathcal{U}$ implies $\mathsf{head}(r) \cap \mathcal{U} \neq \emptyset$.
- $\mathcal{U}$ is a *justified action set* if it is the least superset of $\mathcal{U} \cup \mathsf{neff}_{DB}(\mathcal{U})$ closed under η.
- $\mathcal{U}$ is a justified (weak) repair if $\mathcal{U}$ is a (weak) repair and $\mathcal{U} \cup \mathsf{neff}_{DB}(\mathcal{U})$ is a justified action set.

The notion of justified weak repair, however, is extremely complicated and unwieldy in practice, due to its quantification over sets of size comparable to that of DB. Furthermore, it excludes some repairs that seem quite reasonable and for which it can be argued that the circularity of support they exhibit is much weaker (see Example 20). This motivated proposing yet a third kind of weak repair: well-founded repairs, that are defined by means of an operational semantics inspired by the syntax of AICs [9].

▶ **Definition 5.** Let DB be a database and η be a set of AICs. The *well-founded repair tree* for $\langle DB, \eta\rangle$ is built as follows: its nodes are labeled by sets of update actions, with root $\emptyset$; the descendants of a node with consistent label $\mathcal{U}$ are all sets of the form $\mathcal{U} \cup \{\alpha\}$ such that there exists a rule $r \in \eta$ with $\alpha \in \mathsf{head}(r)$ and $\mathcal{U}(DB) \models \mathsf{body}(r)$. The consistent leaves of this tree are *well-founded weak repairs* for $\langle DB, \eta\rangle$.

Equivalently, a weak repair $\mathcal{U}$ for $\langle DB, \eta\rangle$ is well-founded iff there exists a sequence of actions $\alpha_1, \ldots, \alpha_n$ such that $\mathcal{U} = \{\alpha_1, \ldots, \alpha_n\}$ and, for each $1 \leq i \leq n$, there exists a rule r_i such that $\{\alpha_1, \ldots, \alpha_{i-1}\}(DB) \models \mathsf{body}(r_i)$ and $\alpha_i \in \mathsf{head}(r_i)$.

The availability of multiple actions in the heads of AICs makes the construction of repairs non-deterministic, and a normalization procedure was therefore proposed in [7]. An AIC r is *normal* if $|\mathsf{head}(r)| = 1$. If r is an AIC of the form in (1), then $\mathcal{N}(r) = \{\ell_1, \ldots, \ell_n \supset \alpha_i \mid 1 \leq i \leq k\}$, and $\mathcal{N}(\eta) = \bigcup\{\mathcal{N}(r) \mid r \in \eta\}$. It is straightforward to check that $\mathcal{U}$ is a weak

repair (respectively, repair, founded (weak) repair or well-founded (weak) repair) for $\langle DB, \eta\rangle$ iff $\mathcal{U}$ is a weak repair (resp. repair, founded (weak) repair or well-founded (weak) repair) for $\langle DB, \mathcal{N}(\eta)\rangle$; however, this equivalence does not hold for justified (weak) repairs, as shown in [7].

Parallelization. Determining whether a database satisfies a set of AICs is linear on both the size of the database and the number of constraints. However, determining whether an inconsistent database can be repaired is a much harder problem – NP-complete, if any repair is allowed, but Σ_2^P-complete, when repairs have to be founded or justified. (Here, Σ_2^P is the class of problems that can be solved in non-deterministic polynomial time, given an oracle that can solve any NP-complete problem.) This complexity only depends on the size of the set of AICs [7]. In the normalized case, several of these problems become NP-complete; even so, separating a set of AICs into smaller sets that can be processed independently has a significant practical impact [8].

There are two important splitting techniques: *parallelization*, which splits a set of AICs into smaller sets for which the database can be repaired independently (in principle, in parallel); and *stratification*, which splits a set of AICs into smaller sets, partially ordered, such that repairs can be computed incrementally using a topological sort of the order. We shortly summarize the definitions and results from [8].

▶ **Definition 6.** Let η_1 and η_2 be two sets of AICs over a common set of atoms At.

- η_1 and η_2 are *strongly independent*, $\eta_1 \perp\!\!\!\perp \eta_2$, if, for each pair of rules $r_1 \in \eta_1$ and $r_2 \in \eta_2$, $\mathsf{body}(r_1)$ and $\mathsf{body}(r_2)$ contain no common or dual literals.
- η_1 and η_2 are *independent*, $\eta_1 \perp \eta_2$, if, for each pair of rules $r_1 \in \eta_1$ and $r_2 \in \eta_2$, $\mathsf{lit}(\mathsf{head}(r_i))$ and $\mathsf{body}(r_{3-i})$ contain no common or dual literals, for $i = 1, 2$.
- η_1 *precedes* η_2, $\eta_1 \prec \eta_2$, if, for each pair of rules $r_1 \in \eta_1$ and $r_2 \in \eta_2$, $\mathsf{lit}(\mathsf{head}(r_2))$ and $\mathsf{body}(r_1)$ contain no common or dual literals, but not conversely.

From the syntactic restrictions on AICs, it follows that $\eta_1 \perp\!\!\!\perp \eta_2$ implies $\eta_1 \perp \eta_2$. Given two sets of AICs η_1 and η_2 a set of update actions $\mathcal{U}$, let $\mathcal{U}_i = \mathcal{U} \cap \{\alpha \mid \alpha \in \mathsf{head}(r), r \in \eta_i\}$.

▶ **Lemma 7.** *Let η_1 and η_2 be sets of AICs, $\eta = \eta_1 \cup \eta_2$, and $\mathcal{U}$ be a set of update actions.*

1. *If $\eta_1 \perp\!\!\!\perp \eta_2$, then $\mathcal{U}$ is a repair for $\langle DB, \eta\rangle$ iff $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$ and $\mathcal{U}_i$ is a repair for $\langle DB, \eta_i\rangle$, for $i = 1, 2$.*
2. *If $\eta_1 \perp \eta_2$, then $\mathcal{U}$ is a founded/well-founded/justified repair for $\langle DB, \eta\rangle$ iff $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$ and $\mathcal{U}_i$ is a founded/well-founded/justified repair for $\langle DB, \eta_i\rangle$, for $i = 1, 2$.*
3. *If $\eta_1 \prec \eta_2$, then $\mathcal{U}$ is a founded/justified repair for $\langle DB, \eta\rangle$ iff $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$, $\mathcal{U}_1$ is a founded/justified repair for $\langle DB, \eta_1\rangle$ and $\mathcal{U}_2$ is a founded/justified repair for $\langle \mathcal{U}_1(DB), \eta_2\rangle$.*

3 Repairs as Fixpoints

In this section we show how a set of AICs induces an operator on a suitably defined lattice. This operator is in general non-deterministic; in order to reuse the results from algebraic fixpoint theory, we restrict our attention to the case of normalized AICs, and delay the discussion of the general case to a later section.

The operator $\mathcal{T}$. Throughout this paragraph, we assume DB to be a fixed database over a set of atoms At and η to be a set of AICs over At.

The intuitive reading of an AIC r naturally suggests an operation on sets of update actions $\mathcal{U}$, defined as "if $\mathcal{U}(DB) \models \mathsf{body}(r)$ holds, then add $\mathsf{head}(r)$ to $\mathcal{U}$". However, this

definition quickly leads to inconsistent sets of update actions, which we want to avoid. We therefore propose a slight variant of this intuition.

▶ **Definition 8.** Let $\mathcal{U}$ and $\mathcal{V}$ be consistent sets of update actions over At. The set $\mathcal{U} \uplus \mathcal{V}$ is defined as $(\mathcal{U} \cup \{\alpha \in \mathcal{V} \mid \alpha^D \notin \mathcal{U}\}) \setminus \{\alpha \in \mathcal{U} \mid \alpha^D \in \mathcal{V}\}$.

This operation models sequential composition of repairs in the following sense: if every action in $\mathcal{U}$ changes DB and every action in $\mathcal{V}$ changes $\mathcal{U}(DB)$, then $(\mathcal{U} \uplus \mathcal{V})(DB) = \mathcal{V}(\mathcal{U}(DB))$. Furthermore, if $\mathcal{U}$ and $\mathcal{V}$ are both consistent, then so is $\mathcal{U} \uplus \mathcal{V}$.

We can identify subsets of At with sets of update actions by matching each atom a with the corresponding action that changes the database (i.e. $-a$ if $a \in DB$ and $+a$ otherwize). We will abuse notation and use this bijection implicitly, so that we can reason over the powerset lattice $\langle \wp(At), \subseteq \rangle$ as having sets of update actions as elements.

▶ **Definition 9.** The operator $\mathcal{T}_\eta^{DB} : \wp(At) \to \wp(\wp(At))$ is defined as follows: $\mathcal{U} \uplus \mathcal{V} \in \mathcal{T}_\eta^{DB}(\mathcal{U})$ iff $\mathcal{V}$ can be constructed by picking exactly one action from the head of each rule r such that $\mathcal{U}(DB) \models \mathsf{body}(r)$.

Each set $\mathcal{V}$ may contain less update actions than there are rules r for which $\mathcal{U}(DB) \models \mathsf{body}(r)$, as the same action may be chosen from the heads of different rules; and there may be rules r for which $|\mathsf{head}(r) \cap \mathcal{V}| > 1$. This is illustrated in the following simple example.

▶ **Example 10.** Let $DB = \{a, b\}$ and $\eta = \{a, b, \neg c \supset -a \mid -b; \quad a, b, \neg d \supset -a \mid -b\}$. Then $\mathcal{T}_\eta^{DB}(\emptyset) = \{\{-a\}, \{-b\}, \{-a, -b\}\}$: the bodies of both rules are satisfied in DB, and we can choose $-a$ from the heads of both, $-b$ from the heads of both, or $-a$ from one and $-b$ from the other.

The syntactic restrictions on AICs guarantee that all sets $\mathcal{V}$ in the above definition are consistent: if $+a, -a^D \in \mathcal{V}$, then there are rules r_1 and r_2 such that $\neg a \in \mathsf{body}(r_1)$ and $a \in \mathsf{body}(r_2)$ with $\mathcal{U}(DB) \models \mathsf{body}(r_i)$ for $i = 1, 2$, which is impossible. In the interest of legibility, we will write $\mathcal{T}$ instead of $\mathcal{T}_\eta^{DB}$ whenever DB and η are clear from the context.

The normalized case. In the case that η contains only normalized AICs, the set $\mathcal{T}(\mathcal{U})$ is a singleton, and we can see $\mathcal{T}$ as a lattice operator over $\langle \wp(At), \subseteq \rangle$. We will assume this to be the case throughout the remainder of this section, and by abuse of notation use $\mathcal{T}$ also in this situation. In the normalized case, we thus have

$$\mathcal{T}(\mathcal{U}) = \mathcal{U} \uplus \{\mathsf{head}(r) \mid \mathcal{U}(DB) \models \mathsf{body}(r)\}\,.$$

Since we can always transform η in a set of normalized AICs by the transformation $\mathcal{N}$ defined above, in most cases it actually suffices to consider this simpler scenario, which warrants its study. The exception is the case of justified repairs for non-normalized AICs, which we defer to a later section. All our results also apply to general integrity constraints by seeing them as AICs with maximal heads and applying $\mathcal{N}$ to the result.

The operator $\mathcal{T}$ characterizes the notions of weak repair, repair, founded and well-founded sets of update actions.

▶ **Lemma 11.** *U is a weak repair for $\langle DB, \eta \rangle$ iff $\mathcal{U}$ is a fixpoint of $\mathcal{T}$.*

▶ **Lemma 12.** *U is a repair for $\langle DB, \eta \rangle$ iff $\mathcal{U}$ is a minimal fixpoint of $\mathcal{T}$.*

▶ **Lemma 13.** *A consistent set of update actions $\mathcal{U}$ is founded wrt $\langle DB, \eta \rangle$ iff, for all $\alpha \in \mathcal{U}$, it is the case that $\alpha \in \mathcal{T}(\mathcal{U} \setminus \{\alpha\})$.*

▶ **Lemma 14.** *A weak repair $\mathcal{U}$ for $\langle DB, \eta\rangle$ is well-founded iff there is an ordering $\alpha_1, \ldots, \alpha_n$ of the elements of $\mathcal{U}$ such that $\alpha_i \in \mathcal{T}(\{\alpha_1, \ldots, \alpha_{i-1}\})$ for each $i = 1, \ldots, n$.*

The correspondence between justified repairs and answer sets for particular logic programs [7] shows that justified repairs can also be characterized in a related manner. However, since answer sets of a logic program are models of its Gelfond–Lifschitz transform, the corresponding characterization in terms would be as fixpoints of the corresponding operator for a similarly derived set of AICs, rather than of $\mathcal{T}$. This characteristic of justified repairs also explains the rather unexpected behavior we will see later, in § 5.

Grounded fixpoints of $\mathcal{T}$. Founded, well-founded and justified repairs were all introduced with the purpose of characterizing a class of repairs whose actions are supported (there is a reason for having them in the set), and that support is not circular; in particular, these repairs should be constructible "from the ground up", which was the motivation for defining well-founded repairs. However, all notions exhibit unsatisfactory examples: there exist founded repairs with circular support [7] and repairs with no circular support that are not justified [9]; well-founded repairs, on the other hand, are not stratifiable [8], which impacts their computation in practice.

Following the intuition in [5] that grounded fixpoints capture the idea of building fixpoints "from the ground up", we propose the following notion of $\mathcal{T}$.

▶ **Definition 15.** A repair $\mathcal{U}$ for $\langle DB, \eta\rangle$ is *grounded* if $\mathcal{U}$ is a grounded fixpoint of $\mathcal{T}$.

Since we are working within a powerset lattice, the notions of grounded and strictly grounded fixpoints coincide. As it turns out, the latter notion is most convenient for the proofs of our results. We thus characterize grounded repairs as repairs $\mathcal{U}$ such that: if $\mathcal{V} \subsetneq \mathcal{U}$, then $\mathcal{T}(\mathcal{V}) \cap \mathcal{U} \not\subseteq \mathcal{V}$. Equivalently: if $\mathcal{V} \subsetneq \mathcal{U}$, then $\mathcal{T}(\mathcal{V}) \cap (\mathcal{U} \setminus \mathcal{V}) \neq \emptyset$.

Since all grounded fixpoints are minimal, it makes no sense to define grounded weak repairs. The notion of grounded fixpoint therefore intrinsically embodies the principle of minimality of change, unlike other kinds of weak repairs previously defined. Furthermore, grounded repairs also embody the notion of "support" previously defined.

▶ **Lemma 16.** *Every grounded repair for $\langle DB, \eta\rangle$ is both founded and well-founded.*

However, the notion of grounded repair is strictly stronger than both of these: the first example, from [9], also shows that some forms of circular justifications are avoided by grounded repairs.

▶ **Example 17.** Let $DB = \{a, b\}$ and $\eta = \{a, \neg b \supset -a; \quad a, \neg c \supset +c; \quad \neg a, b \supset -b; \quad b, \neg c \supset +c\}$. Then $\mathcal{U} = \{-a, -b\}$ is a founded repair that is not grounded: $\mathcal{V} = \emptyset$ satisfies $\mathcal{T}(\mathcal{V}) \cap \mathcal{U} = \{+c\} \cap \mathcal{U} = \emptyset \subseteq \mathcal{V}$. The more natural repair $\mathcal{U}' = \{+c\}$ is both founded and grounded.

▶ **Example 18.** Let $DB = \emptyset$ and $\eta = \{a, \neg b, \neg c \supset +c; \quad \neg a, \neg b \supset +b; \quad \neg a \supset +a\}$. There are two well-founded repairs for $\langle DB, \eta\rangle$: $\mathcal{U}_1 = \{+a, +c\}$ (obtained by applying the last rule and then the first) and $\mathcal{U}_2 = \{+b, +a\}$ (obtained by applying the second rule and then the last). However, $\mathcal{U}_2$ is not founded ($+b$ is not founded), so it cannot be grounded: indeed, $\mathcal{V} = \{+a\}$ is a strict subset of $\mathcal{U}_2$, and $\mathcal{T}(\mathcal{V}) \cap \mathcal{U} = \{+a, +b\} \cap \mathcal{U} = \emptyset \subseteq \mathcal{V}$.

Also in this last example the grounded repair ($\mathcal{U}_1$) is somewhat more natural.

We now investigate the relation to justified repairs, and find that all justified repairs are grounded, but not conversely – confirming our earlier claim that the notion of justified repair is too strong.

▶ **Lemma 19.** *Every justified repair for $\langle DB, \eta\rangle$ is grounded.*

This result is not very surprising: justified weak repairs are answer sets of a particular logic program (Theorem 6 in [7]), and in turn answer sets of logic programs are grounded fixpoints of the consequence operator (see remark at the top of § 5 in [5]). However, the translation defined in [7] is from logic programs to databases with AICs (rather than the other way around), so Lemma 19 is *not* a direct consequence of those results.

The notion of justified repair is also stricter than that of grounded repair, as the following example from [7] shows.

▶ **Example 20.** Let $DB = \{a, b\}$ and $\eta = \{a, b \supset -a; \quad a, \neg b \supset -a; \quad \neg a, b \supset -b\}$. Then $\mathcal{U} = \{-a, -b\}$ is not justified (see [7]), but it is grounded: if $-a \in \mathcal{V} \subsetneq \mathcal{U}$, then $\mathcal{T}(\mathcal{V}) \cap \mathcal{U}$ contains $-b \in \mathcal{U} \setminus \mathcal{V}$, else $\mathcal{T}(\mathcal{V}) \cap \mathcal{U}$ contains $-a \in \mathcal{U} \setminus \mathcal{V}$.

This example was used in [9] to point out that justified repairs sometimes eliminate "natural" repairs; in this case, the first rule clearly motivates the action $-a$, and the last rule then requires $-b$. This is in contrast to Example 17, where there was no clear reason to include either $-a$ or $-b$ in a repair. So grounded repairs avoid this type of unreasonable circularities, without being as restrictive as justified repairs.

We thus have that grounded repairs are always founded and well-founded; the next example shows that they do not correspond to the intersection of those classes.

▶ **Example 21.** Assume that $DB = \emptyset$ and η contains the following integrity constraints.

$$\neg a, \neg b \supset +a \quad a, \neg b \supset +b \quad \neg a, b \supset -b \quad a, b, \neg c \supset +c \quad a, \neg b, c \supset +b \quad \neg a, b, c \supset +a$$

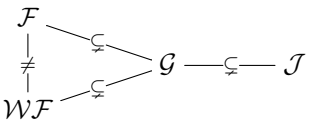
Then $\mathcal{U} = \{+a, +b, +c\}$ is a repair for $\langle DB, \eta\rangle$: the first three constraints require $+a$ and $+b$ to be included in any repair for $\langle DB, \eta\rangle$, and the last three state that no 2-element subset of $\mathcal{U}$ is a repair. Furthermore, $\mathcal{U}$ is founded (the three last rules ensure that) and well-founded (starting with $\mathcal{U}$, the rules force us to add $+a$, $+b$ and $+c$, in that order).

However, $\mathcal{U}$ is not strictly grounded for $\mathcal{T}$: if $\mathcal{V} = \{+b\}$, then $\mathcal{V} \subsetneq \mathcal{U}$, but $\mathcal{T}(\mathcal{V}) \cap \mathcal{U} = \emptyset \cap \mathcal{U} = \emptyset \subseteq \mathcal{V}$.

In this situation, $\mathcal{U}$ actually seems reasonable; however, observe that the support for its actions *is* circular: it is the three rules in the second row that make $\mathcal{U}$ founded, and none of them is applicable to DB. Also, note that $\mathcal{V}(DB)$ is a database for which the given set η behaves very awkwardly: the only applicable AIC tells us to remove b, but the only possible repair is actually $\{+a, +c\}$.

We do not feel that this example weakens the case for studying ground repairs, though: the consensual approach to different notions of repair is that they express *preferences*. In this case, where $\langle DB, \eta\rangle$ admits no grounded repair, it is sensible to allow a repair in a larger class – and a repair that is both founded and well-founded is a good candidate. The discussion in § 8 of [7] already proposes such a "methodology": choose a repair from the most restrictive category (justified, founded, or any). We advocate a similar approach, but dropping justified repairs in favor of grounded repairs, and preferring well-founded to founded repairs.

The relations between the different classes of repairs are summarized in the picture below.



We conclude this section with a note on complexity.

▶ **Theorem 22.** *The problem of deciding whether there exist grounded repairs for* $\langle DB, \eta\rangle$ *is* Σ_2^P*-complete.*

This result still holds if we allow a truly first-order syntax for AICs, where the atoms can include variables that are implictly universally quantified.

4 Parallelism

Lemma 7 shows that splitting a set of AICs into smaller ones transforms the problem of deciding whether an inconsistent database can be repaired (and computing founded or justified repairs) into smaller ones, with important practical consequences. The goal of this section is to show that grounded repairs enjoy similar properties. This is even more relevant, as deciding whether grounded repairs exist is presumably[1] more complex than for the other cases, in view of Theorem 22. For parallelization, we will go one step further, and propose a lattice-theoretical concept of splitting an operator into "independent" operators in such a way that strictly grounded fixpoints can be computed in parallel.

We make some notational conventions for the remainder of this section. We will assume as before a fixed database DB and set of AICs η over the same set of atoms At. Furthermore, we will take η_1 and η_2 to be disjoint sets with $\eta = \eta_1 \cup \eta_2$, and write $\mathcal{T}_i$ for $\mathcal{T}_{\eta_i}^{DB}$. Also, we write $\hat{\imath}$ for $3 - i$ (so $\hat{\imath} = 1$ if $i = 2$ and vice-versa).

Independence. We begin with a simple consequence of independence.

▶ **Lemma 23.** *If* $\eta_1 \perp \eta_2$*, then* $\mathcal{T}_1$ *and* $\mathcal{T}_2$ *commute and* $\mathcal{T} = \mathcal{T}_1 \circ \mathcal{T}_2 = \mathcal{T}_2 \circ \mathcal{T}_1$*.*

The converse is not true.

▶ **Example 24.** Let $\eta_1 = \{a, b \supset -b\}$ and $\eta_2 = \{\neg a, \neg b \supset +b\}$. Then $\eta_1 \not\perp \eta_2$, but $\mathcal{T}_1$ and $\mathcal{T}_2$ commute: if $a \in \mathcal{U}(DB)$, then $\mathcal{T}_1(\mathcal{T}_2(\mathcal{U})) = \mathcal{T}_1(\mathcal{U}) = \mathcal{T}_2(\mathcal{T}_1(\mathcal{U}))$; otherwise, $\mathcal{T}_1(\mathcal{T}_2(\mathcal{U})) = \mathcal{T}_2(\mathcal{U}) = \mathcal{T}_2(\mathcal{T}_1(\mathcal{U}))$.

▶ **Lemma 25.** *A set of update actions* $\mathcal{U}$ *is a grounded repair for* $\langle DB, \eta\rangle$ *iff* $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$ *and* $\mathcal{U}_1$ *is a grounded repair for* $\langle DB, \eta_1\rangle$ *and* $\mathcal{U}_2$ *is a grounded repair for* $\langle DB, \eta_2\rangle$*.*

These properties are actually not specific to operators induced by AICs, but can be formulated in a more general lattice-theoretic setting.

▶ **Definition 26.** Let $\langle L, \leq\rangle$ be a complete distributive lattice with complements. An operator $\mathcal{O} : L \to L$ is an (u, v)-operator, with $u \leq v \in L$, if, for every $x \in L$,

$$\mathcal{O}(x) = (\mathcal{O}(x \wedge v) \wedge u) \vee (x \wedge \bar{u})\,.$$

Intuitively, an (u, v)-operator only depends on the "v-part" of its argument, and the result only differs from the input in its "u-part". In this context, Proposition 3.5 of [5] applies, so grounded and strictly grounded fixpoints coincide; furthermore, we can extend the definition of independence to this setting and generalize Lemmas 23 and 25.

Observe that, by construction, $\mathcal{T}_\eta$ is a $(\mathcal{U}, \mathcal{V})$-operator with $\mathcal{U} = \{\mathsf{head}(r) \mid r \in \eta\}$ and $\mathcal{V} = \{\mathsf{ua}(l) \mid l \in \mathsf{body}(r), r \in \eta\}$.

[1] I.e., assuming that P $\neq$ NP.

▶ **Definition 27.** Two operators $\mathcal{O}_1, \mathcal{O}_2 : L \to L$ are *independent* if each $\mathcal{O}_i$ is an (u_i, v_i)-operator with $u_i \wedge v_{\hat{i}} = \bot$.

▶ **Lemma 28.** *If $\mathcal{O}_1$ and $\mathcal{O}_2$ are independent, then $\mathcal{O}_1$ and $\mathcal{O}_2$ commute. In this case, if $\mathcal{O}$ is their composition, then $x \in L$ is (strictly) grounded for $\mathcal{O}$ iff $x = (x \wedge v_1) \vee (x \wedge v_2)$ and $x \wedge v_i$ is (strictly) grounded for $\mathcal{O}_i$.*

This provides an algebraic counterpart to the parallelization of AICs, albeit requiring that the underlying lattice be distributive and complemented: we say that $\mathcal{O}$ is parallelizable if there exist $\mathcal{O}_1$ and $\mathcal{O}_2$ in the conditions of Lemma 28, with $\mathcal{O} = \mathcal{O}_1 \circ \mathcal{O}_2$. As in the original work [8], it is straightforward to generalize these results to finite sets of independent operators.

Stratification. We now consider the case where η_1 and η_2 are not independent, but rather stratified, and show that part 3 of Lemma 7 also applies to grounded repairs.

▶ **Lemma 29.** *Suppose that $\eta_1 \prec \eta_2$. Then $\mathcal{U}$ is a grounded repair for $\langle DB, \eta\rangle$ iff $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$, $\mathcal{U}_1$ is a grounded repair for $\langle DB, \eta_1\rangle$, and $\mathcal{U}_2$ is a grounded repair for $\langle \mathcal{U}_1(DB), \mathcal{U}_2\rangle$.*

Unlike parallelization, there is no clear generalization of these results to a more general setting: the definition of $\mathcal{T}_2$ is dependent of the particular fixpoint for $\mathcal{T}_1$, and to express this dependency we are using the sets η_1 and η_2 in an essential way.

5 General AICs and Non-deterministic Operators

We now return to the original question of defining grounded repairs for databases with arbitrary (not necessarily normal) sets of active integrity constraints. This requires generalizing the definition of (strictly) grounded element to non-deterministic lattice operators, a question that was left open in [5]. We propose possible definitions for these concepts, and show that they exhibit desirable properties in our topic of interest.

Let $\mathcal{O} : L \to L$ be a lattice operator, and define its non-deterministic counterpart $\mathcal{O}^\uparrow : L \to \wp(L)$ by $\mathcal{O}^\uparrow(x) = \{\mathcal{O}(x)\}$. A reasonable requirement is that x should be (strictly) grounded for $\mathcal{O}^\uparrow$ iff x is (strictly) grounded for $\mathcal{O}$. Furthermore, in the case of AICs we can also define a converse transformation: since every set of AICs η can be transformed into a normalized set $\mathcal{N}(\eta)$, we will also require that $\mathcal{U}$ be a grounded repair for $\mathcal{T}_\eta$ iff $\mathcal{U}$ is a grounded repair for $\mathcal{T}_{\mathcal{N}(\eta)}$.

▶ **Definition 30.** Let $\mathcal{O} : L \to \wp(L)$ be a non-deterministic operator over a complete lattice L. An element $x \in L$ is:

- *grounded* for $\mathcal{O}$ if $(\bigvee \mathcal{O}(x \wedge v)) \leq v$ implies $x \leq v$;
- *strictly grounded* for $\mathcal{O}$ if there is no $v < x$ such that $(\bigvee \mathcal{O}(v)) \wedge x \leq v$.

Clearly these definitions satisfy the first criterion stated above: given $\mathcal{O} : L \to L$, $\bigvee(\mathcal{O}^\uparrow(x)) = \mathcal{O}(x)$ for every $x \in L$. The choice of a join instead of a meet is motivated by the second criterion, which we will show is satisfied by this definition. Furthermore, all grounded elements are again strictly grounded, and the two notions coincide over powerset lattices – the proofs in [5] are trivial to adapt to this case.

As before, we assume that the database DB is fixed, and omit it from the superscript in the operators below.

▶ **Lemma 31.** *For every $\mathcal{U}$, $\mathcal{T}_{\mathcal{N}(\eta)}(\mathcal{U}) \subseteq \bigcup \mathcal{T}_\eta(\mathcal{U})$.*

Note that the set $\{\bigcup \mathsf{head}(r) \mid \mathcal{U}(DB) \models \mathsf{body}(r), r \in \eta\}$ is consistent, due to the syntactic restrictions on AICs and the fact that all rules are evaluated in the same context.

▶ **Example 32.** The inclusion in Lemma 31 is, in general, strict: consider $DB = \emptyset, \mathcal{U} = \{+a\}$, and let $\eta = \{a, \neg b \supset -a \mid +b\}$. Then $\mathcal{N}(\eta)$ contains the two AICs $a, \neg b \supset -a$ and $a, \neg b \supset +b$. In this case, $\mathcal{T}_\eta(\mathcal{U}) = \{\emptyset, \{+a, +b\}\}$ and $\mathcal{T}_{\mathcal{N}(\eta)}(\mathcal{U}) = \{+b\}$.

▶ **Lemma 33.** *$\mathcal{U}$ is strictly grounded for $\mathcal{T}_\eta$ iff $\mathcal{U}$ is strictly grounded for $\mathcal{T}_{\mathcal{N}(\eta)}$.*

A fixpoint of a non-deterministic operator $\mathcal{O} : L \to \wp(L)$ is a value $x \in L$ such that $x \in \mathcal{O}(L)$ (see e.g. [16]). From the definition of $\mathcal{T}_\eta$, it is immediate that $\mathcal{U} \in \mathcal{T}_\eta(\mathcal{U})$ iff $\mathcal{T}(\mathcal{U}) = \{\mathcal{U}\}$. Furthermore, Lemmas 11 and 12 still hold in this non-deterministic case, allowing us to derive the following consequence of the previous lemma.

▶ **Corollary 34.** *$\mathcal{U}$ is a grounded repair for $\langle DB, \eta\rangle$ iff $\mathcal{U}$ is a grounded repair for $\langle DB, \mathcal{N}(\eta)\rangle$.*

Since repairs, founded repairs and well-founded repairs for η and for $\mathcal{N}(\eta)$ also coincide, we immediately obtain generalizations of Lemma 16 for the general setting, and the parallelization and independence results from § 4 also apply.

As observed in [7], normalization does not preserve justified repairs. Therefore, Lemma 19 does not guarantee that justified repairs are always grounded in the general case. Indeed, the next example shows that this is *not* true.

▶ **Example 35.** Let $DB = \emptyset$ and take η to be the following set of AICs.

$$a, b, \neg c \supset -a \mid -b \mid +c \quad (1) \qquad a, \neg b \supset -a \quad (3) \qquad \neg a, b, c \supset +a \mid -b \mid -c \quad (5)$$
$$\neg a, b, \neg c \supset +a \mid -b \mid +c \quad (2) \qquad a, \neg b, c \supset +b \mid -c \quad (4) \qquad \neg a, \neg b, c \supset +a \mid +b \mid -c \quad (6)$$
$$\neg a, \neg b, \neg c \supset +a \mid +b \mid +c \quad (7)$$

Then $\mathcal{U} = \{+a, +b, +c\}$ is the only repair for $\langle DB, \eta\rangle$, and it is justified. Indeed, if $\mathcal{V} \subseteq \mathcal{U}$ is such that $\mathcal{V} \cup \mathsf{neff}_{DB}(\mathcal{U})$ is closed under η, then $\mathcal{V}$ must contain an action in the head of each of rules (1), (2), (5), (6) and (7). Since $\mathcal{V} \subseteq \mathcal{U}$, it follows that $+c \in \mathcal{V}$ (by (1)) and that $+a \in \mathcal{V}$ (by (5)). But then $\mathcal{V}$ contains the actions corresponding to the non-updatable literals in rule (4) (namely, $+a$), and hence also $+b \in \mathcal{V}$, so $\mathcal{V} = \mathcal{U}$.

However, $\mathcal{U}$ is not a strictly grounded fixpoint of $\mathcal{T}$: taking $\mathcal{V} = \{+a\}$, we see that the only rule applicable in $\mathcal{V}(DB)$ is rule (3), and thus $\mathcal{T}(\mathcal{V}) = \{\emptyset\}$, from which trivially $(\bigcup \mathcal{T}(\mathcal{V})) \cap \mathcal{U} \subseteq \mathcal{V}$.

An examination of the conditions under which $\langle DB, \eta\rangle$ may admit a justified repair that is not strictly grounded shows that this example is among the simplest possible. It is important to point out that $\mathcal{U}$ is also not a justified repair for $\langle DB, \mathcal{N}(\eta)\rangle$, either, which seems to suggest that origin of the problem lies in the standard interpretation of AICs with non-singleton heads. We plan to look further into the semantics of repairs for non-normal AICs in future work.

6 Conclusions and Future Work

We have presented a formalization of the theory of active integrity constraints in lattice theory, by showing how a set of AICs η over a database DB induces an operator $\mathcal{T}_\eta^{DB}$ over a suitably defined lattice of database repairs. We characterized the standard notions of (weak) repairs, founded and well-founded repairs in terms of this operator. By studying the grounded fixpoints of $\mathcal{T}_\eta^{DB}$ in the normalized case, we showed that we obtain a notion

of repair that is stricter than founded or well-founded repairs, but more general than the problematic notion of justified repairs. Furthermore, by suitably extending the notions of grounded and strictly grounded fixpoint of a lattice operator to the non-deterministic case, we gained a general notion of grounded repair also in the non-normalized case. We also showed that grounded repairs are preserved under normalization, and that they share the parallelization and stratification properties of founded and justified repairs that are important for their practical applications.

Conversely, we were able to state some of the results in the database setting more generally. Thus, not only did we propose an extension of the notion of (strictly) grounded fixpoint to the case of non-deterministic lattice operators, but we also defined what it means for an operator to be parallelizable, and showed that several properties of parallelizable operators are not specific to the database case.

We believe the concept of grounded repair to be the one that better captures our intuitions on what a "good" repair is, in the framework of AICs. We plan to use this notion as the basis for future work on this topic, namely concerning the extension of AICs to more general knowledge representation formalisms, following the proposals in [11].

References

1. Serge Abiteboul. Updates, a new frontier. In Marc Gyssens, Jan Paredaens, and Dirk van Gucht, editors, *ICDT*, volume 326 of *LNCS*, pages 1–18. Springer, 1988.
2. Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison Wesley, 1995.
3. Grigoris Antoniou. A tutorial on default logics. *ACM Computing Surveys*, 31(3):337–359, 1999.
4. Catriel Beeri and Moshe Y. Vardi. The implication problem for data dependencies. In *Colloquium on Automata, Languages and Programming*, pages 73–85, London, UK, 1981. Springer.
5. Bart Bogaerts, Joost Vennekens, and Marc Denecker. Grounded fixpoints and their applications in knowledge representation. *Artif. Intell.*, 224:51–71, 2015.
6. Luciano Caroprese, Sergio Greco, Cristina Sirangelo, and Ester Zumpano. Declarative semantics of production rules for integrity maintenance. In Sandro Etalle and Miroslaw Truszczynski, editors, *ICLP*, volume 4079 of *LNCS*, pages 26–40. Springer, 2006.
7. Luciano Caroprese and Miroslaw Truszczynski. Active integrity constraints and revision programming. *Theory Pract. Log. Program.*, 11(6):905–952, November 2011.
8. Luís Cruz-Filipe. Optimizing computation of repairs from active integrity constraints. In Christoph Beierle and Carlo Meghini, editors, *FoIKS*, volume 8367 of *LNCS*, pages 361–380. Springer, 2014.
9. Luís Cruz-Filipe, Patrícia Engrácia, Graça Gaspar, and Isabel Nunes. Computing repairs from active integrity constraints. In Hai Wang and Richard Banach, editors, *TASE*, pages 183–190. IEEE, July 2013.
10. Luís Cruz-Filipe, Michael Franz, Artavazd Hakhverdyan, Marta Ludovico, Isabel Nunes, and Peter Schneider-Kamp. repAIrC: A tool for ensuring data consistency by means of active integrity constraints. In Ana L.N. Fred, Jan L.G. Dietz, David Aveiro, Kecheng Liu, and Joaquim Filipe, editors, *KMIS*, pages 17–26. SciTePress, 2015.
11. Luís Cruz-Filipe, Isabel Nunes, and Peter Schneider-Kamp. Integrity constraints for general-purpose knowledge bases. In Marc Gyssens and Guillermo Ricardo Simari, editors, *FoIKS*, volume 9616 of *LNCS*, pages 235–254. Springer, 2016.
12. Thomas Eiter and Georg Gottlob. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artif. Intell.*, 57(2–3):227–270, 1992.

13 Melvin Fitting. Fixpoint semantics for logic programming: a survey. *Theor. Comput. Sci.*, 278(1–2):25–51, 2002.

14 Sergio Flesca, Sergio Greco, and Ester Zumpano. Active integrity constraints. In Eugenio Moggi and David Scott Warren, editors, *PPDP*, pages 98–107. ACM, 2004.

15 Ahmed Guessoum. Abductive knowledge base updates for contextual reasoning. *J. Intell. Inf. Syst.*, 11(1):41–67, 1998.

16 Mohammed A. Khamsi, Vladik Kreinovich, and Driss Misane. A new method of proving the existence of answer sets for disjunctive logic programs. In *Proceedings of the Workshop on Logic Programming with Incomplete Information*, 1993.

17 V. Wiktor Marek and Miroslav Truszczynski. Revision programming, database updates and integrity constraints. In Georg Gottlob and Moshe Y. Vardi, editors, *ICDT*, volume 893 of *LNCS*, pages 368–382. Springer, 1995.

18 Enric Mayol and Ernest Teniente. Addressing efficiency issues during the process of integrity maintenance. In Trevor J.M. Bench-Capon, Giovanni Soda, and A Min Tjoa, editors, *DEXA*, volume 1677 of *LNCS*, pages 270–281. Springer, 1999.

19 Shamim A. Naqvi and Ravi Krishnamurthy. Database updates in logic programming. In Chris Edmondson-Yurkanan and Mihalis Yannakakis, editors, *PODS*, pages 251–262. ACM, 1988.

20 Teodor C. Przymusinski and Hudson Turner. Update by means of inference rules. *J. Log. Program.*, 30(2):125–143, 1997.

21 Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

22 Ernest Teniente and Antoni Olivé. Updating knowledge bases while maintaining their consistency. *VLDB J.*, 4(2):193–241, 1995.

23 Jennifer Widom and Stefano Ceri, editors. *Active Database Systems: Triggers and Rules For Advanced Database Processing*. Morgan Kaufmann, 1996.

24 Marianne Winslett. *Updating Logical Databases*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1990.